Event-B Course

10. An Access Control System

Jean-Raymond Abrial

September-October-November 2011

Purpose of this Lecture

- To study again a complete system (like Car or Press)

- To encounter some interesting data structure

- To exercise ourselves in the human reasoning while making the formal development

- To see an example of decomposition of formal models

The System: Requirement Document (1)

- To control the accesses of persons to locations of a workspace.

The system concerns people and locations	FUN-1

- It is based on permanent authorization given to people

People are permanently assigned the right to access certain locations only	FUN-2

The System: Requirement Document (2)

- We want to be sure that people which are present in a location are authorized to do so

A person which is in a location must be authorized to be there	FUN-3

- This requirement is the main purpose of this system

The System: Requirement Document (3)

- People are identified by means of magnetic cards

Each person receives a personal magnetic card	EQP-1

- For entering into a location people put their card in the fence of a card reader

Each entrance and exit of a location is equipped with a card reader	EQP-2

The System: Requirement Document (4)

- Card readers are equipped with two lamps: one green and one red

- When a person puts his card in the fence, then one lamp is lit

- When the green lamp is lit, it means the person is accepted

- When the red lamp is lit, it means the person is not accepted

Each card reader has two lamps: one green light and one red light.	EQP-3

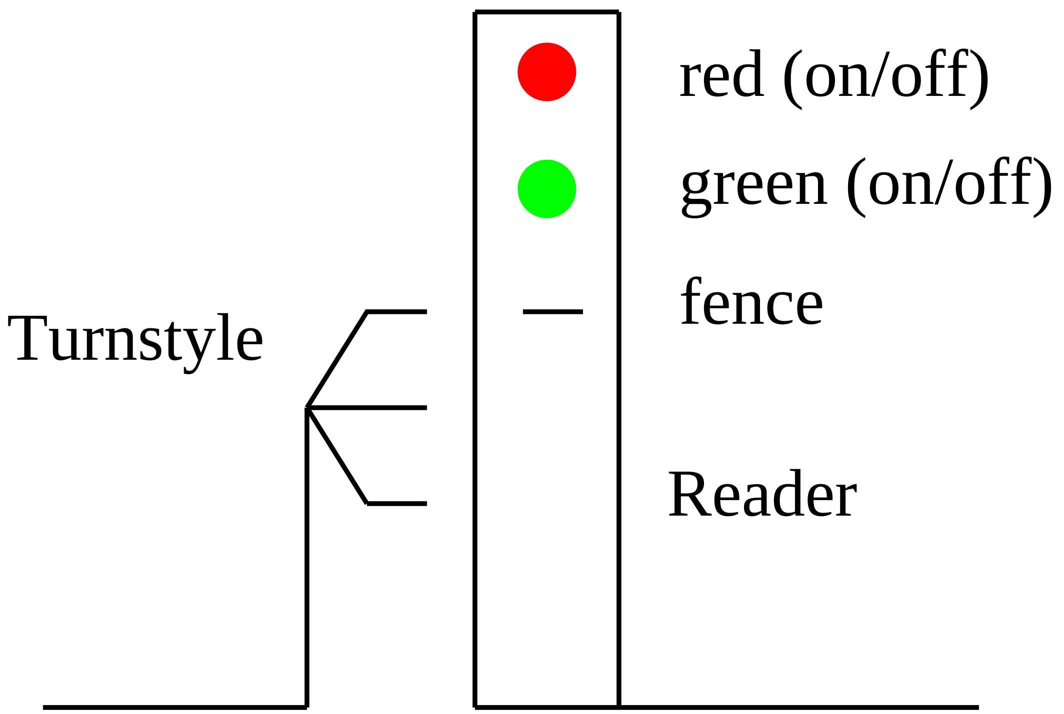
The System: Requirement Document (5)

Each lamp has two status

Each light can be “on” of “off”.	EQP-4

Each door is equipped with a turnstile which works one way only

Locations communicate via one-way turnstiles	EQP-5



BASIC

EQUIPMENT

When nobody is willing to move from one location to another, the corresponding turnstile is blocked

Turnstiles are normally blocked	FUN-4

In order to change location, a person first put his card in the fence of the corresponding card reader

A person willing to pass through a turnstile puts its card in the fence of the card reader	FUN-5

Access Protocol (after introducing card in reader)

- If access is permitted
  - green light is turned on
  - turnstile is unblocked for 30 sec

- Passing, or 30 sec elapsed
  - green light is turned off
  - turnstile is blocked again

- If access is refused
  - red light is turned on for 2 sec
  - turnstile stays blocked

More Functional Requirements (1)

If the person is accepted, the green light is lit and the turnstile is unblocked for at most 30 seconds.	FUN-6

If the person is not accepted, the red light is lit for 2 seconds and the turnstile remains blocked	FUN-7

More Functional Requirements (2)

As soon as an accepted person has gone through an unblocked turnstile, the green light is turned off and the turnstile is blocked again.	FUN-8

If nobody goes through an unblocked turnstile during the 30 seconds period, the green light is turned off and the turnstile is blocked again.	FUN-9

Problems to be Solved During the Design (1)

- Many problems have not been solved in the requirements

- Sharing of control between Hardware and Software
    - A computer in each card reader?
    - A unique centralized computer?
    - A mixed situation with some “intelligence” in the card reader?

Problems to be Solved During the Design (2)

- Precise behavior of the equipment
  - Does the turnstile block itself after lamps are turned off?
  - Or does the turnstile wait for an order to do so?
  - Does the lamp system of each card reader have a local clock?
  - Is the fence obstructed after inserting a card into it?

- Answering these questions is important

- It will allow us to define the precise spec of the equipment we buy

Problems to be Solved During the Design (3)

- Tackling safety questions
    - The Requirement Document says nothing on this
    - Is it important or not?
    - If it is important, what are the precise safety questions?
    - Should we extend the Requirement Document?

Problems to be Solved During the Design (4)

- Synchronization problems
    - Requirements say nothing about the precise timing
    - Synchronization between the lamps and the turnstile
    - Which one comes first?
    - Is it important to know that?

Problems to be Solved During the Design (5)

- Functioning at the limits
  - Again, it is not treated in the Requirements
  - Introducing several cards successively into green card reader?
  - Introducing the same card quickly into different card readers?
  - Strange behavior of people must not be excluded

Refinement Strategy

-Intitial model: Persons and locations

- 1st refinement: Communicatons between locations

- 2nd refinement: Doors

- 3rd refinement: Card readers

- 4th refinement: Lights and turnstile

- Decomposition

Initial Model: the state

- We introduce:

  - The two carrier sets of persons, P, and locations, L

  - The constant authorization, aut, as a relation between P and L

  - The variable, sit, denoting where a person is

Initial Model: the state (cont'd)

- A person cannot be in two locations at a time

- Therefore sit is a function from P to L

- Is it a partial or a total function? Would be nice to have it total

- We introduce a special constant "location", out, for outside

- Everyone is authorized to be in out

- the variable, sit, is therefore a total function

carrier sets: P, L

constants: aut, out

axm0_1: $aut \in P \leftrightarrow L$

axm0_2: $out \in L$

axm0_3: $P \times \{out\} \subseteq aut$

variables: sit

inv0_1: $sit \in P \rightarrow L$

inv0_2: $sit \subseteq aut$

Initial Model: the events

init
$$sit := P \times \{out\}$$

pass
any p, l **where**
$p \mapsto l \in aut$
$sit(p) \neq l$
then
$sit(p) := l$
end

- It is still very abstract

- We do not know how people go from one location to another

Example

Sets

$$\text{persons} = \{\,\text{p1},\ \text{p2},\ \text{p3}\,\}$$
$$\text{locations} = \{\,\text{l1},\ \text{l2},\ \text{l3},\ \text{out}\,\}$$

Authorizations

p1	l2, out
p2	l1, l3, out
p3	l2, l3, out

Correct scenario

p1	out
p2	out
p3	out

→

p1	l2
p2	out
p3	out

→

p1	l2
p2	l1
p3	out

→

p1	out
p2	l1
p3	out

→

p1	out
p2	l1
p3	l3

What we Have to Prove

- init establishes the invariant (easy)

- pass maintains the invariant (easy)

- Deadlock freeness

Axioms of constants	$aut \in P \leftrightarrow L$
	$out \in L$
	$P \times \{out\} \subseteq aut$
Invariant	$sit \in P \rightarrow L$
	$sit \subseteq aut$
$\vdash$	$\vdash$
Disjunction of Guards	$\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ sit(p) \neq l \end{pmatrix}$

- This cannot be proved

- We know that each person has the authorization to be in out

- We have to say now that each person also has the authorization to be in a location different from out

$$\textbf{axm0\_4:} \quad \forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \end{array} \right)$$

What Remains to be Proved for Deadlock Freeness

$$
\begin{array}{l}
P \neq \varnothing \\
P \times \{out\} \;\subseteq\; aut \\
\forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \end{array} \right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array} \right)
\end{array}
\qquad
\begin{array}{l}
\exists q \cdot (\, q \;\in\; P \,) \\
P \times \{out\} \;\subseteq\; aut \\
\forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \end{array} \right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array} \right)
\end{array}
$$

- We replace $P \neq \varnothing$ by $\exists q \cdot (\, q \;\in\; P \,)$

Deadlock Freeness Proof (1)

$$
\begin{array}{l}
\exists q \cdot (\, q \;\in\; P \,) \\
P \times \{out\} \;\subseteq\; aut \\
\forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array}\right) \end{array}\right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array}\right)
\end{array}
\qquad
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
\forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array}\right) \end{array}\right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array}\right)
\end{array}
$$

- We eliminate the first existential quantification.

Deadlock Freeness Proof (2)

$$
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
\forall p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists\, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \end{array} \right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array} \right)
\end{array}
$$

$$
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
\exists\, l \cdot \left(\begin{array}{l} q \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \\
\vdash \\
\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{array} \right)
\end{array}
$$

- We instantiate the quantified variable p with q

Deadlock Freeness Proof (3)

$$
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
\exists\, l \cdot \begin{pmatrix} q \mapsto l \;\in\; aut \\ l \neq out \end{pmatrix} \\
\vdash \\
\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{pmatrix}
\end{array}
\qquad
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
q \mapsto l \;\in\; aut \\
l \neq out \\
\vdash \\
\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{pmatrix}
\end{array}
$$

- We eliminate the first existential quantification

- We envisage two cases: $\begin{cases} sit(q) \neq l \\ \\ sit(q) = l \end{cases}$

First case: $sit(q) \neq l$

$$
\begin{array}{l}
sit(q) \neq l \\
q \ \in \ P \\
P \times \{out\} \ \subseteq \ aut \\
q \mapsto l \ \in \ aut \\
l \neq out \\
\vdash \\
\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \ \in \ aut \\ sit(p) \neq l \end{pmatrix}
\end{array}
\qquad\qquad
\begin{array}{l}
sit(q) \neq l \\
q \ \in \ P \\
P \times \{out\} \ \subseteq \ aut \\
q \mapsto l \ \in \ aut \\
l \neq out \\
\vdash \\
q \mapsto l \ \in \ aut \\
sit(q) \neq l
\end{array}
$$

- We propose q and l as witnesses

Second case: $sit(q) = l$

$$\begin{array}{l} sit(q) = l \\ q \;\in\; P \\ P \times \{out\} \;\subseteq\; aut \\ q \mapsto l \;\in\; aut \\ l \neq out \\ \vdash \\ \exists\, p, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ sit(p) \neq l \end{pmatrix} \end{array}$$

$$\begin{array}{l} sit(q) = l \\ q \;\in\; P \\ P \times \{out\} \;\subseteq\; aut \\ q \mapsto l \;\in\; aut \\ l \neq out \\ \vdash \\ q \mapsto out \;\in\; aut \\ sit(q) \neq out \end{array}$$

- We propose q and out as witnesses

Second case: $sit(q) = l$ (cont'd)

$$
\begin{array}{l}
sit(q) = l \\
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
q \mapsto l \;\in\; aut \\
l \neq out \\
\vdash \\
q \mapsto out \;\in\; aut \\
sit(q) \neq out
\end{array}
$$

$$
\begin{array}{l}
q \;\in\; P \\
P \times \{out\} \;\subseteq\; aut \\
q \mapsto l \;\in\; aut \\
l \neq out \\
\vdash \\
q \mapsto out \;\in\; aut \\
l \neq out
\end{array}
$$

- We apply equality

- We introduce the communication between the locations

- This is done by means of a binary relation, com, built on locations

- com is irreflexive: a location does not communicate with itself

Reminder: Some Classical Relations

Given a relation r defined on a set S: $r \in S \leftrightarrow S$

r is reflexive:	$\mathrm{id}(\mathrm{dom}(r)) \subseteq r$
r is irreflexive:	$r \cap \mathrm{id}(S) = \varnothing$
r is symmetric:	$r = r^{-1}$
r is transitive:	$r\,;r \subseteq r$
r is anti-symmetric:	$r \cap r^{-1} \subseteq \mathrm{id}(S)$

carrier sets: P, L

constants: aut, out, com

variables: sit

axm1_1: $com \in L \leftrightarrow L$

axm1_2: $com \cap \mathrm{id}(L) = \varnothing$

init

$$sit := P \times \{out\}$$

pass

any p, l **where**

$$p \mapsto l \in aut$$

$$sit(p) \mapsto l \in com$$

then

$$sit(p) := l$$

end

What we Have to Prove

- Event init refines its abstraction (easy)

- Event pass refines its abstraction (easy)
    - Guard strengthening
    - Refinement of B-A predicate

- Deadlock freeness

(abstract-)pass
any p, l **where**
$p \mapsto l \in aut$
$sit(p) \neq l$
then
$sit(p) := l$
end

(concrete-)pass
any p, l **where**
$p \mapsto l \in aut$
$sit(p) \mapsto l \in com$
then
$sit(p) := l$
end

axm1_1: $com \in L \leftrightarrow L$

axm1_2: $com \cap \mathrm{id}(L) = \varnothing$

Axioms of constants

$$aut \in P \leftrightarrow L$$
$$out \in L$$
$$P \times \{out\} \subseteq aut$$

Invariant

$$sit \in P \rightarrow L$$
$$sit \subseteq aut$$

Gluing Invariant

Abstract Guard

$$\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ sit(p) \neq l \end{pmatrix}$$

$\vdash$

Concrete Guard

$$\vdash$$

$$\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ sit(p) \mapsto l \in com \end{pmatrix}$$

No More Deadlock in Concrete Space: Counter-example

$$P = \{p\}$$

$$L = \{out, l\}$$

$$aut = \{p \mapsto out, p \mapsto l\}$$

$$com = \{out \mapsto l\}$$

- In abstract space, p can go from out to l and vice-versa

- In concrete space, p can go from out to l and remains blocked in l

- This is because l does not communicate with out

Adding a Safety Requirement

No person must remain blocked in a location	SAF-1

- In abstract space we have (**axm0_4**)

$$\forall\, p \cdot \left(\begin{array}{l} p \;\in\; P \\ \Rightarrow \\ \exists l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ l \neq out \end{array} \right) \end{array} \right)$$

- And we already proved (this is deadlock freeness in abstraction)

$$\exists\, p, l \cdot \left(\begin{array}{l} p \mapsto l \;\in\; aut \\ sitp(p) \neq l) \end{array} \right)$$

No more deadlock in concrete space (1)

- We have then just to prove

$$\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ sitp(p) \mapsto l \;\in\; com \end{pmatrix}$$

- This could be re-written as follows

$$\exists\, p, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ \exists\, m \cdot \begin{pmatrix} p \mapsto m \;\in\; sit \\ m \mapsto l \;\in\; com \end{pmatrix} \end{pmatrix}$$

- This is equivalent to

$$\exists\, p, m \cdot \begin{pmatrix} p \mapsto m \;\in\; sit \\ \exists\, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ l \mapsto m \;\in\; com^{-1} \end{pmatrix} \end{pmatrix}$$

No more deadlock in concrete space (2)

$$\exists\, p, m \cdot \begin{pmatrix} p \mapsto m \;\in\; sit \\ \exists\, l \cdot \begin{pmatrix} p \mapsto l \;\in\; aut \\ l \mapsto m \;\in\; com^{-1} \end{pmatrix} \end{pmatrix}$$

- That is

$$(aut; com^{-1}) \;\cap\; sit \neq \varnothing$$

- It is sufficient to prove

$$sit \;\subseteq\; (aut; com^{-1})$$

No more deadlock in concrete space (3)

$$sit \;\subseteq\; (aut; com^{-1})$$

- This can be developed as follows

$$\forall p, m \cdot \begin{pmatrix} p \mapsto m \;\in\; sit \\ \Rightarrow \\ \exists\, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ l \mapsto m \;\in\; com^{-1} \end{pmatrix} \end{pmatrix}$$

- that is

$$\forall p \cdot \exists\, l \cdot (\, p \mapsto l \in aut \;\;\wedge\;\; sit(p) \mapsto l \in com \,)$$

- p is authorized to go in a certain l communicating with $sit(p)$

No more deadlock in concrete space (4)

- For proving this

$$sit \subseteq aut ; com^{-1}$$

- It is sufficient to prove (since $sit \subseteq aut$ according to **inv0_2**)

$$\boxed{aut \subseteq aut ; com^{-1}}$$

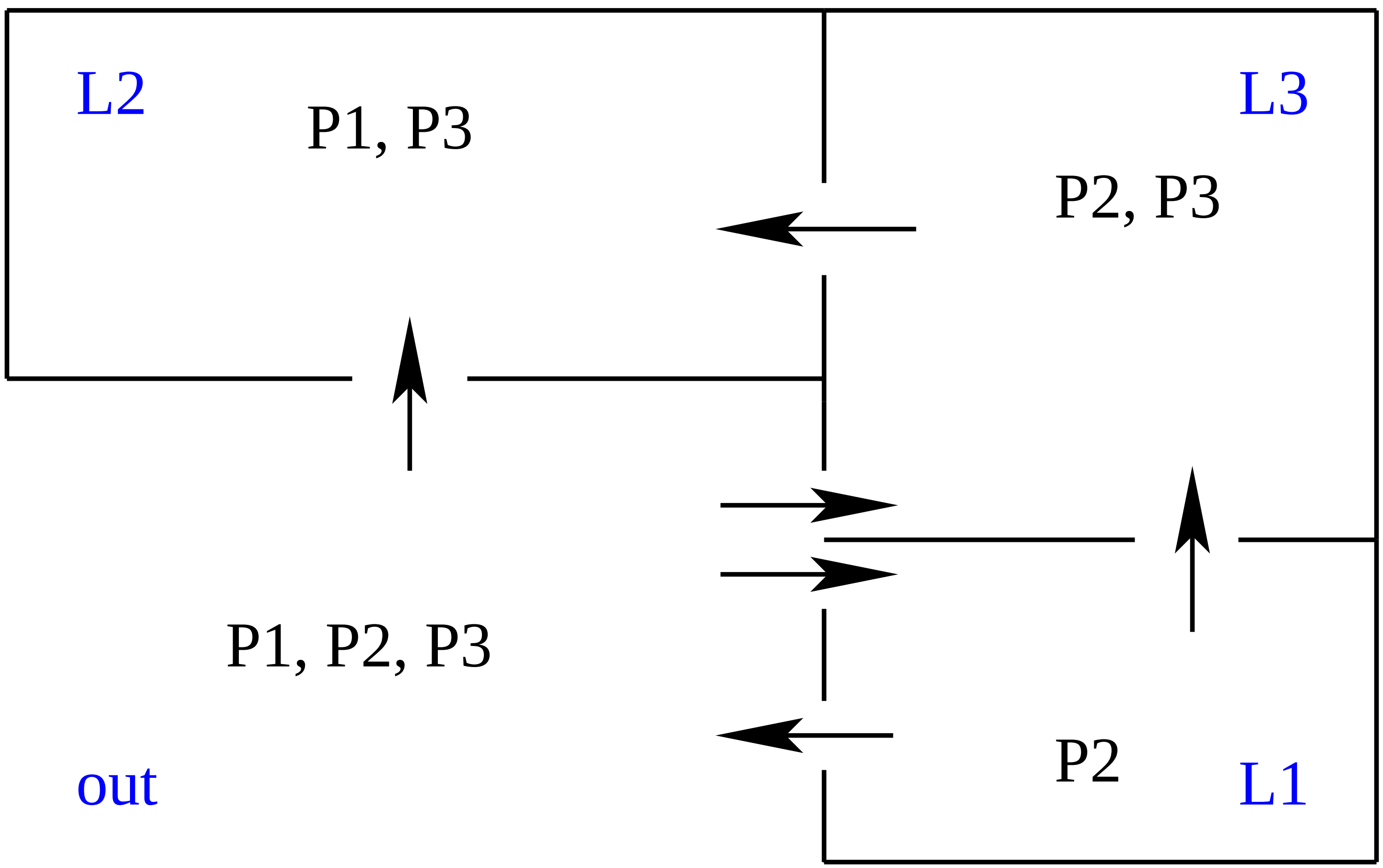
- That is

$$\forall p, l \cdot (p \mapsto l \in aut \Rightarrow \exists m \cdot (p \mapsto m \in aut \wedge l \mapsto m \in com))$$

$$\forall\, p, l \cdot (\, p \mapsto l \in aut \;\;\Rightarrow\;\; \exists\, m \cdot (\, p \mapsto m \in aut \;\wedge\; l \mapsto m \in com \,)\,)$$

This can be translated in English as follows:

Any person authorized to be in a location must also be authorized to go in another location which communicates with the first one.	SAF-2



L2
P1, P3
L3
P2, P3
P1, P2, P3
P2
out
L1

Example

p1	l2	p2	out
p1	out	p3	l2
p2	l1	p3	l3
p2	l3	p3	out

aut

l1	l3
l1	out
l3	l2
out	l1
out	l2
out	l3

com

l1	out
l2	l3
l2	out
l3	l1
l3	out
out	l1

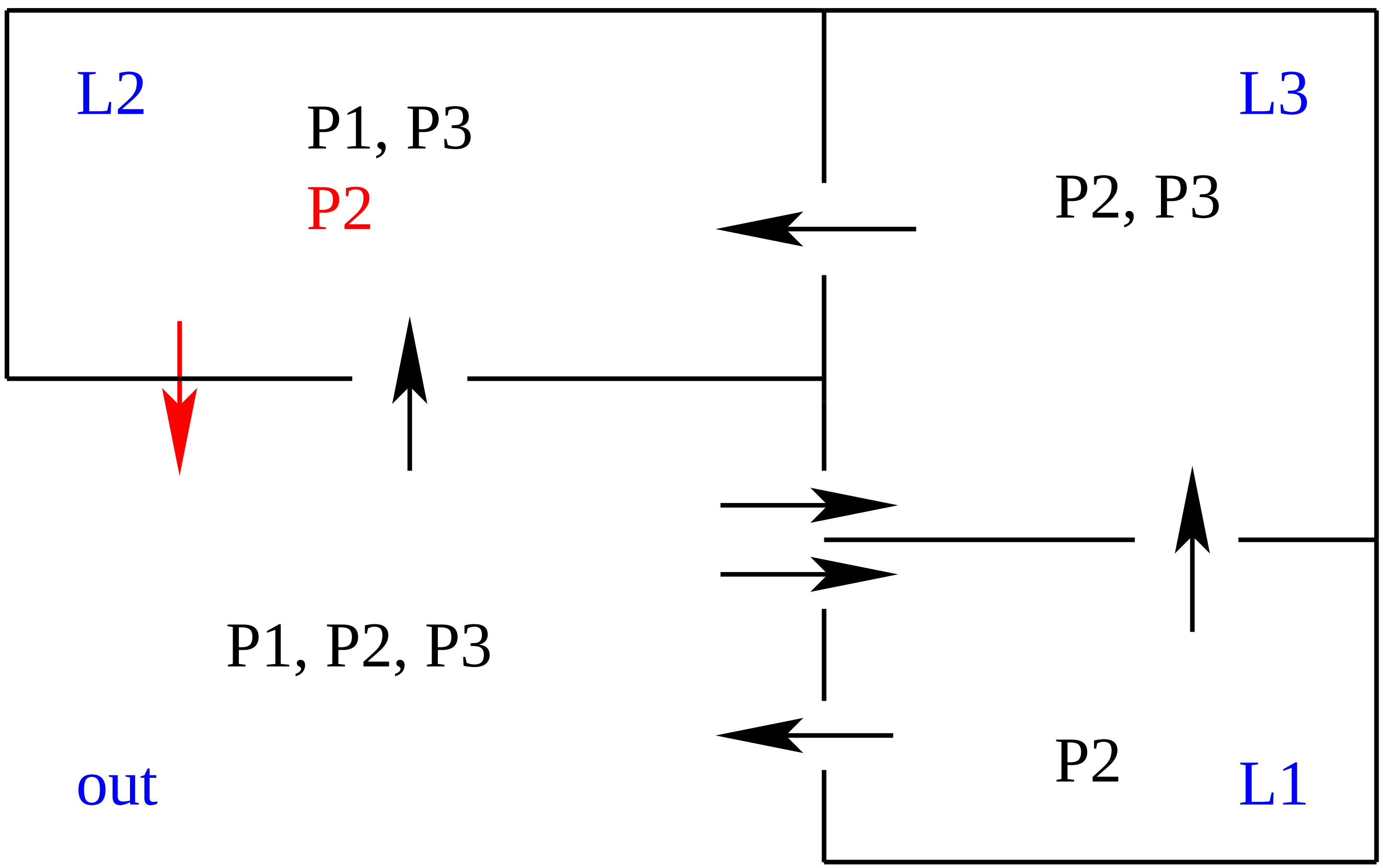
com^{-1}

p1	l1	p2	out
p1	l3	p3	l1
p1	out	p3	l3
p2	l1	p3	out

$aut; com^{-1}$

- Opening a door between l2 and out

- Authorizing p2 to go to l2


L2
P1, P3
P2
L3
P2, P3
P1, P2, P3
out
P2
L1

Solution

p1	l2	p2	out
p1	out	p3	l2
p2	l1	p3	l3
p2	l2	p3	out
p2	l3		

aut

l1	l3
l1	out
l2	out
l3	l2
out	l1
out	l2
out	l3

com

l1	out
l2	l3
l2	out
l3	l1
l3	out
out	l1
out	l2

com^{-1}

p1	l1	p2	l3
p1	l2	p2	out
p1	l3	p3	l1
p1	out	p3	l2
p2	l1	p3	l3
p2	l2	p3	out

$aut; com^{-1}$

A More Adequate Requirement

- Moving to another location is not sufficient

- We want people being able to reach “outside”

- For this, we introduce an “exit” sign in each location (except out)

- the constant $exit$ is defined everywhere except at out

- $exit$ is a function which must be compatible with com

- Every person must be entitled to follow $exit$

Extensions of the constants

carrier sets: P, L

constants: $aut, out, com, exit$

variables: sit

axm1_3 : $exit \in L \setminus \{out\} \rightarrow L$

axm1_4 : $exit \subseteq com$

axm1_5 : $aut \rhd \{out\} \subseteq aut \,;\, exit^{-1}$

Any 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 outside.	SAF-3

$$aut \rhd \{out\} \subseteq aut \,;\, exit^{-1}$$

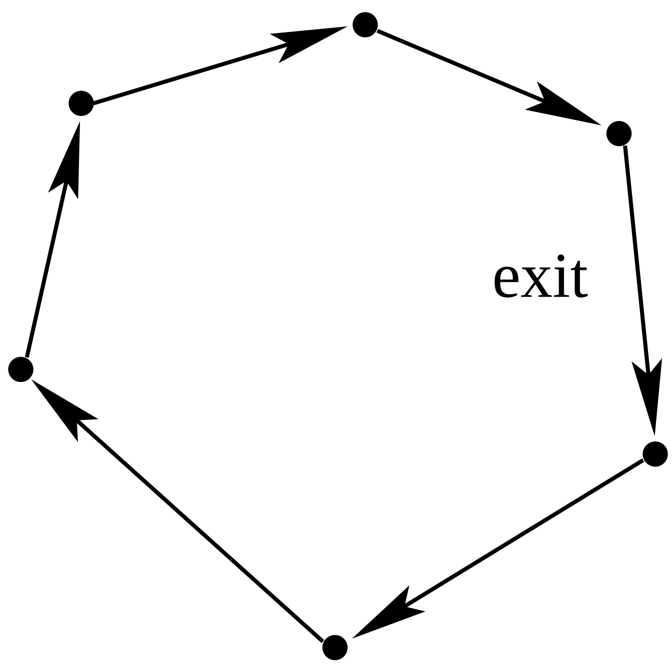
$$\forall\, p, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ l \neq out \\ \Rightarrow \\ \exists\, m \cdot (\, p \mapsto m \in aut \quad \land \quad l \mapsto m \in exit\,) \end{pmatrix}$$

$$\forall\, p, l \cdot \begin{pmatrix} p \mapsto l \in aut \\ l \neq out \\ \Rightarrow \\ p \mapsto exit(l) \in aut \end{pmatrix}$$

- Every person p authorized to be in l (except out) must be authorized to go in $exit(l)$. Is it sufficient?

- Being able to follow the exit sign is not sufficient

- We must be sure that following the exit sign does not put us in a cycle



- Suppose that we have a set s of locations forming a cycle with $exit$

- It means that for any l in s then $exit(l)$ is also in s

$$\forall\, l \cdot (\, l \;\in\; s \;\;\Rightarrow\;\; exit(l) \;\in\; s\,)$$

$$\forall\, l \cdot (\, l \;\in\; s \;\;\Rightarrow\;\; l \;\in\; exit^{-1}[s]\,)$$

$$s \;\subseteq\; exit^{-1}[s]$$

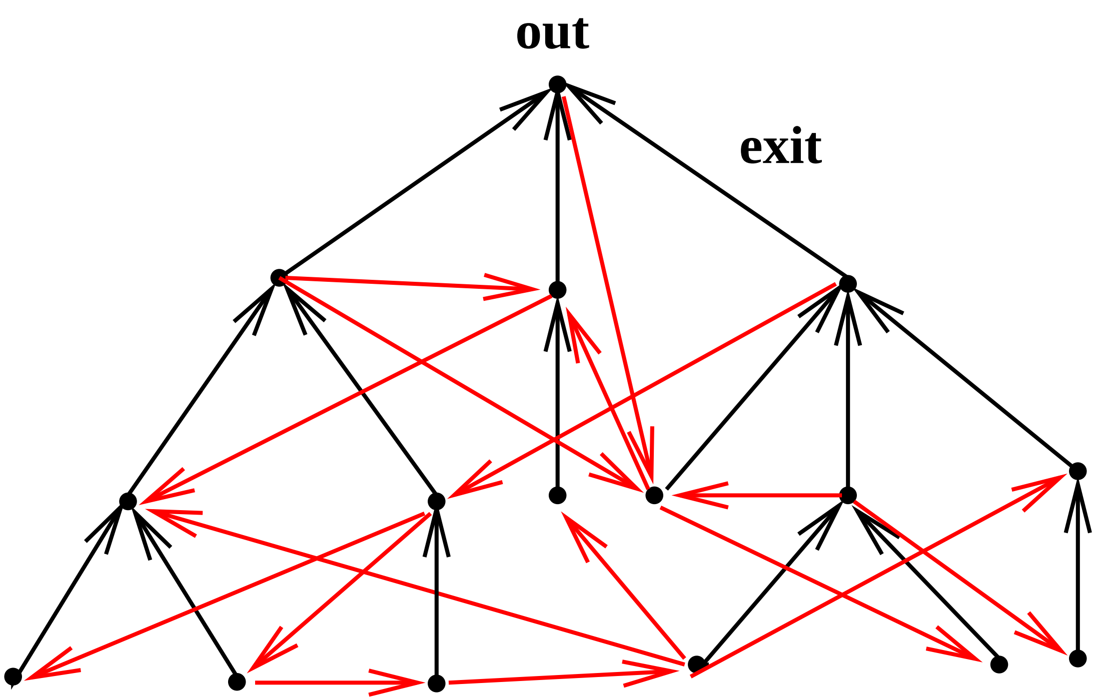
- We want to preclude cycles, therefore the only set with that property must be the empty set

axm1_6 :
$$\forall s \cdot \begin{pmatrix} s \subseteq L \\ s \subseteq exit^{-1}[s] \\ \Rightarrow \\ s = \varnothing \end{pmatrix}$$

- Note that this property is equivalent to the following

$$\forall t \cdot \begin{pmatrix} t \subseteq L \\ out \ \in \ t \\ exit^{-1}[t] \subseteq t \\ \Rightarrow \\ L \subseteq t \end{pmatrix}$$

- Prop. **axm1_3** to **axm1_6** characterize a tree spanning the graph com



- From any location one can reach out by following the $exit$ sign

$$\exists\, p, l \cdot (\, p \mapsto l \;\in\; aut \;\wedge\; sit(p) \neq l\,)$$
$$\Rightarrow$$
$$\exists\, p, l \cdot (\, p \mapsto l \;\in\; aut \;\wedge\; sit(p) \mapsto l \;\in\; com\,)$$

- Still a problem

- From the inside, people can always go outside

- BUT from outside, people must be able to go inside!!!

axm1_7 : $\forall p \cdot \left(p \in P \;\Rightarrow\; \exists l \cdot \begin{pmatrix} p \mapsto l \subseteq aut \\ l \neq out \\ out \mapsto l \in com \end{pmatrix}\right)$

axm1_7 : $$\forall p \cdot \left(\begin{array}{l} p \in P \\ \Rightarrow \\ \exists l \cdot \left(\begin{array}{l} p \mapsto l \subseteq aut \\ out \mapsto l \in com \end{array} \right) \end{array} \right)$$

axm0_4: $$\forall p \cdot \left(\begin{array}{l} p \in P \\ \Rightarrow \\ \exists l \cdot \left(\begin{array}{l} p \mapsto l \in aut \\ l \neq out \end{array} \right) \end{array} \right)$$

Second Refinement: Doors

- Introducing the doors

- Each door has a location of origin and a location of destination

- Doors must be compatible with the communication

- Each door is assigned to the person who attempts to use it.

- A door could be temporarily green or red

- Each door is equipped with a local clock

Door Assigned to a Person: Injective Function dap

- From the moment a person p is accepted by a door d (event accept)

- To the moment where either:

    - the person passes through the door

    - the 30 sec time has passed (event off_green)

- The door d is uniquely assigned to the person p

- We do not want two doors being assigned to the same person

- We do not want two persons being assigned the same door

Second Refinement: the State (1)

carrier sets: P, L, D

constants: $aut, out, com, exit, org, dst$

variables: sit, dap, grn, red

axm2_1 : $org \in D \rightarrow L$

axm2_2 : $dst \in D \rightarrow L$

axm2_3 : $com = (org^{-1} ; dst)$

inv2_1 : $dap \in P \rightarrowtail\!\!\!\!\!\!\!+\; D$

inv2_2 : $(dap \,;\, org) \subseteq sit$

inv2_3 : $(dap \,;\, dst) \subseteq aut$

inv2_4 : $grn \subseteq D$

inv2_5 : $red \subseteq D$

inv2_6 : $grn = \mathsf{ran}(dap)$

inv2_7 : $grn \cap red = \varnothing$

Second Refinement: the Events (1)

accept
any p, d **where**
$p \in P$
$d \in D$
$d \notin grn \cup red$
$sit(p) = org(d)$
$p \mapsto dst(d) \in aut$
$p \notin \mathrm{dom}\,(dap)$
then
$dap(p) := d$
$grn := grn \cup \{d\}$
end

refuse
any p, d **where**
$p \in P$
$d \in D$
$d \notin grn \cup red$
$\neg\,(\,sit(p) = org(d)$
$p \mapsto dst(d) \in aut$
$p \notin \mathrm{dom}\,(dap)\,)$
then
$red := red \cup \{d\}$
end

off_grn
  any d **where**
    $d \in grn$
  then
    $dap := dap \rhd \{d\}$
    $grn := grn \setminus \{d\}$
  end

off_red
  any d **where**
    $d \in red$
  then
    $red := red \setminus \{d\}$
  end

Second Refinement: the Events (3)

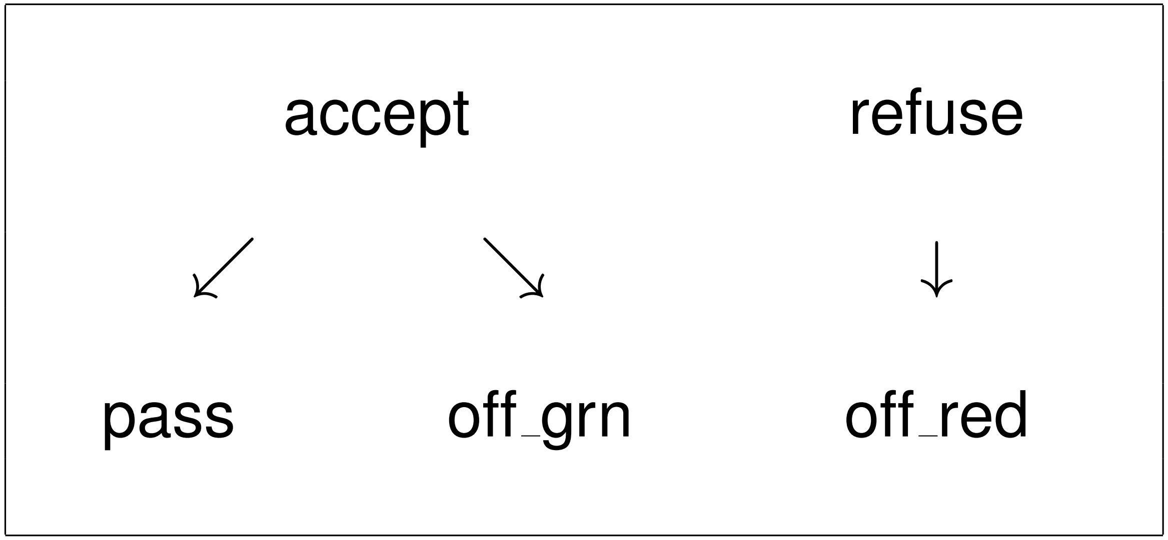
(abstract-)pass
  any p, l **where**
    $p \mapsto l \in aut$
    $sit(p) \mapsto l \in com$
  then
    $sit(p) := l$
  end

(concrete-)pass
  any d **where**
    $d \in grn$
  then
    $sit(dap^{-1}(d)) := dst(d)$
    $dap := dap \rhd \{d\}$
    $grn := grn \setminus \{d\}$
  end

- Witness for refinement:

$$\begin{cases} p = dap^{-1}(d) \\ l = dst(d) \end{cases}$$

Second Refinement: Synchronization



- This diagram shows how the present events are synchronized

- This is done through their guards

Status of Proofs

- Event pass refines its abstraction: success

- New events refine skip: success

- Deadlock freeness: success

- New events cannot take control for ever: failure

- Events accept, refuse, off_grn, off_red can take control for ever

- People with authorization always change mind at the last moment and then retry

- People without authorization always retry

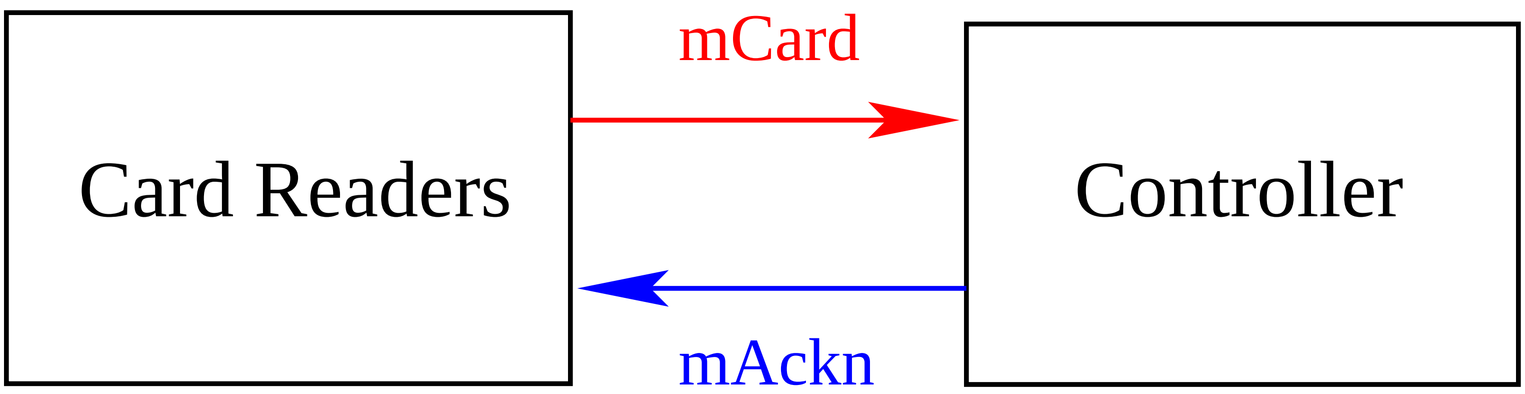
- Detecting people with bad behavior and taking their card
  (like in a cash dispenser)

- Not possible because then people cannot leave the location

- Accepting the (low) risk

- The risk is well understood and accepted by everyone concerned

- Variables $mCard$ and $mAckn$ denote the channels between the card readers and the controller



- A card reader involved with a person is physically blocked until it receives an acknowledgment (DECISION)

carrier sets: P, L, D

constants: aut, out, com, org, dst

variables: sit, dap, grn, red
$BLR, mCard, mAckn$

inv3_1 : $BLR \subseteq D$

inv3_2 : $mCard \in D \nrightarrow P$

inv3_3 : $mAckn \subseteq D$

Third Refinement: the state (2)

inv3_4 : $\mathrm{dom}\,(mCard) \;\cup\; grn \;\cup\; red \;\cup\; mAckn \;=\; BLR$

inv3_5 : $\mathrm{dom}\,(mCard) \;\cap\; (grn \;\cup\; red \;\cup\; mAckn) \;=\; \varnothing$

inv3_6 : $mAckn \;\cap\; (grn \;\cup\; red) \;=\; \varnothing$

Sets $\mathrm{dom}\,(mCard), grn, red, mAckn$ partition the set BLR

Third Refinement: Events (1)

CARD
any p, d **where**
$p \in P$
$d \in D \setminus BLR$
then
$BLR := BLR \cup \{d\}$
$mCard := mCard \cup \{d \mapsto p\}$
end

- This is a physical event

- It corresponds to a person p putting his card in the fence of the unblocked card reader of door d

Third Refinement: Events (2)

accept
any p, d **where**
$d \mapsto p \in mCard$
$sit(p) = org(d)$
$p \mapsto dst(d) \in aut$
$p \notin \mathrm{dom}\,(dap)$
then
$dap(p) := d$
$grn := grn \cup \{d\}$
$mCard := \{d\} \lhd mCard$
end

refuse
any p, d **where**
$d \mapsto p \in mCard$
$\neg\,(\ sit(p) = org(d)$
$p \mapsto dst(d) \in aut$
$p \notin \mathrm{dom}\,(dap)\ )$
then
$red := red \cup \{d\}$
$mCard := \{d\} \lhd mCard$
end

Third Refinement: Events (3)

off_grn
any d **where**
$d \in grn$
then
$dap := dap \rhd\!\!\!> \{d\}$
$grn := grn \setminus \{d\}$
$mAckn := mAckn \cup \{d\}$
end

off_red
any d **where**
$d \in red$
then
$red := red \setminus \{d\}$
$mAckn := mAckn \cup \{d\}$
end

pass

any d **where**

$d \in grn$

then

$sit(dap^{-1}(d)) := dst(d)$

$dap := dap \rhd \{d\}$

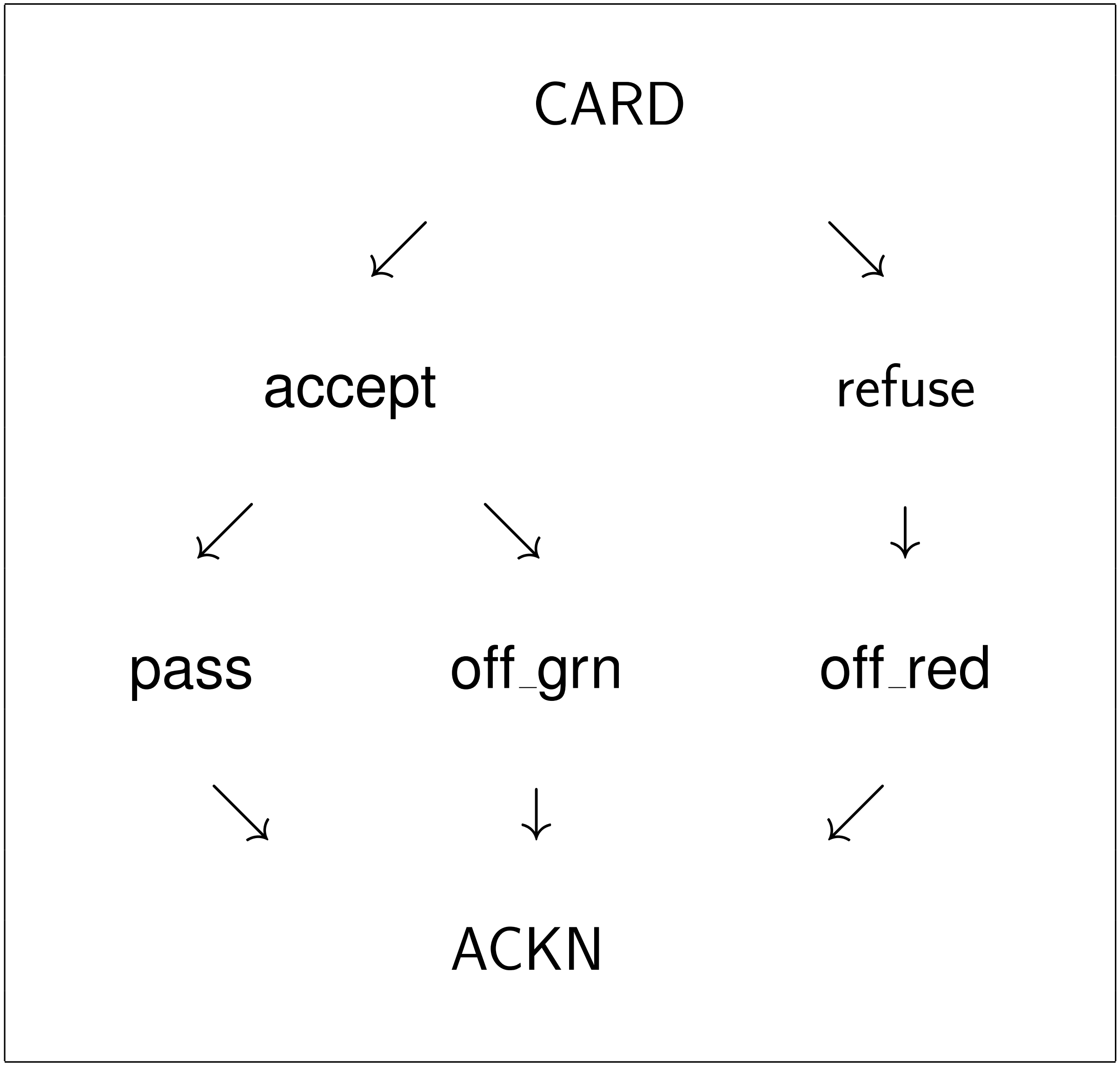
$grn := grn \setminus \{d\}$

$mAckn := mAckn \cup \{d\}$

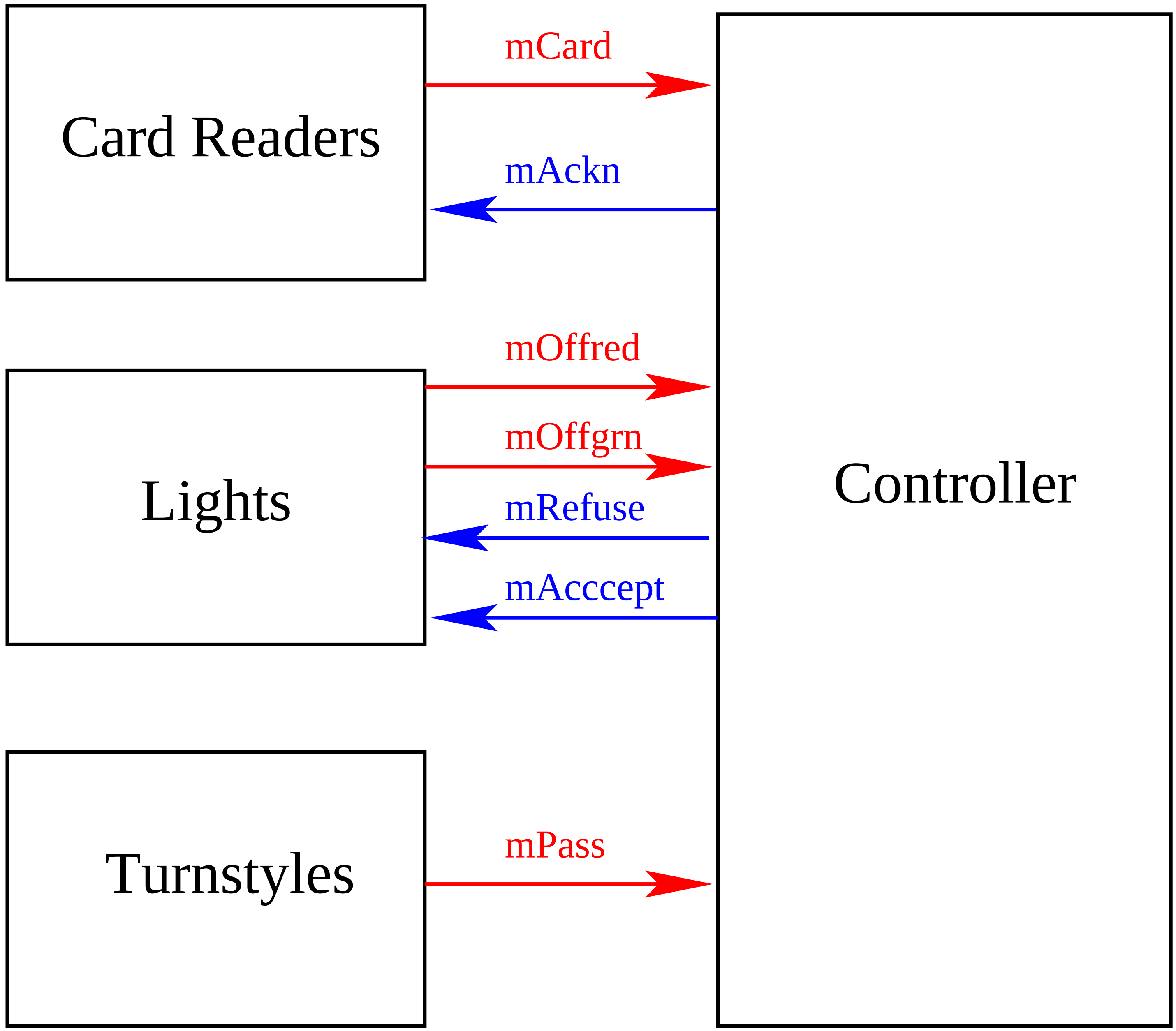
end

ACKN
any d **where**
$d \in mAckn$
then
$BLR := BLR \setminus \{d\}$
$mAckn := mAckn \setminus \{d\}$
end

Synchronization



Fourth Refinement: Lights and Turnstile



Fourth Refinement: the Green Chain (1)

carrier sets: P, L, D

constants: $aut, out, com,$
$exit, org, dst$

variables: $sit, dap, BLR,$
$mCard, mAckn,$
$GRN, mAccept,$
$mOff\_grn,$
$mPass$

inv4_1 : $GRN \subseteq D$

inv4_2 : $mAccept \subseteq D$

inv4_3 : $mPass \subseteq D$

inv4_4 : $mOff\_grn \subseteq D$

Fourth Refinement: the Green Chain (2)

inv4_5 : $mAccept \cup mPass \cup mOff\_grn = grn$

inv4_6 : $mAccept \cap (mPass \cup mOff\_grn) = \varnothing$

inv4_7 : $mPass \cap mOff\_grn = \varnothing$

inv4_8 : $GRN \subseteq mAccept$

Fourth Refinement: the Red Chain (1)

carrier sets: P, L, D

constants: $aut, out, com,$
$exit, org, dst$

variables: $sit, dap, BLR,$
$mCard, mAckn,$
$GRN, mAccept,$
$mOff\_grn,$
$mPass, RED,$
$mOff\_red,$
$mRefuse$

inv4_9 : $RED \subseteq D$

inv4_10 : $mRefuse \subseteq D$

inv4_11 : $mOff\_red \subseteq D$

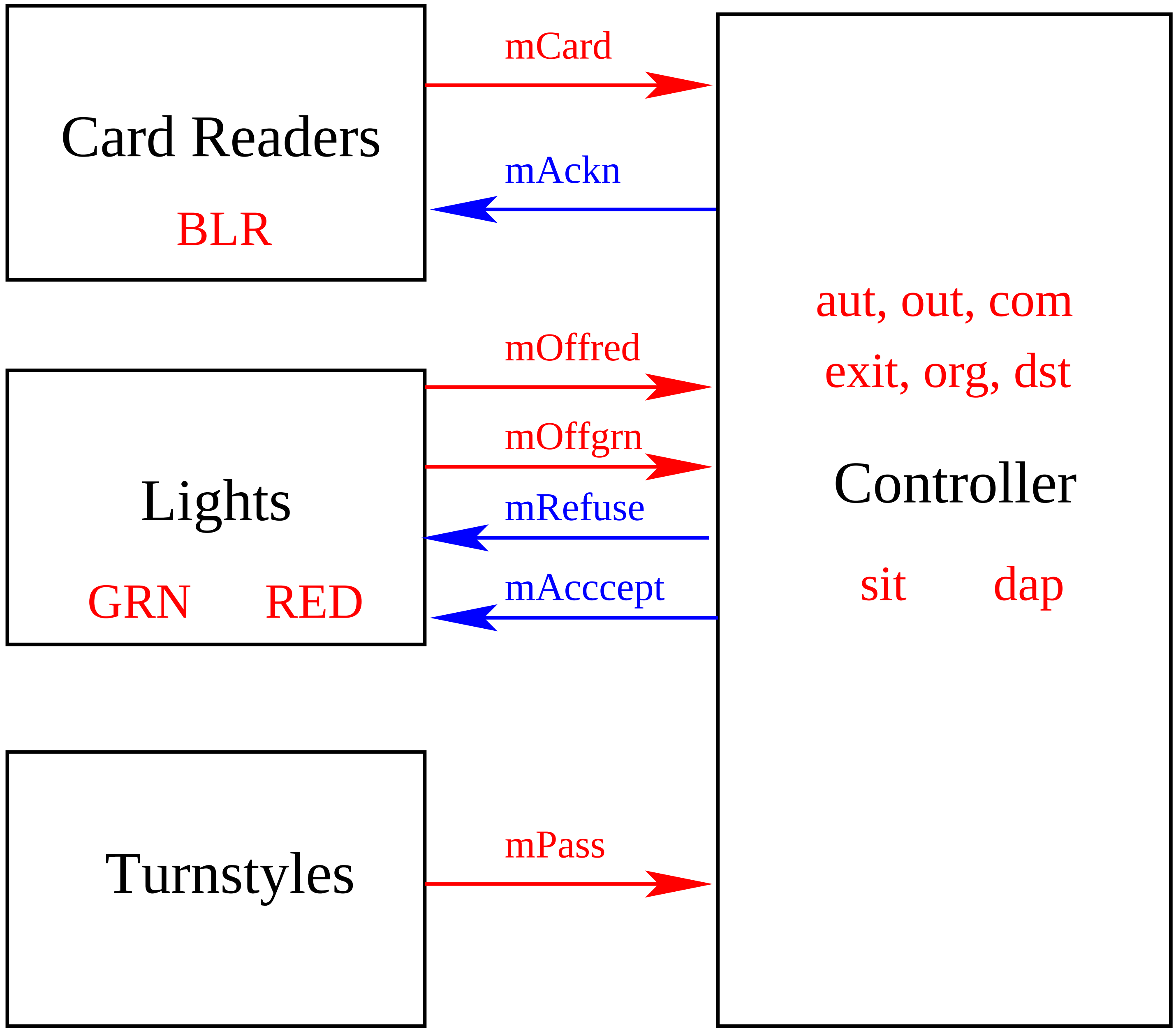
Fourth Refinement: the Red Chain (2)

inv4_12 : $mRefuse \cup mOff\_red = red$

inv4_13 : $mRefuse \cap mOff\_red = \varnothing$

inv4_14 : $RED \subseteq mRefuse$

Fourth Refinement: the final state



accept
any p, d **where**
$d \mapsto p \in mCard$
$sit(p) = org(d)$
$p \mapsto dst(d) \in aut$
$p \notin \mathrm{dom}\,(dap)$
then
$dap(p) := d$
$mCard := mCard \setminus \{d \mapsto p\}$
$mAccept := mAccept \cup \{d\}$
end

ACCEPT
any d **where**
$d \in mAccept$
then
$GRN := GRN \cup \{d\}$
end

PASS

any d **where**

$d \in GRN$

then

$GRN := GRN \setminus \{d\}$

$mPass := mPass \cup \{d\}$

$mAccept := mAccept \setminus \{d\}$

end

pass

any d **where**

$d \in mPass$

then

$sit(dap^{-1}(d)) := dst(d)$

$dap := dap \rhd \{d\}$

$mAckn := mAckn \cup \{d\}$

$mPass := mPass \setminus \{d\}$

end

Events (5)

OFF_GRN

any d **where**

$d \in GRN$

then

$GRN := GRN \setminus \{d\}$

$mOff\_grn := mOff\_grn \cup \{d\}$

$mAccept := mAccept \setminus \{d\}$

end

off_grn

any d **where**

$d \in mOff\_grn$

then

$dap := dap \ntriangleright \{d\}$

$mAckn := mAckn \cup \{d\}$

$mOff\_grn := mOff\_grn \setminus \{d\}$

end

refuse

any p, d **where**

$d, p \in mCard$

$\neg\,(\,sit(p) = org(d)$

$p \mapsto dst(d) \in aut$

$p \notin \mathrm{dom}\,(dap)\,)$

then

$red := red \cup \{d\}$

$mCard := mCard \setminus \{d \mapsto p\}$

$mRefuse := mRefuse \cup \{d\}$

end

REFUSE
  any d **where**
    $d \in mRefuse$
  then
    $RED := RED \cup \{d\}$
  end

OFF_RED

any d **where**

$d \in RED$

then

$RED := RED \setminus \{d\}$

$mOff\_red := mOff\_red \cup \{d\}$

$mRefuse := mRefuse \setminus \{d\}$

end

off_red

any d **where**

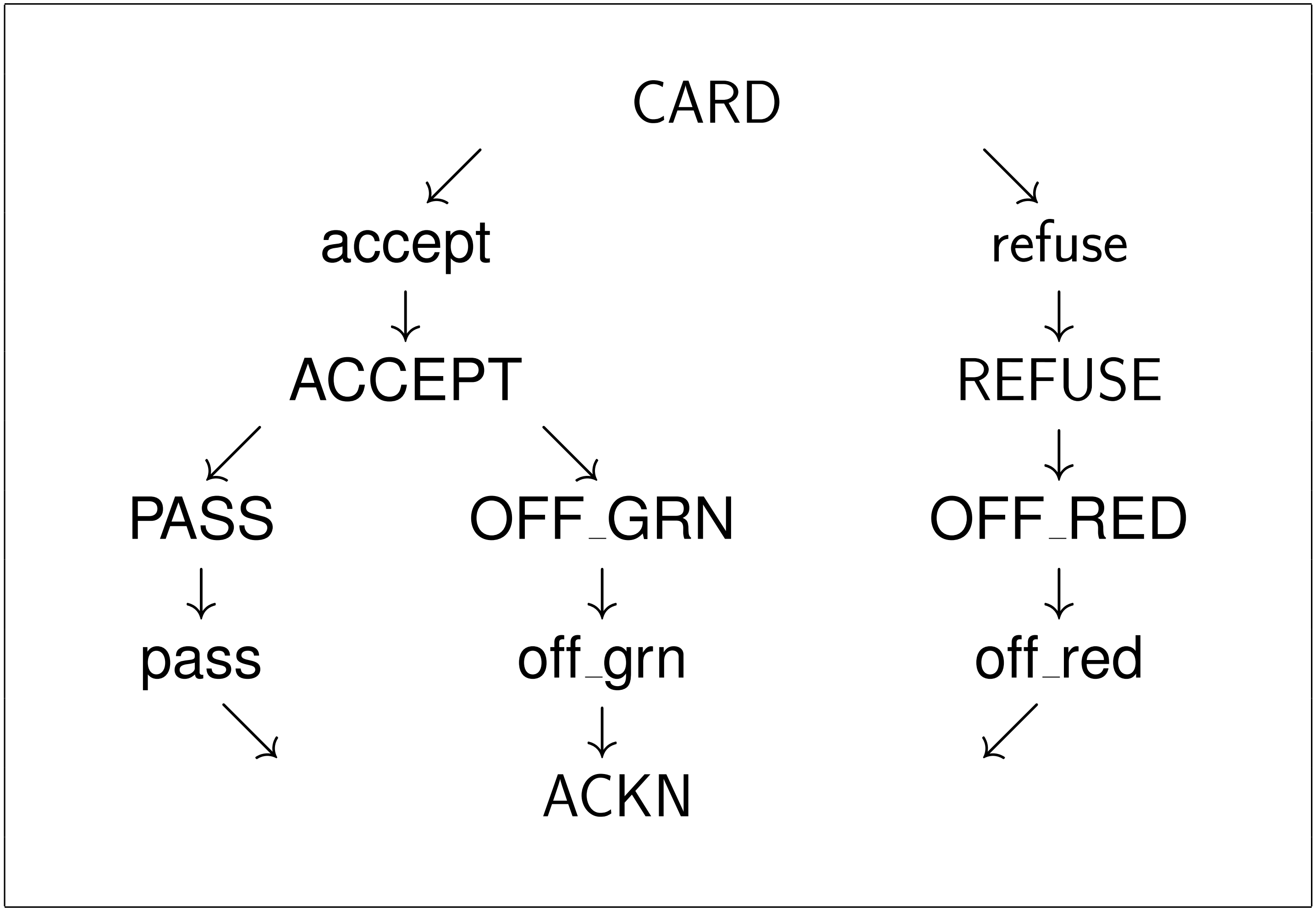
$d \in mOff\_red$

then

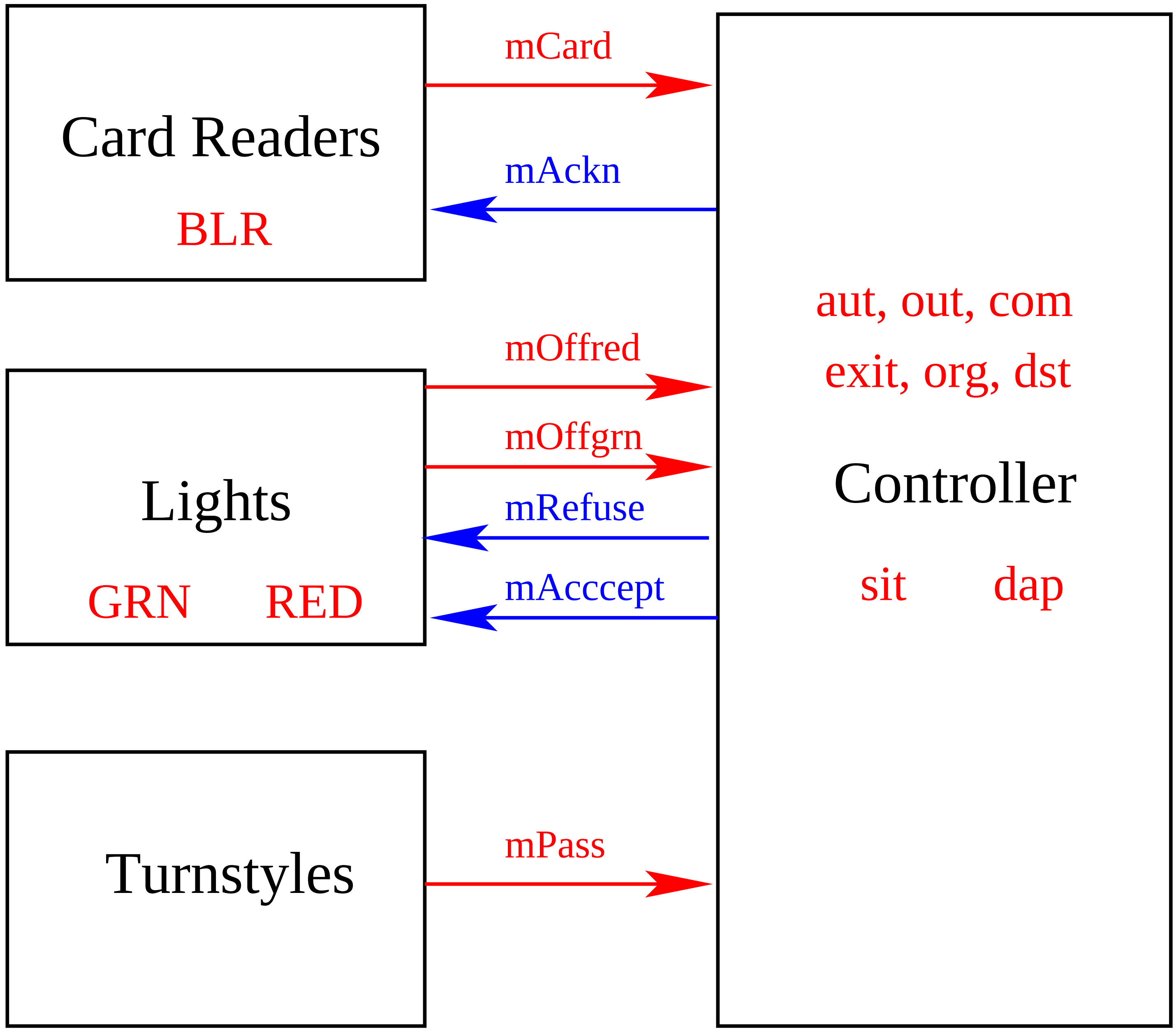
$mAckn := mAckn \cup \{d\}$

$mOff\_red := mOff\_red \setminus \{d\}$

end


CARD
accept
refuse
ACCEPT
REFUSE
PASS
OFF_GRN
OFF_RED
pass
off_grn
off_red
ACKN

Fourth Refinement: the final state



carrier sets: D, P

internal variables: BLR
$GRN,$
RED

external variables: $mCard,$
$mAckn,$
$mAccept,$
$mPass,$
$mRefuse,$
$mOff\_grn$
$mOff\_red$

invH_1 : $BLR \subseteq D$
invH_2 : $RED \subseteq D$
invH_3 : $GRN \subseteq D$
invH_4 : $mCard \subseteq D \leftrightarrow P$
invH_5 : $mAckn \subseteq D$
invH_6 : $mAccept \subseteq D$
invH_7 : $mPass \subseteq D$
invH_8 : $mRefuse \subseteq D$
invH_9 : $mOff\_grn \subseteq D$
invH_10 : $mOff\_red \subseteq D$

carrier sets: D, L, P

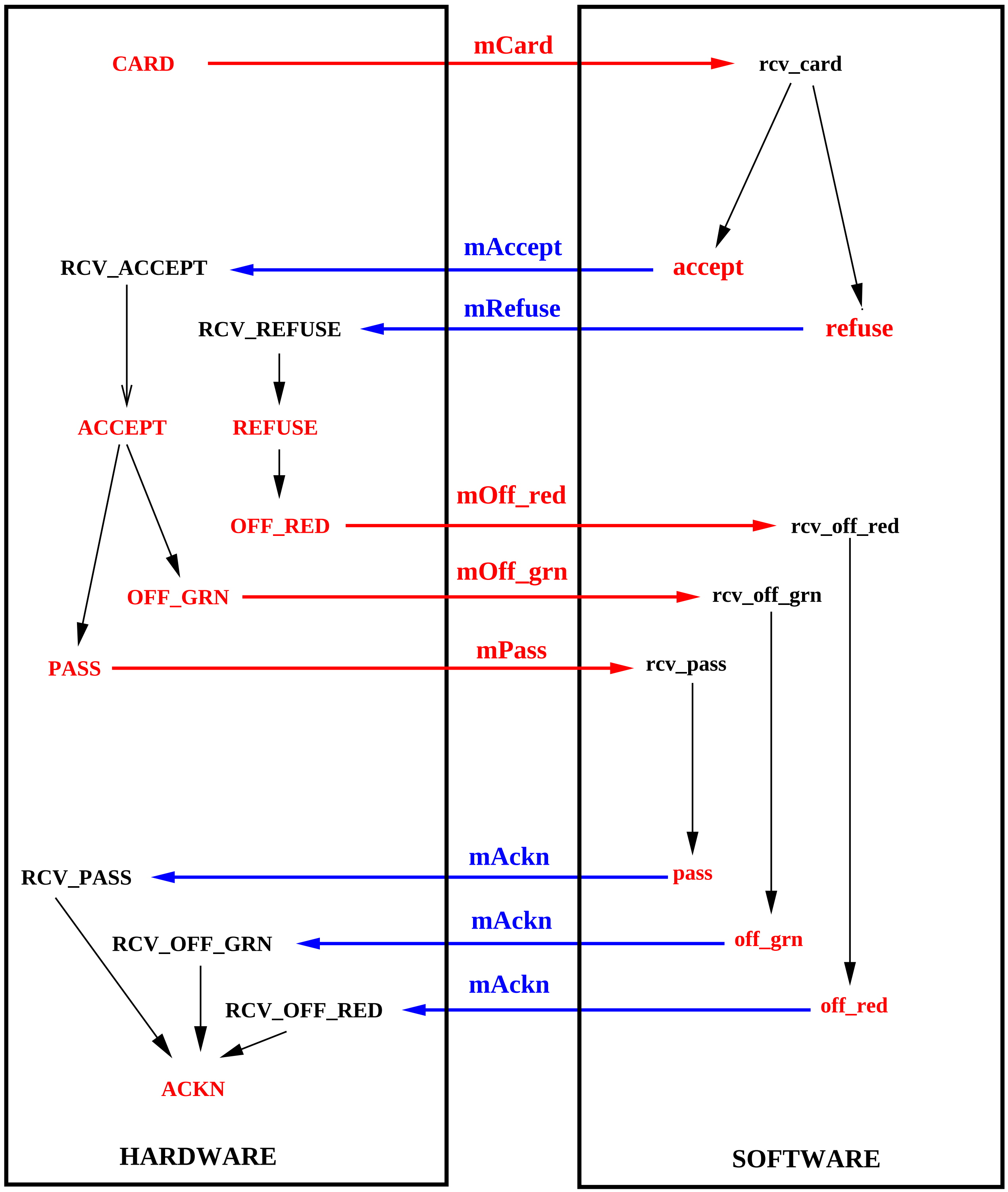
constants: $aut, out,$
org, dst

internal variables: sit
$dap,$

external variables: $mCard,$
$mAckn,$
$mAccept,$
$mPass,$
$mRefuse,$
$mOff\_grn$
$mOff\_red$

axmS_1 : $aut \in P \leftrightarrow L$
axmS_2 : $out \in L$
axmS_3 : $org \in D \rightarrow L$
axmS_4 : $dst \in D \rightarrow L$

invS_1 : $sit \subseteq P \rightarrow L$
invS_2 : $dap \subseteq P \rightarrowtail D$
invS_3 : $mCard \subseteq D \leftrightarrow P$
invS_4 : $mAckn \subseteq D$
invS_5 : $mAccept \subseteq D$
invS_6 : $mPass \subseteq D$
invS_7 : $mRefuse \subseteq D$
invS_8 : $mOff\_grn \subseteq D$
invS_9 : $mOff\_red \subseteq D$



CARD
mCard
rcv_card
mAccept
RCV_ACCEPT
accept
mRefuse
RCV_REFUSE
refuse
ACCEPT
REFUSE
mOff_red
OFF_RED
rcv_off_red
mOff_grn
OFF_GRN
rcv_off_grn
mPass
PASS
rcv_pass
mAckn
RCV_PASS
pass
mAckn
RCV_OFF_GRN
off_grn
mAckn
RCV_OFF_RED
off_red
ACKN
HARDWARE
SOFTWARE

Conclusion (1)

- Functional Requirements: 9

- Safety Requirements: 3

- Equipment Requirements: 5

- Constants: 6

- Variables: 12   (2 Software, 3 Hardware, 7 channels)

- Axioms: 13

Conclusion (2)

- Invariants: 29

- Events: 12   (5 Software, 7 Hardware)

- Refinements: 4

- Design Decisions: 4
    - Possible Card Readers obstruction
    - Automatic physical blocking of Card Readers
    - Automatic physical blocking of Doors
    - Setting up of clocks on Doors

- Proofs: 103 (1easy interactive)