

The Rodin Platform

Jean-Raymond Abrial

2nd Rodin Industrial Day

September 10th 2007

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)
- **Contribution:** "The invention and development of **particle detectors**"

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)
- **Contribution**: "The invention and development of **particle detectors**"
- **Question** asked to Charpak: "Are you a **theoretician**?"

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)
- **Contribution**: "The invention and development of **particle detectors**"
- **Question** asked to Charpak: "Are you a **theoretician**?"
- **Answer**: "No, **I am not**. Although **I know the theory**"

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)
- **Contribution**: "The invention and development of **particle detectors**"
- **Question** asked to Charpak: "Are you a **theoretician**?"
- **Answer**: "No, **I am not**. Although **I know the theory**"
- Answer (cont'd):

- **Georges Charpak** is a French physicist (**Nobel Prize** winner in 1992)
- **Contribution**: "The invention and development of **particle detectors**"
- **Question** asked to Charpak: "Are you a **theoretician**?"
- **Answer**: "No, **I am not**. Although **I know the theory**"
- Answer (cont'd):

"AND MY TOOL IS THE MIRROR OF THE THEORY"

- The theory of **discrete transition systems**

- The theory of **discrete transition systems**
- As **engineers**, a theory is used to build a **methodology**

- The theory of **discrete transition systems**
- As **engineers**, a theory is used to build a **methodology**
- A **methodology** describes the **process allowing to apply a theory**

- The theory of **discrete transition systems**
- As **engineers**, a theory is used to build a **methodology**
- A **methodology** describes the **process allowing to apply a theory**
- The methodology must **precede the tool**

- The theory of **discrete transition systems**
- As **engineers**, a theory is used to build a **methodology**
- A **methodology** describes the **process allowing to apply a theory**
- The methodology must **precede the tool**
- But the **tool will help us** applying the methodology

- That of **developing systems** which will be **correct by construction**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**
 - **concurrent**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**
 - **concurrent**
 - **parallel**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**
 - **concurrent**
 - **parallel**
 - **sequential**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**
 - **concurrent**
 - **parallel**
 - **sequential**
 - **including external environment**

- That of **developing systems** which will be **correct by construction**
- Developing **formal models** in many **different system areas**:
 - **distributed**
 - **concurrent**
 - **parallel**
 - **sequential**
 - **including external environment**
- The tool must be **an aid** for doing this, **not a burden**

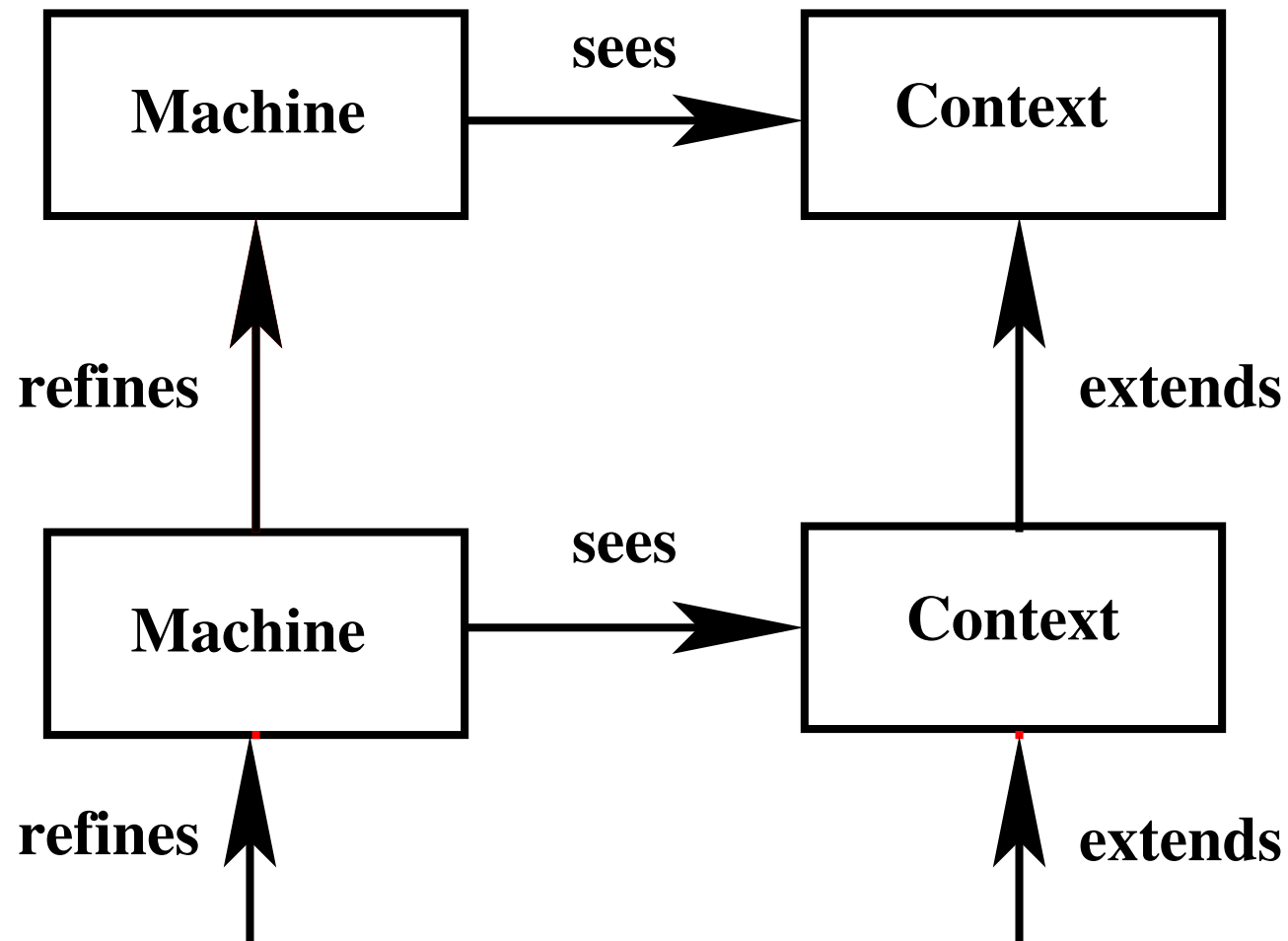
- Models are made of **machines** and **contexts**

Machine

variables
invariants
theorems
events

Context

carrier sets
constants
axioms
theorems



Each section contains **modeling elements**

machine

< machine_identif ier >

refines *

< machine_identif ier >

sees *

< context_identif ier >

...

variables

< variable_identif ier >

...

invariants

< label >: < predicate >

...

theorems *

< label >: < predicate >

...

variant *

< variant >

events

< event >

...

end

context

< context_identif ier >

extends *

< context_identif ier >

...

sets *

< set_identif ier >

...

constants *

< constant_identif ier >

...

axioms *

< label >: < predicate >

...

theorems *

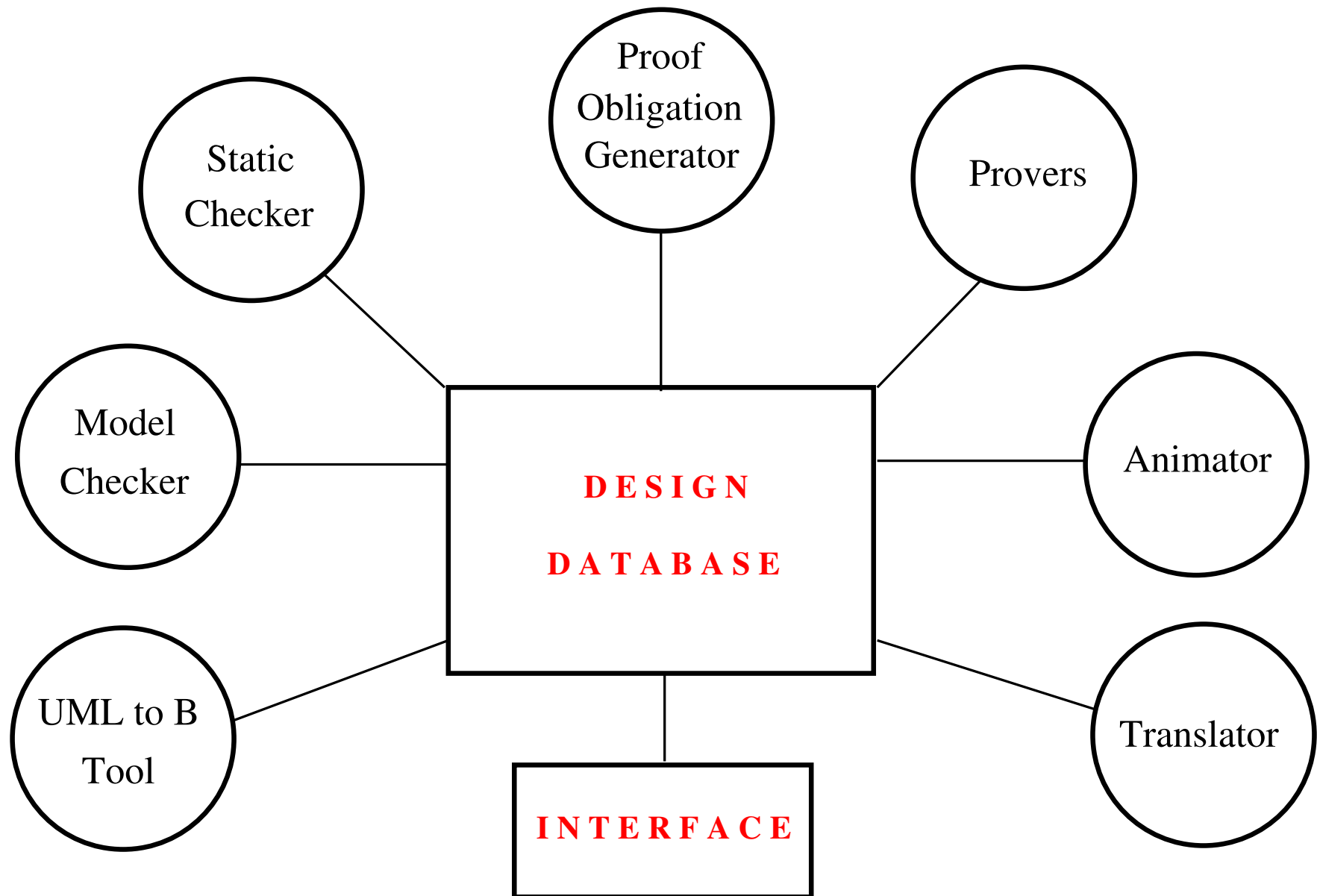
< label >: < predicate >

...

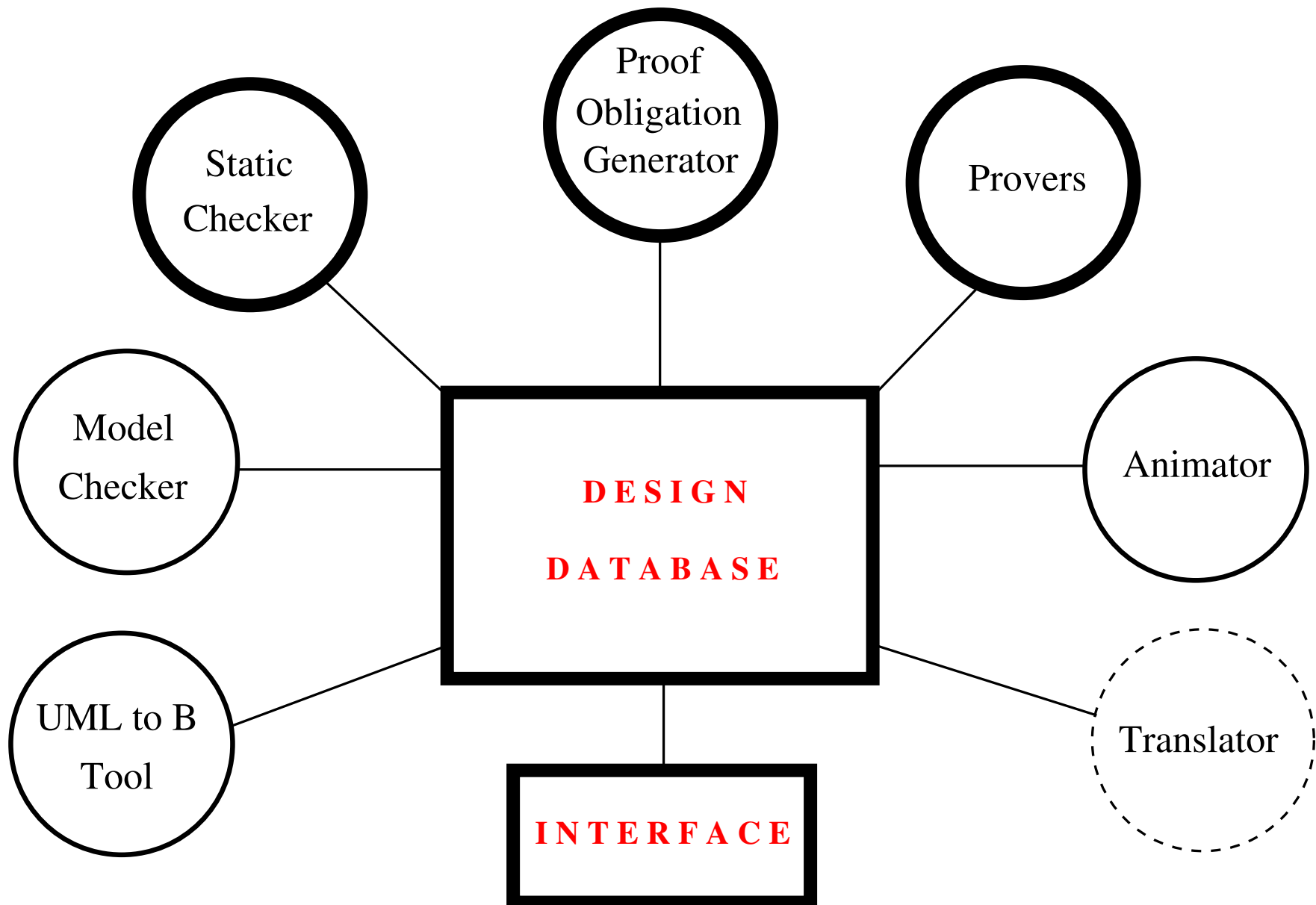
end


```
< event_identifier > ≐  
status  
  {ordinary, convergent, anticipated}  
refines ★  
  < event_identifier >  
  ...  
any ★  
  < parameter_identifier >  
  ...  
where ★  
  < label >: < predicate >  
  ...  
with ★  
  < label >: < witness >  
  ...  
then ★  
  < label >: < action >  
  ...  
end
```


DESIGN AND VERIFICATION PLUG-INS



DESIGN AND VERIFICATION PLUG-INS



- The team

- The **team**
- Main **characteristics** of the Rodin Platform

- The **team**
- Main **characteristics** of the Rodin Platform
- A **Grand Tour** of the Rodin Platform by means of an **example**

- The **team**
- Main **characteristics** of the Rodin Platform
- A **Grand Tour** of the Rodin Platform by means of an **example**
- **Various demos** performed **during example presentation**

- Laurent Voisin: Main Architect

- **Laurent Voisin**: Main Architect
- **Stefan Hallerstede**: Static Checker, Builder, PO Generator

- **Laurent Voisin**: Main Architect
- **Stefan Hallerstede**: Static Checker, Builder, PO Generator
- **Thai Son Hoang**: User Interface, Interactive Prover Rules

- **Laurent Voisin**: Main Architect
- **Stefan Hallerstede**: Static Checker, Builder, PO Generator
- **Thai Son Hoang**: User Interface, Interactive Prover Rules
- **Farhad Mehta**: Proof Manager, Well-definedness

- **Laurent Voisin**: Main Architect
- **Stefan Hallerstede**: Static Checker, Builder, PO Generator
- **Thai Son Hoang**: User Interface, Interactive Prover Rules
- **Farhad Mehta**: Proof Manager, Well-definedness
- **Francois Terrier**: New Prover

- **Laurent Voisin**: Main Architect
- **Stefan Hallerstede**: Static Checker, Builder, PO Generator
- **Thai Son Hoang**: User Interface, Interactive Prover Rules
- **Farhad Mehta**: Proof Manager, Well-definedness
- **Francois Terrier**: New Prover
- External participants: **Dominique Cansell** and **Christophe Metayer**

- Build on top of **Eclipse**

- Build on top of **Eclipse**
- Running on **Windows, Mac-OS, and Linux**

- Build on top of **Eclipse**
- Running on **Windows**, **Mac-OS**, and **Linux**
- Development: **3 years**

- Build on top of **Eclipse**
- Running on **Windows, Mac-OS, and Linux**
- Development: **3 years**
- **120,000 lines** of Java (after removing **blank lines** and **comment lines**)

- Build on top of **Eclipse**
- Running on **Windows, Mac-OS, and Linux**
- Development: **3 years**
- **120,000 lines** of Java (after removing **blank lines** and **comment lines**)
- **Open source**

- Build on top of **Eclipse**
- Running on **Windows, Mac-OS, and Linux**
- Development: **3 years**
- **120,000 lines** of Java (after removing **blank lines** and **comment lines**)
- **Open source**
- **Extensible** (plug-ins)

- A hotel key card system

- A hotel key card system
- It is not a software system

- A hotel key card system
- It is not a software system
- The model will allow us to analyze this system

- A hotel key card system
- It is not a software system
- The model will allow us to analyze this system
- References:

- A **hotel key card system**
- It is **not a software** system
- The **model** will allow us **to analyze this system**
- References:
 - Daniel Jackson, *Software Abstractions*. MIT Press, 2006

- A **hotel key card system**
- It is **not a software** system
- The **model** will allow us **to analyze this system**
- References:
 - Daniel Jackson, *Software Abstractions*. MIT Press, 2006
 - Tobias Nipkow, *Verifying a Hotel Key Card System*. ICTAC, 2006

- Upon checking-in, you are given a magnetic card.

- Upon **checking-in**, you are given a **magnetic card**.
- Between your **check-in** and your **check-out**, the system must:

- Upon **checking-in**, you are given a **magnetic card**.
- Between your **check-in** and your **check-out**, the system must:
 - (1) allow you to **enter the room** you booked,

- Upon **checking-in**, you are given a **magnetic card**.
- Between your **check-in** and your **check-out**, the system must:
 - (1) allow you to **enter the room** you booked,
 - (2) guarantee that **no one else can do so**.

- Upon **checking-in**, you are given a **magnetic card**.
- Between your **check-in** and your **check-out**, the system must:
 - (1) allow you to **enter the room** you booked,
 - (2) guarantee that **no one else can do so**.
- At **check-out**, you are **not obliged to return the magnetic card**

- Upon **checking-in**, you are given a **magnetic card**.
- Between your **check-in** and your **check-out**, the system must:
 - (1) allow you to **enter the room** you booked,
 - (2) guarantee that **no one else can do so**.
- At **check-out**, you are **not obliged** to return the magnetic card
- In a classical hotel **metallic key** system, (2) is not guaranteed (a **previous user** may have "borrowed" the metallic key)

- Each check-in starts a **new booking**

- Each check-in starts a **new booking**
- Each booking is associated with a **key-card**

- Each check-in starts a **new booking**
- Each booking is associated with a **key-card**
- Each key-card carries **two electronic keys**:

- Each check-in starts a **new booking**
- Each booking is associated with a **key-card**
- Each key-card carries **two electronic keys**:
 - (1) Its own **new key**: guaranteed to be fresh

- Each check-in starts a **new booking**
- Each booking is associated with a **key-card**
- Each key-card carries **two electronic keys**:
 - (1) Its own **new key**: guaranteed to be fresh
 - (2) the "previous" booking key: the "new" key from the **previous booking card**

- Each door has an **independent electronic lock**

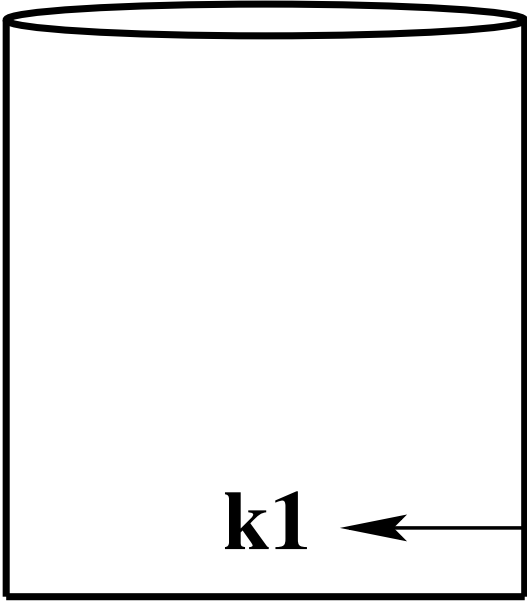
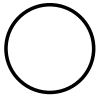
- Each door has an **independent electronic lock**
- The **lock** at the door **holds one electronic key**

- Each door has an **independent electronic lock**
- The **lock** at the door **holds one electronic key**
- You insert your card in the lock

- Each door has an **independent electronic lock**
- The **lock** at the door **holds one electronic key**
- You insert your card in the lock
- The lock **reads your card**

- Each door has an **independent electronic lock**
- The **lock** at the door **holds one electronic key**
- You insert your card in the lock
- The lock **reads your card**
- The door opens if the door key **is one of the two keys of your card**

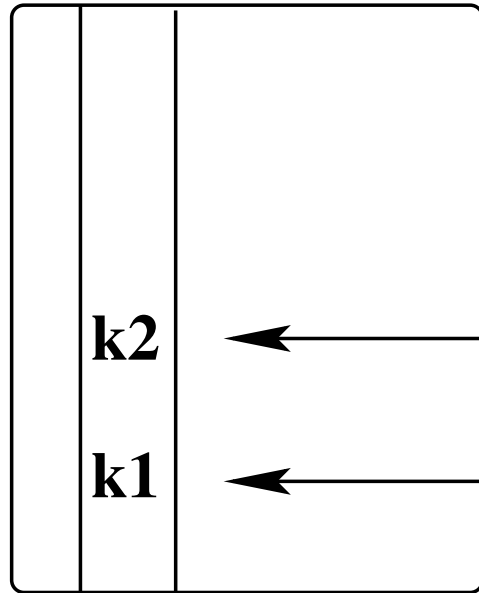
- Each door has an independent electronic lock
- The lock at the door holds one electronic key
- You insert your card in the lock
- The lock reads your card
- The door opens if the door key is one of the two keys of your card
- The door key is replaced by your own key (as read on the card)



k1

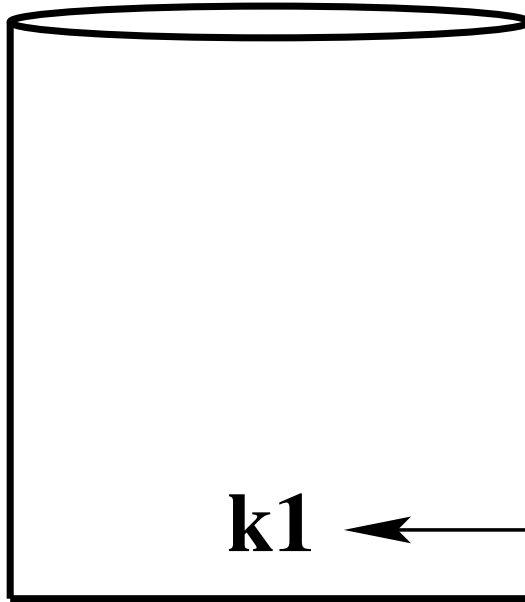
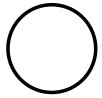


door key

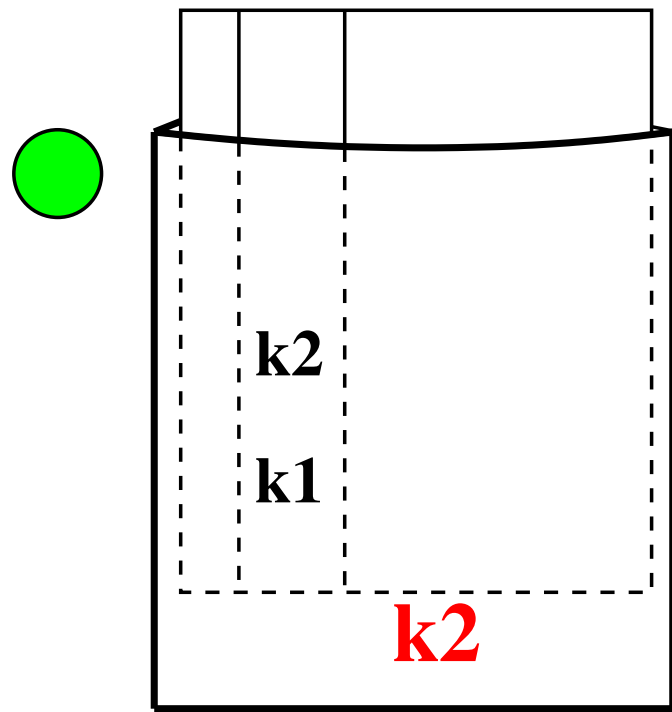
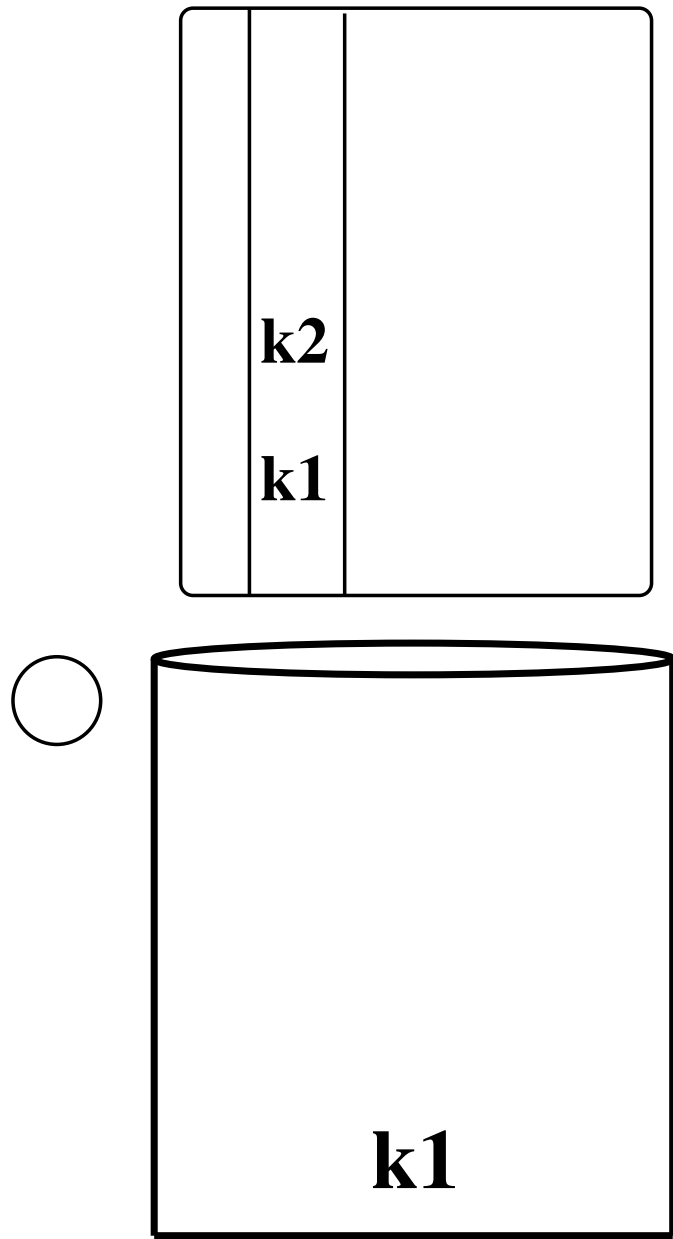


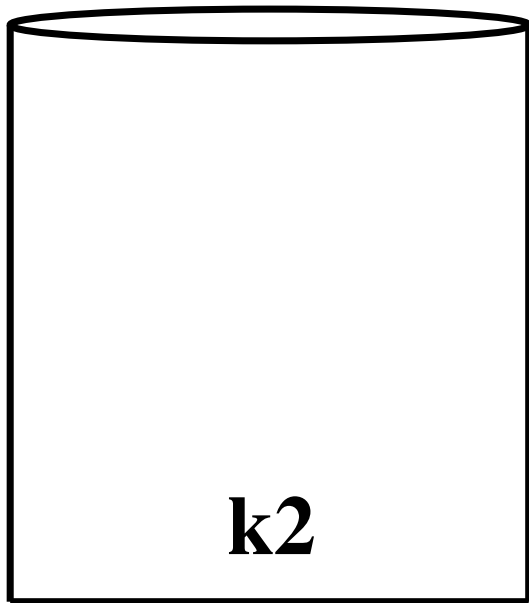
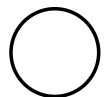
new key

previous key

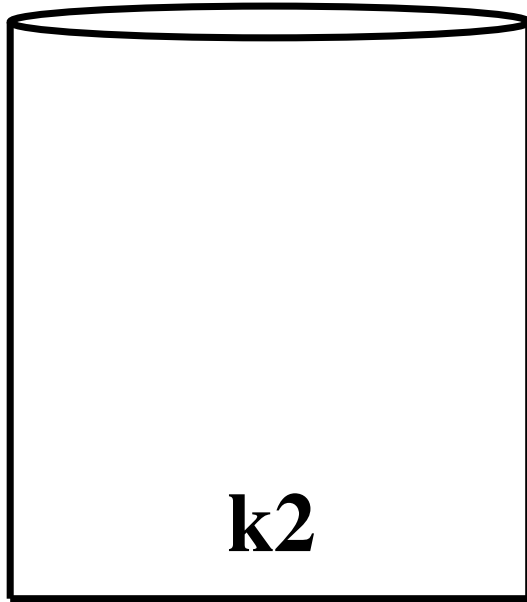
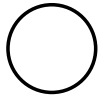
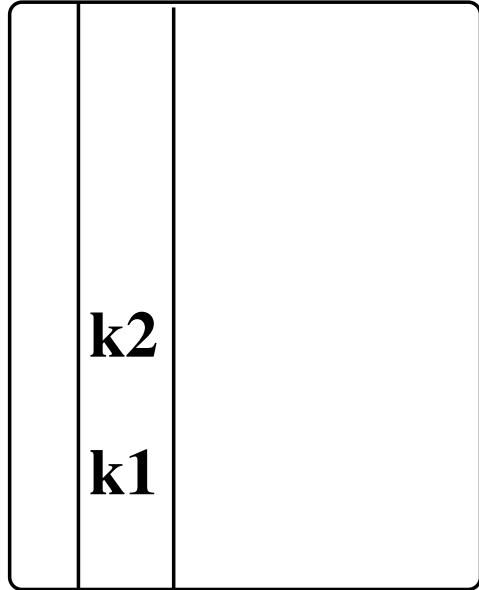


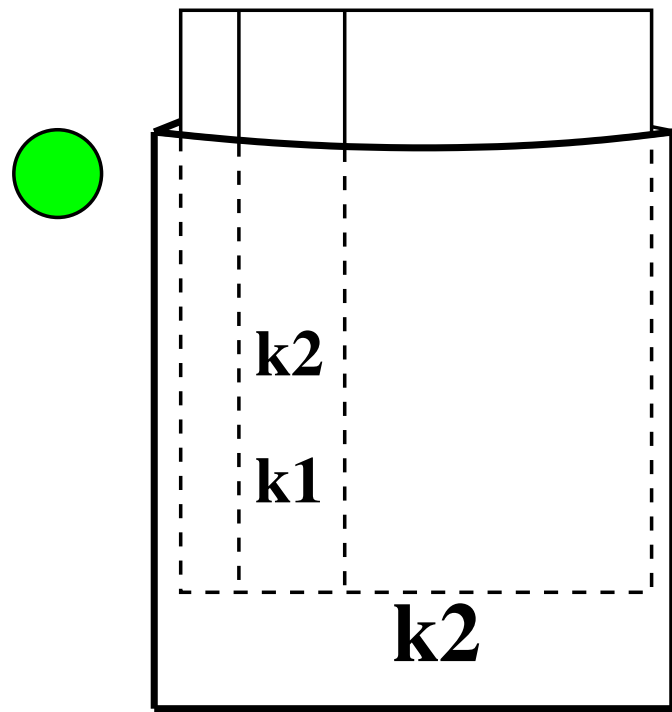
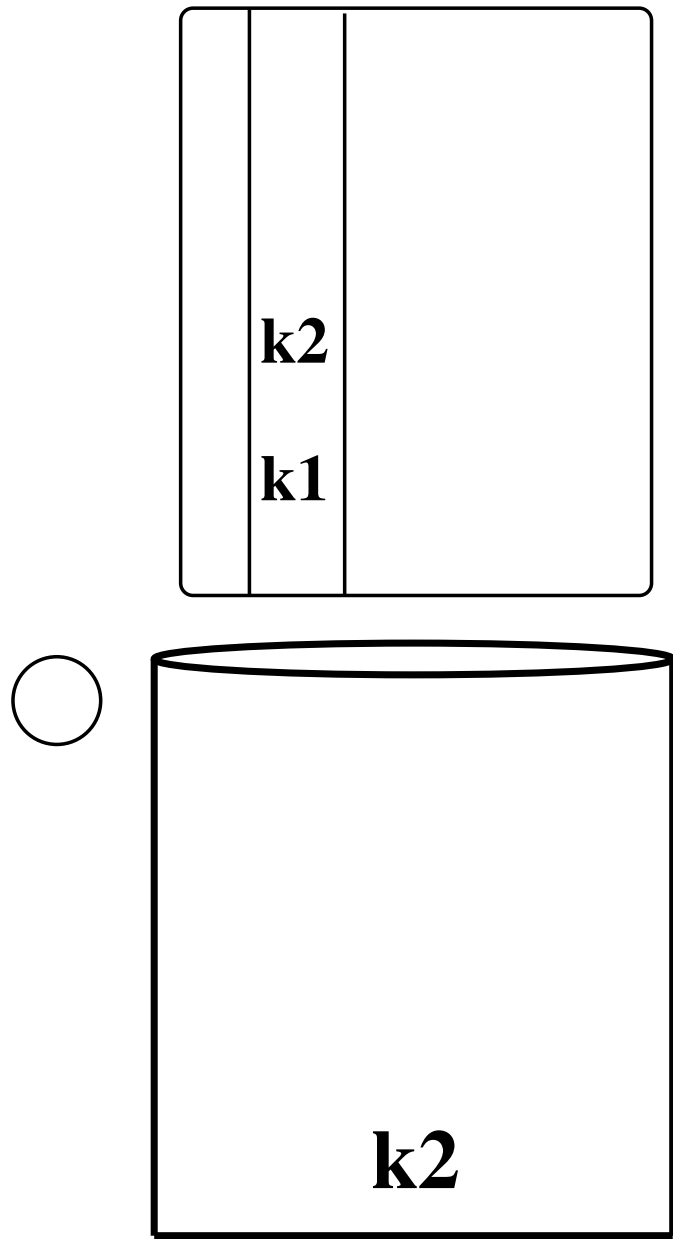
door key

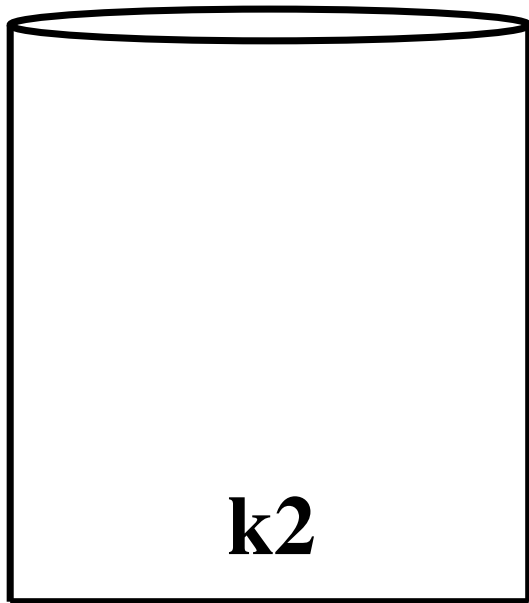
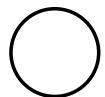




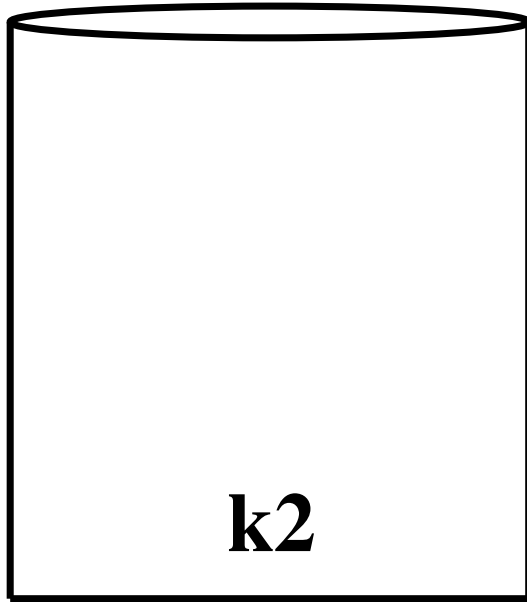
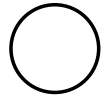
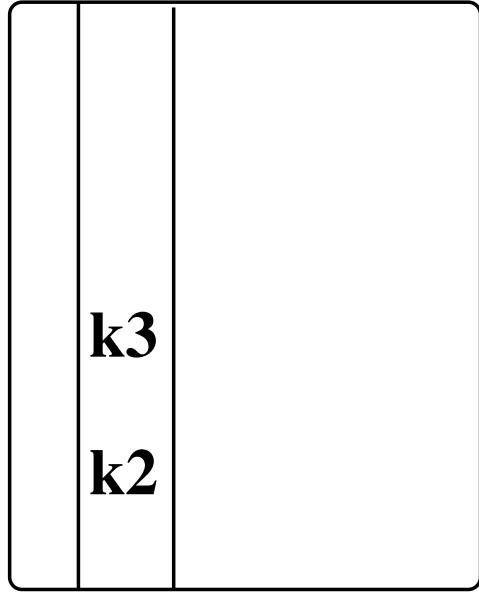
k2

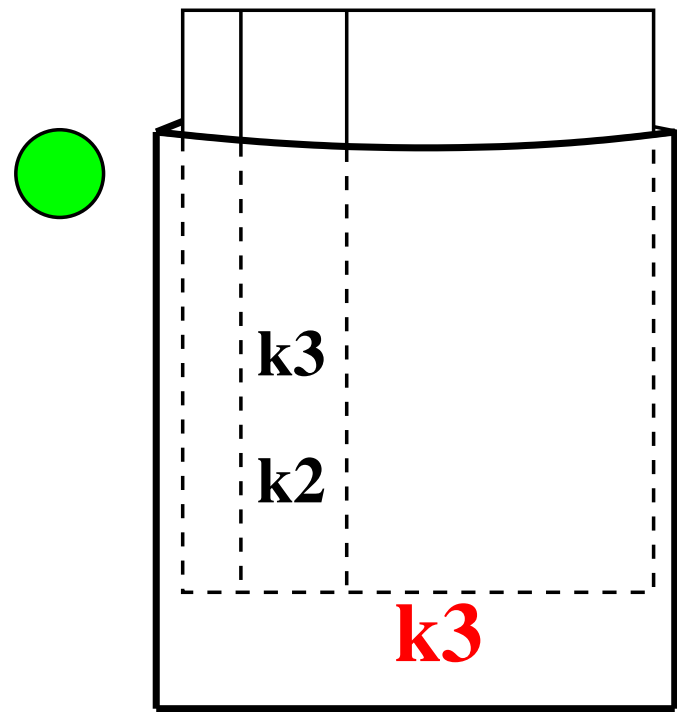
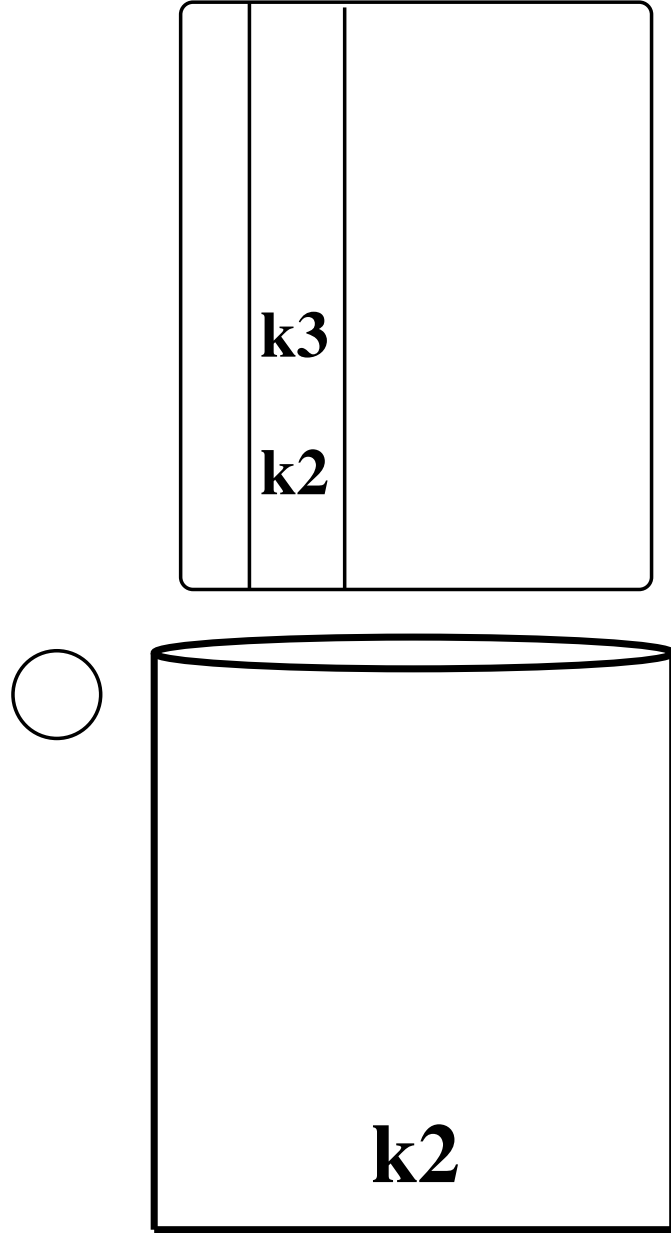


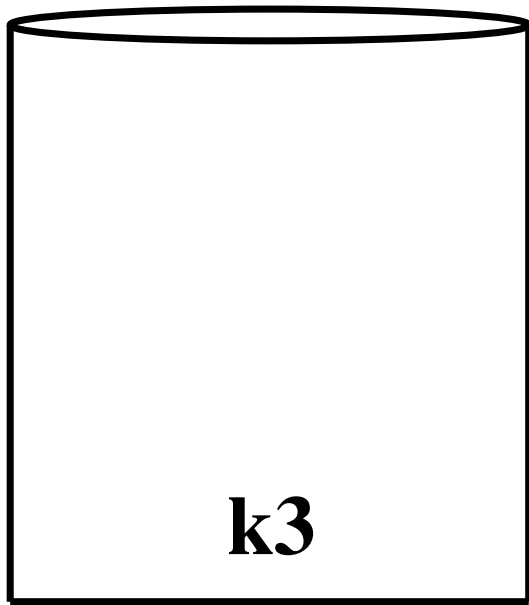
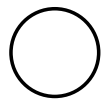




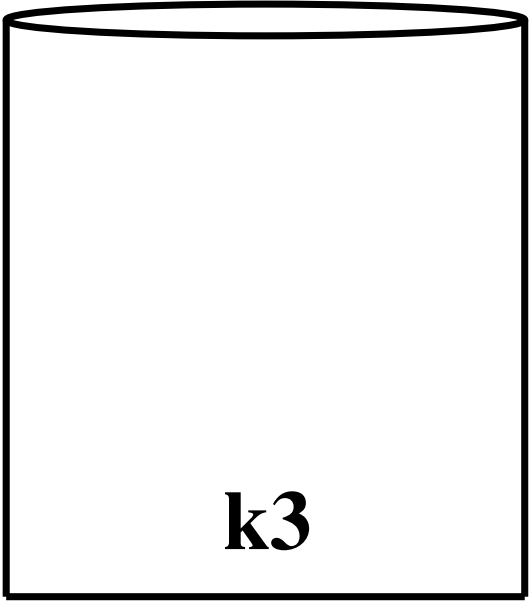
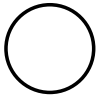
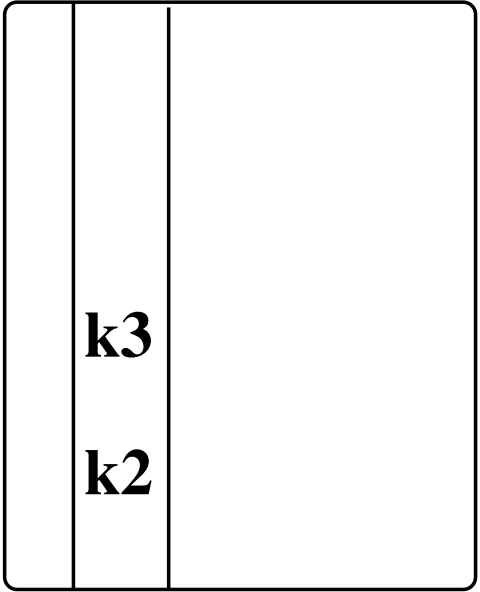
k2

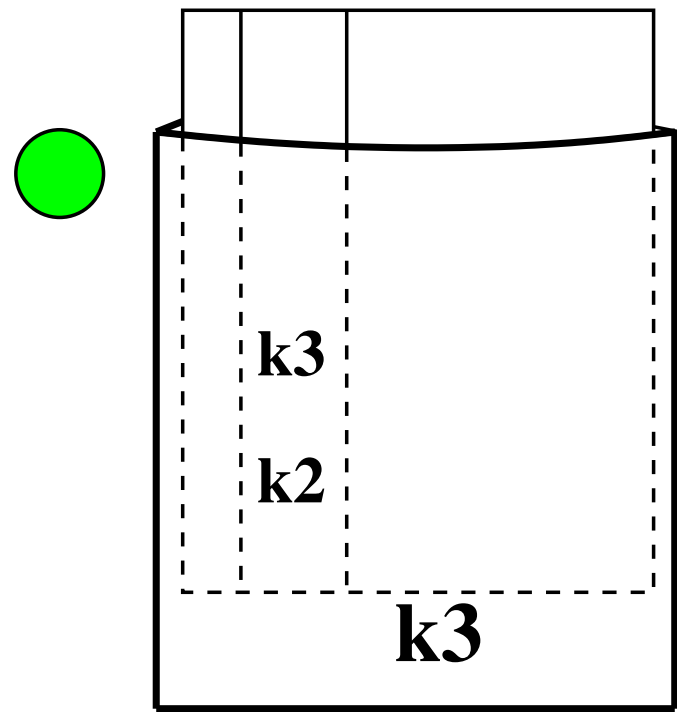
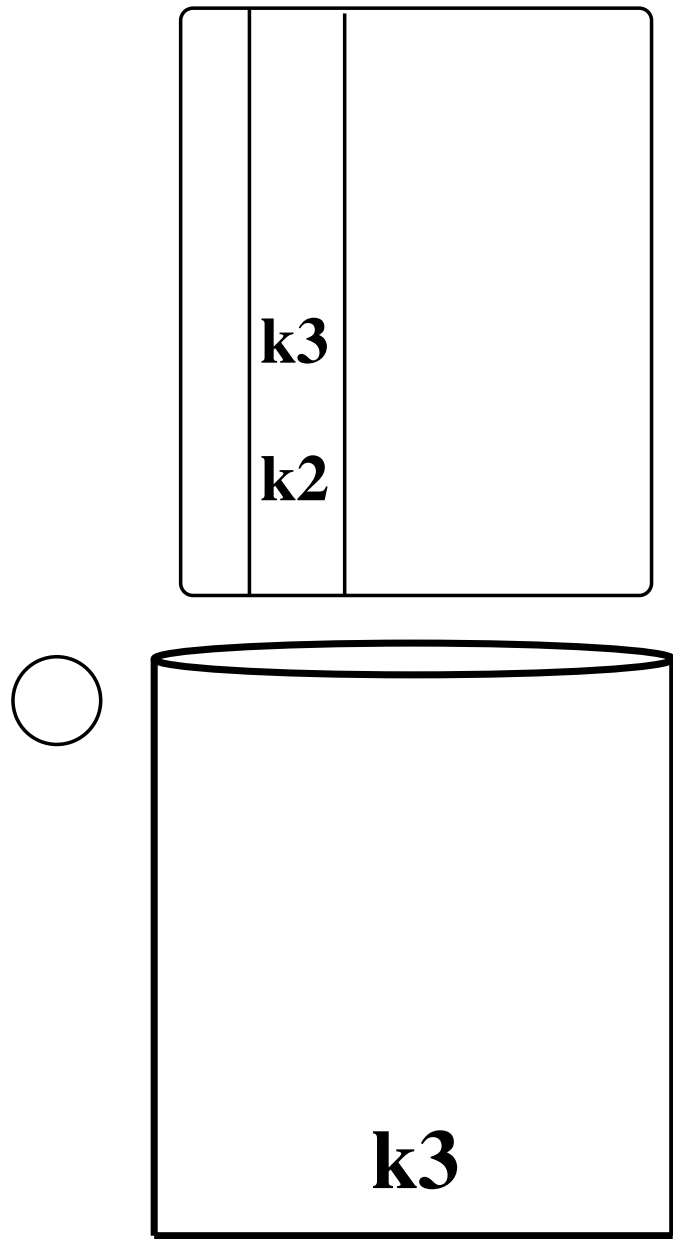


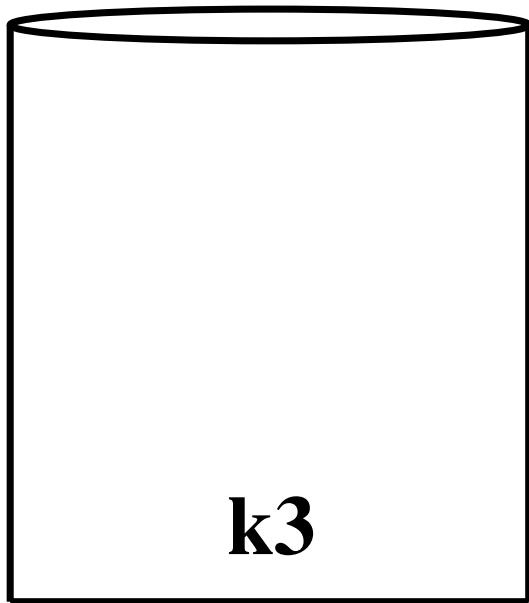
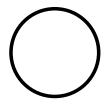




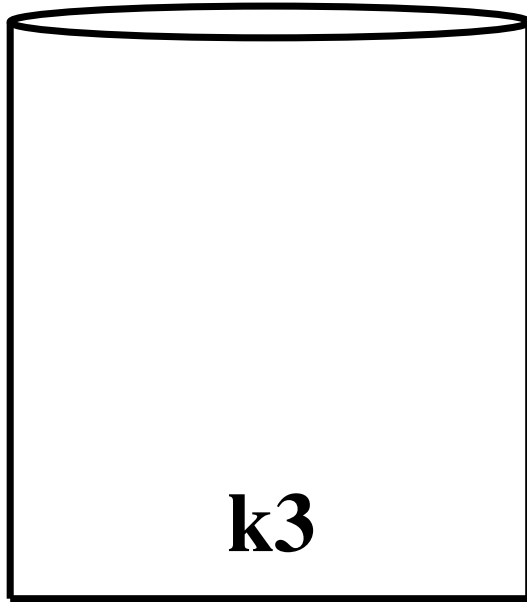
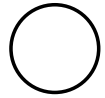
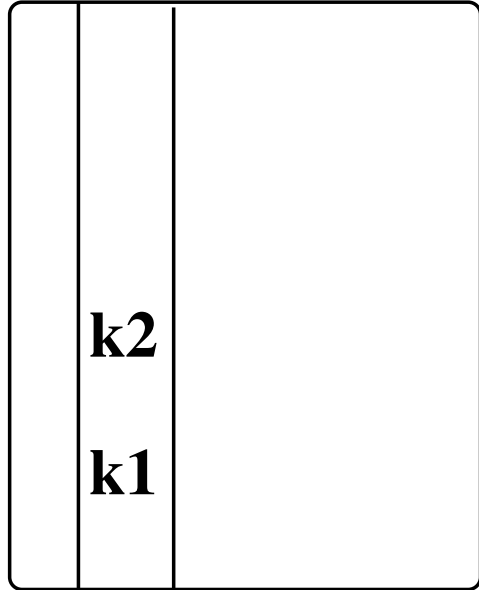
k3

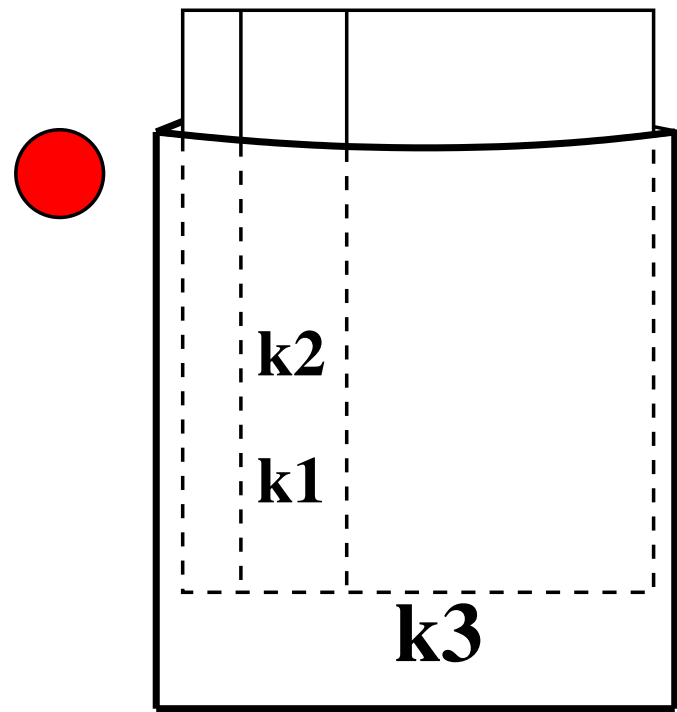
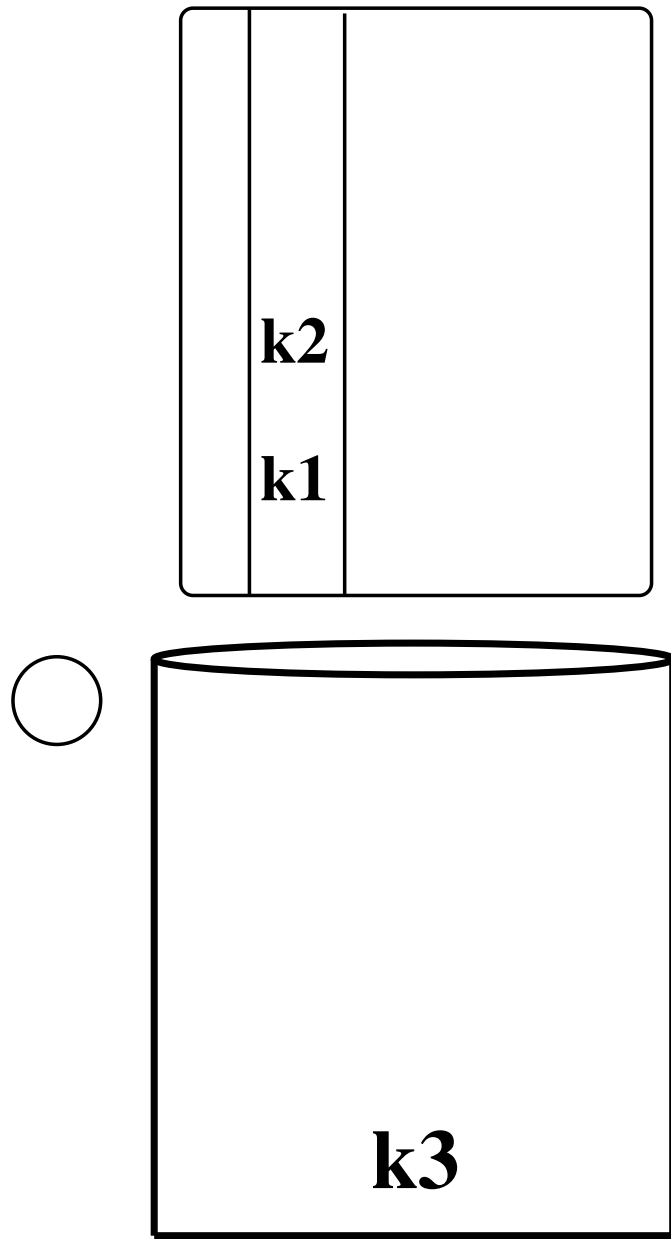











k3





	open	closed
PAST bookings		
PRESENT booking		
FUTUR bookings		

- **check_in**: performed by **hotel management** and **guest**

- **check_in**: performed by **hotel management** and **guest**
- **check_out**: performed by **hotel management** (and guest)

- **check_in**: performed by **hotel management** and **guest**
- **check_out**: performed by **hotel management** (and guest)
- **use**: performed by **guest** (this is the **first entrance** in the room)

Booking b_1 is a dummy

b_1



Booking *b2* checks in

b1



b2



Booking b_2 uses the room.

b1



b2



Booking b_1 becomes a past booking

Booking *b2* checks out

b1



b2



Booking *b3* checks in

b1



b2



b3



Booking b_3 uses the room.

b1



b2



b3



Booking b_2 becomes a past booking

Booking b_4 checks in

b1



b2



b3



b4



NOT ALLOWED

At most one open booking

Booking *b2* uses the room.

b1



b2



b3



Booking *b3* becomes a past booking

NOT ALLOWED

A past booking cannot "use" the room for the first time

Booking *b3* checks out

b1



b2



b3



Booking *b*₄ checks in

b1



b2



b3



b4



Booking *b4* checks out **without using the room**

b1



b2



b3



b4



Booking *b*5 checks in

b1



b2



b3



b4



b5



Booking *b*5 checks out **without using the room**

b1



b2



b3



b4



b5



Booking *b6* checks in

b1



b2



b3



b4



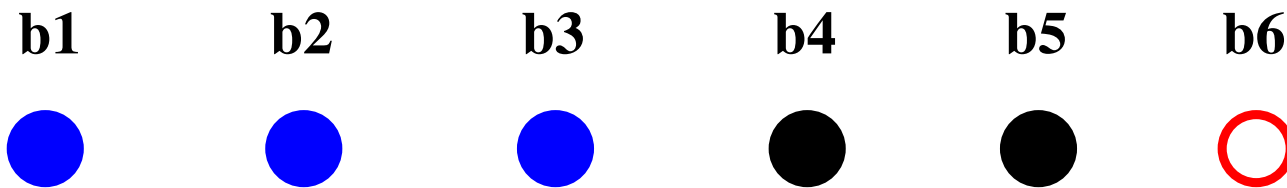
b5



b6

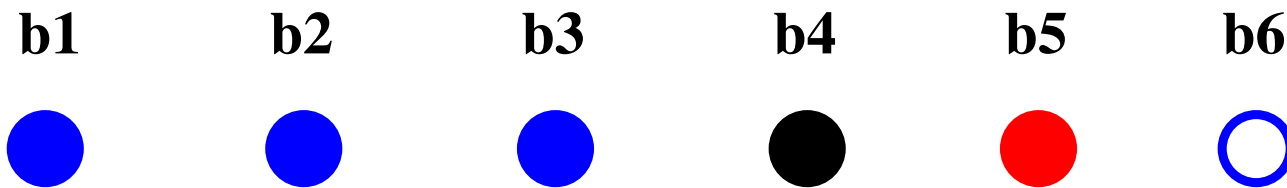


Booking *b6* uses the room.



Booking *b3* becomes a past booking

- Booking b_5 uses the room.

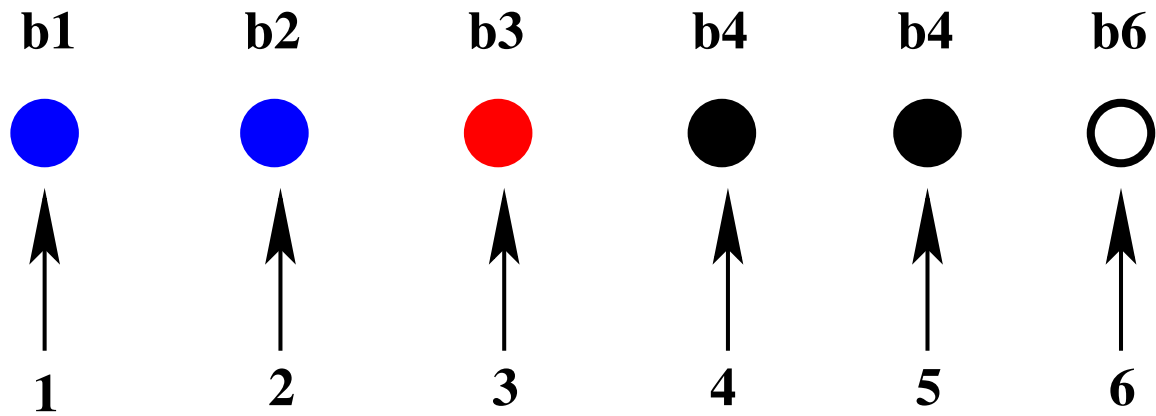
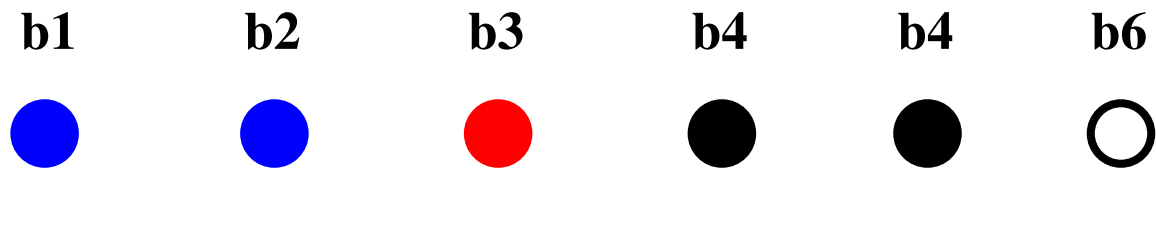


Open booking b_6 becomes a past booking.

NOT ALLOWED.

Before using a room, one must be sure that its user has checked out.

This point will be solved "naturally" in the next refinement



b1



1

b1



1

b2



2

b1



1

b2



2

b1



1

b2



2

b1



1

b2



2

b3



3

b1



1

b2



2

b3



3

b1



1

b2



2

b3



3

b1



1

b2



2

b3



3

b4



4

b1



1

b2



2

b3



3

b4



4

b1



1

b2



2

b3



3

b4



4

b5



5

b1



1

b2



2

b3



3

b4



4

b5



5

b1



1

b2



2

b3



3

b4



4

b5



5

b6



6

b1



1

b2



2

b3



3

b4



4

b5



5

b6



6

b1



1

b2



2

b3



3

b4



4

b5



5

b6



6

b1



1

b2



2

b3



3

b4



4

b5

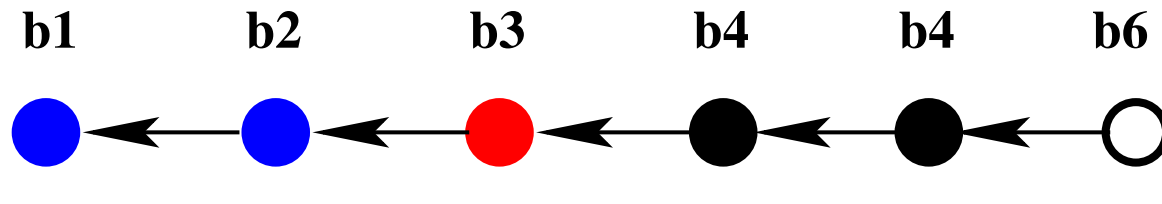
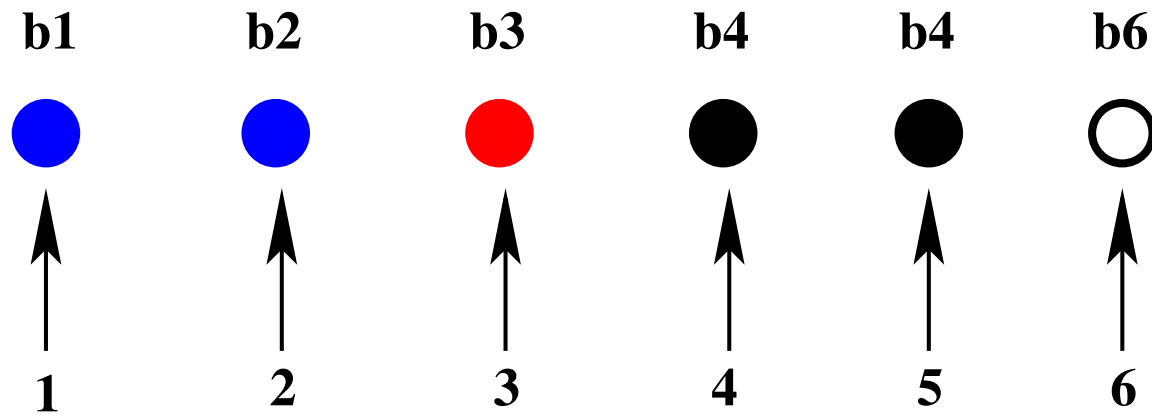


5

b6

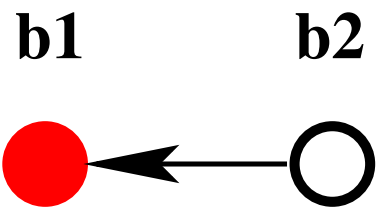


6



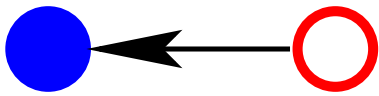
b1

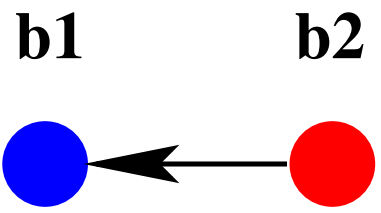


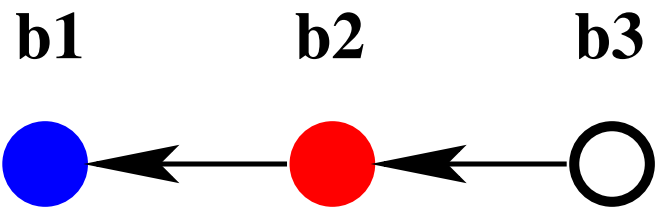


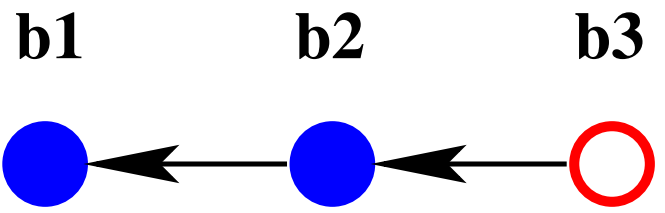
b1

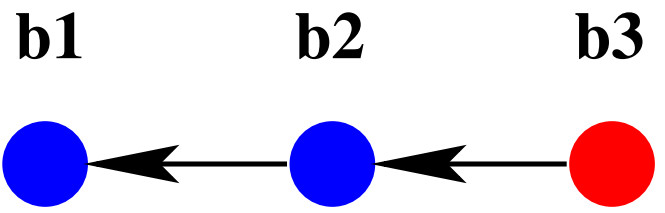
b2

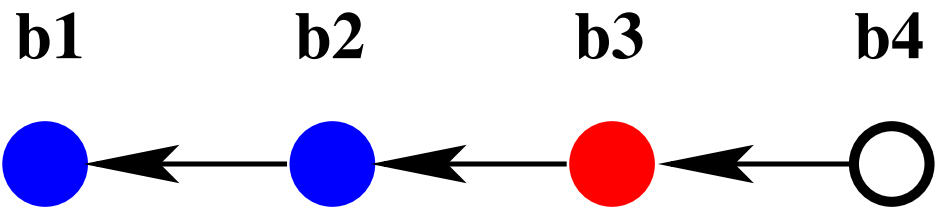


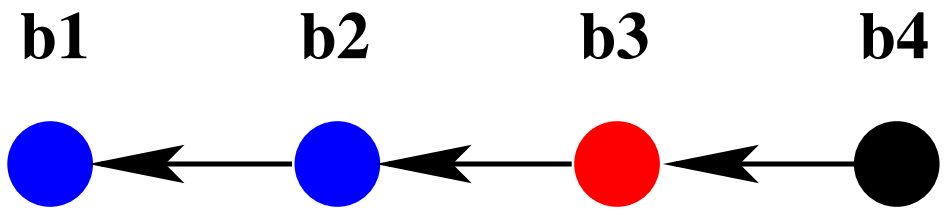


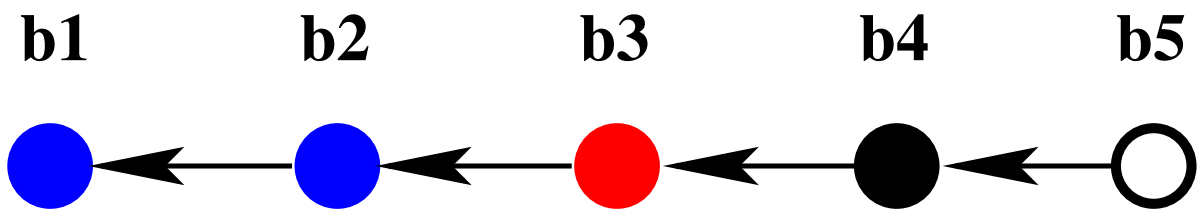


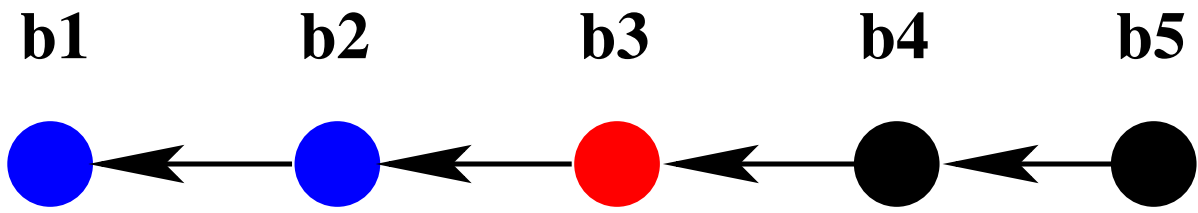


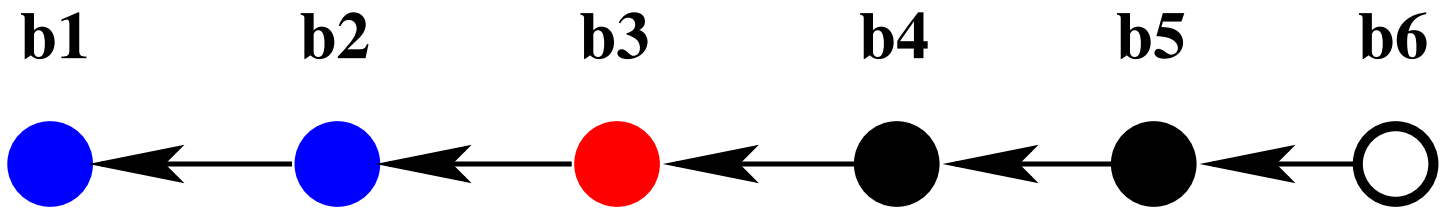


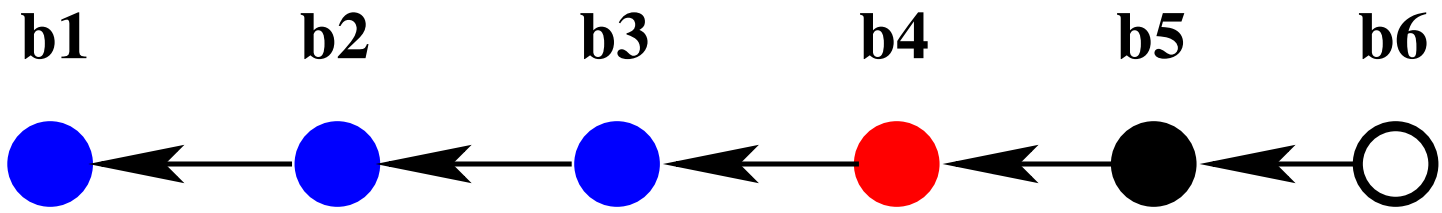


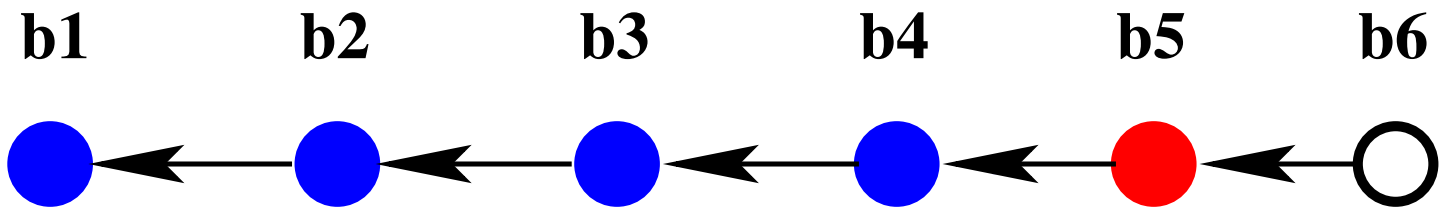


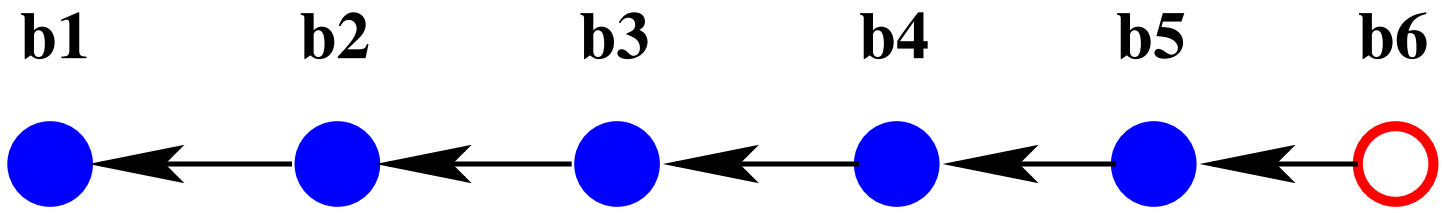


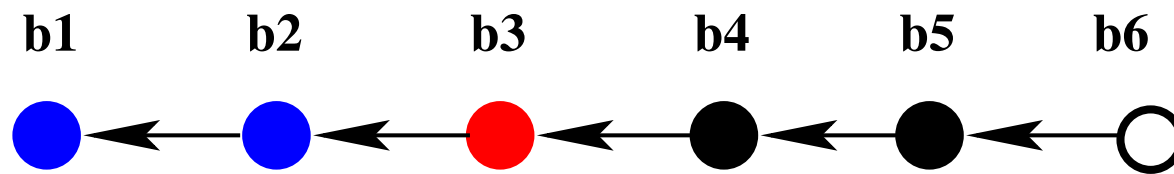




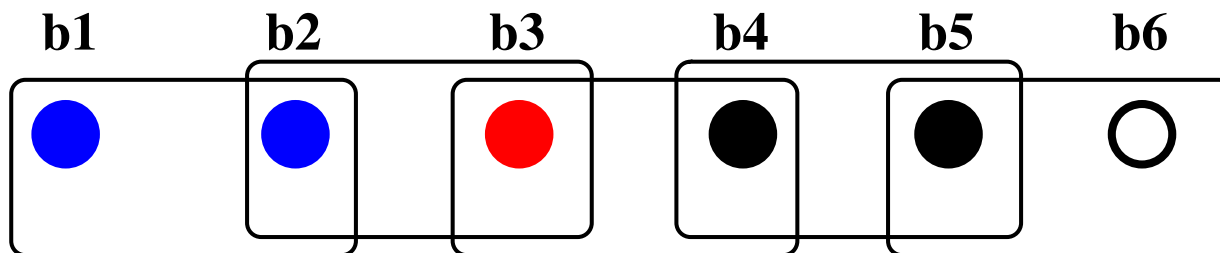








**Abstract
(chain)**



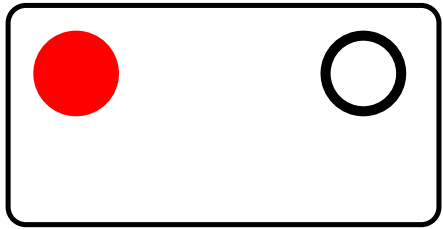
**Concrete
(card)**

b1



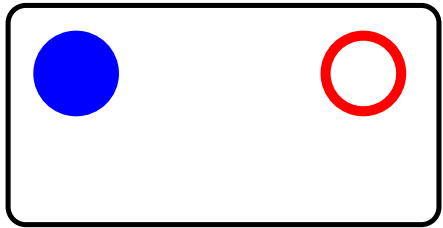
b1

b2



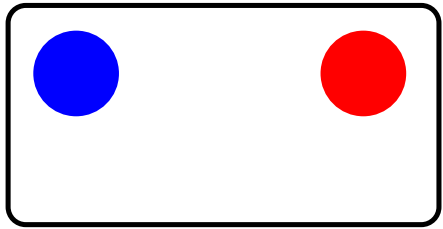
b1

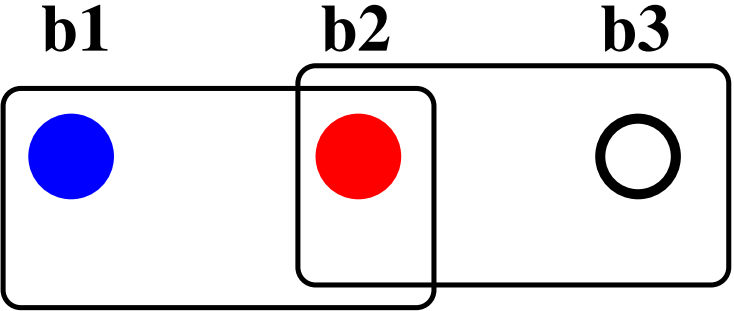
b2

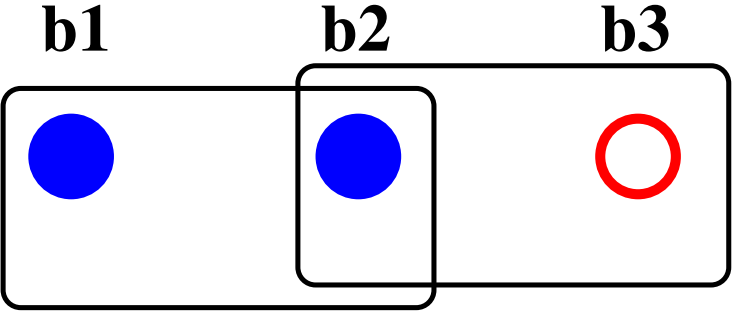


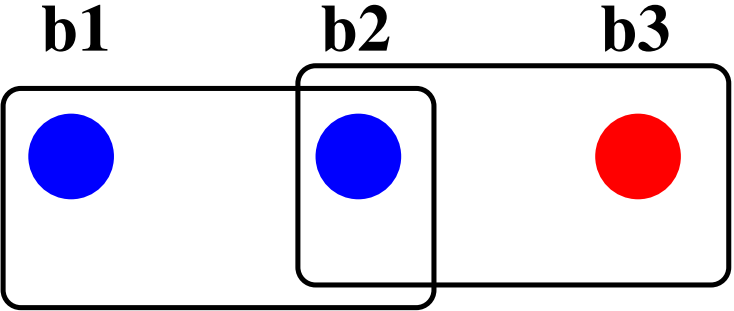
b1

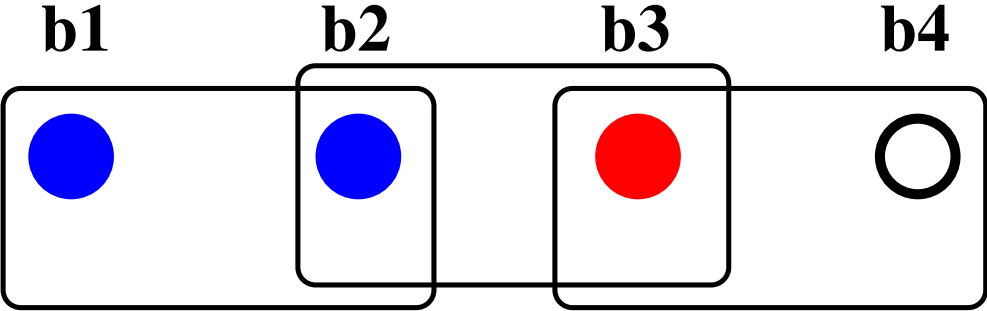
b2

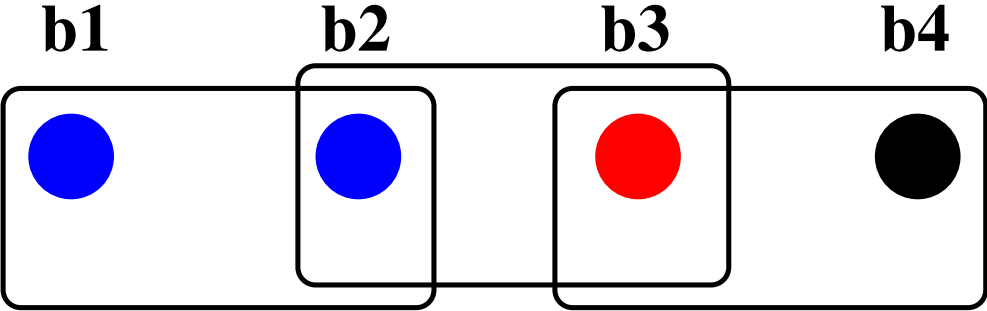


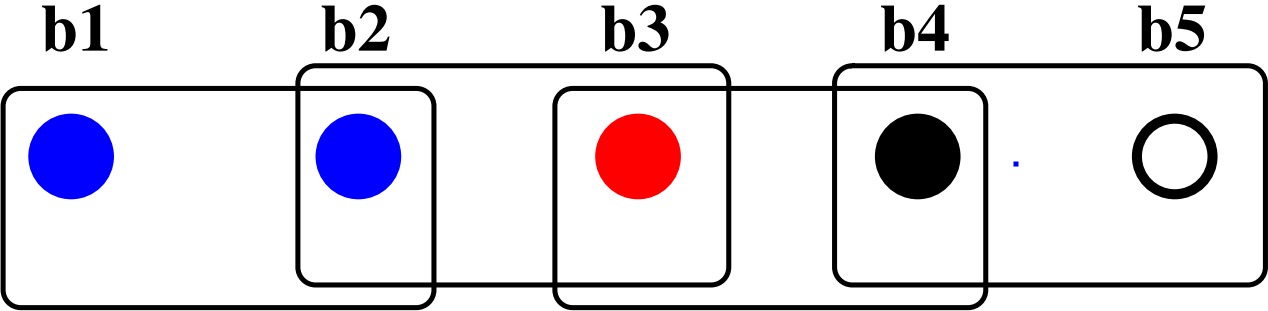


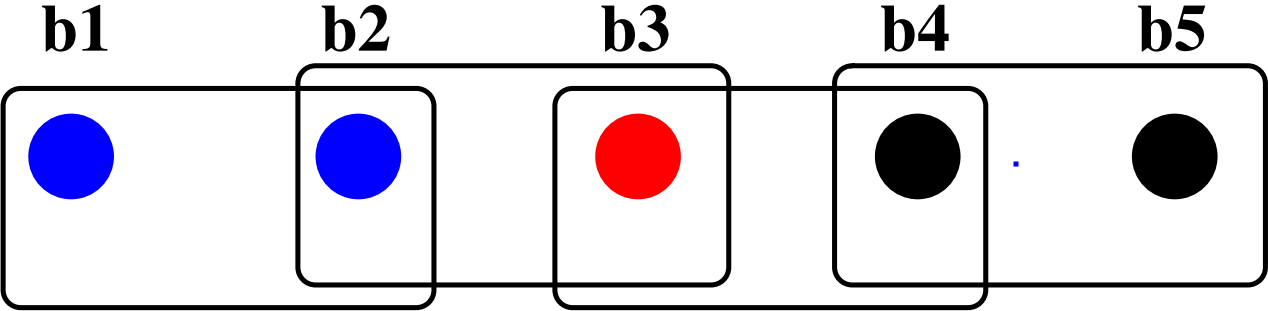


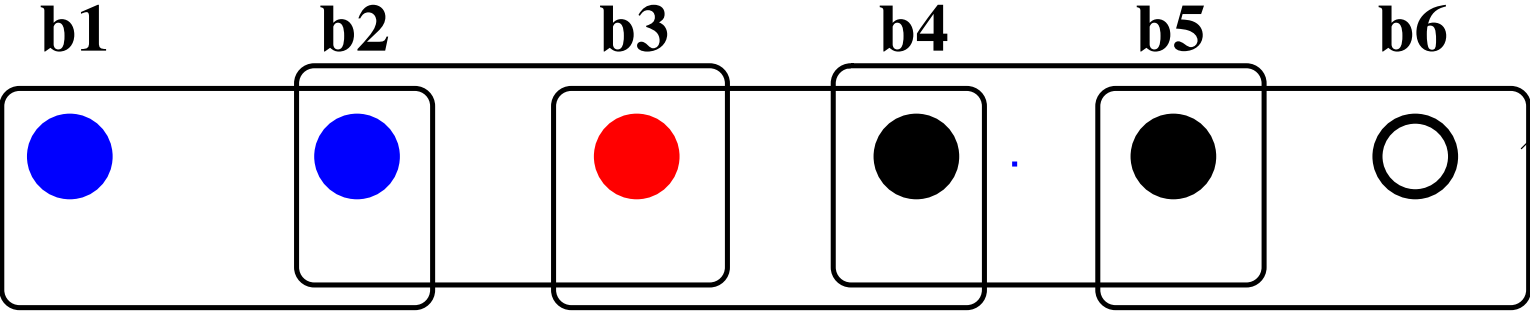


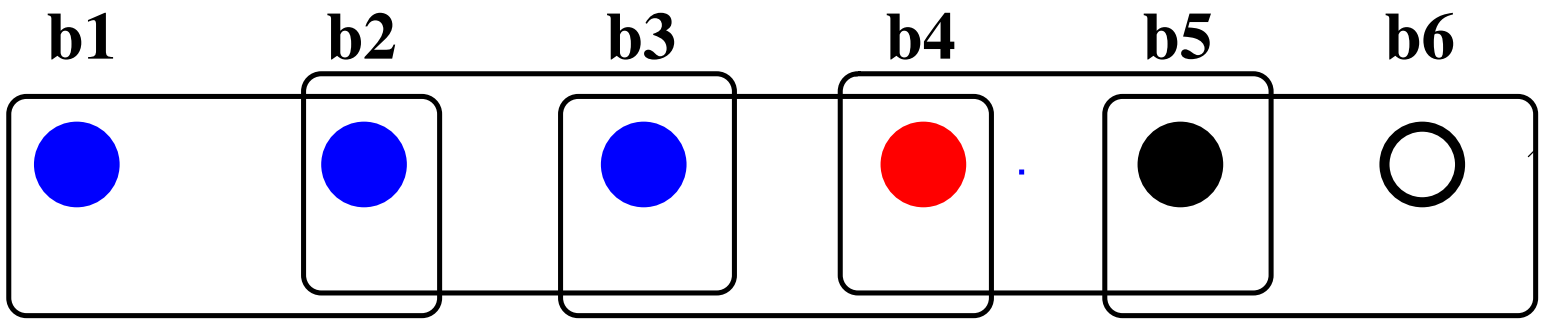


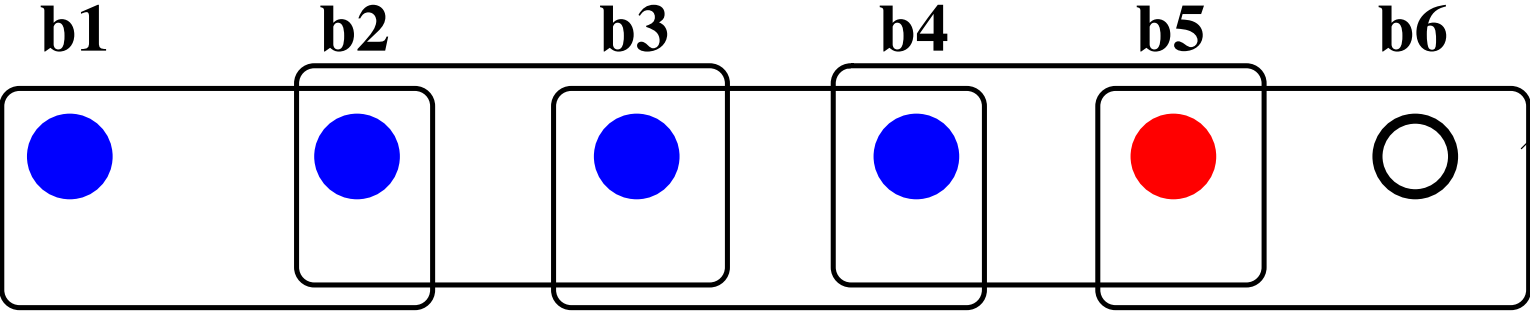


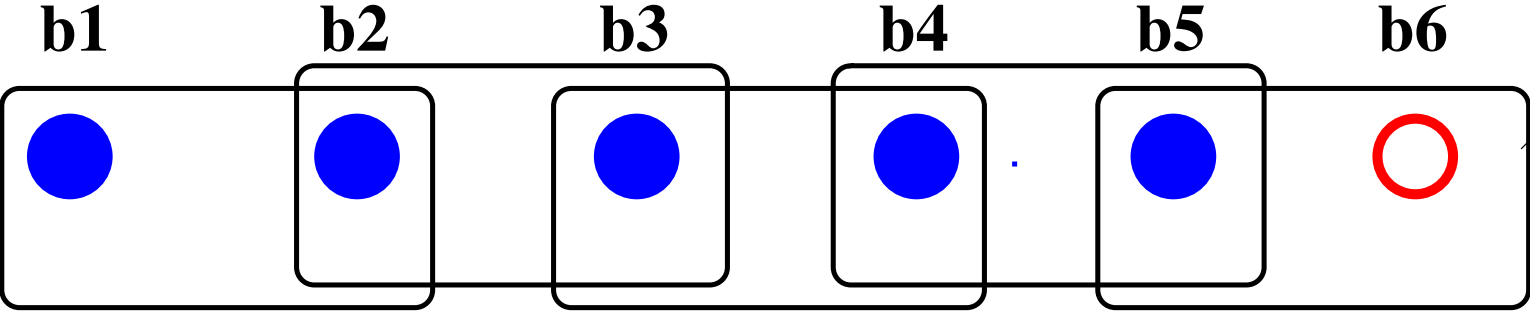


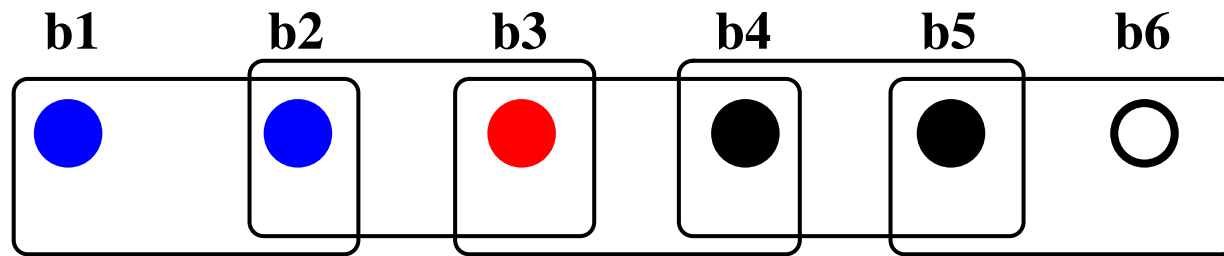




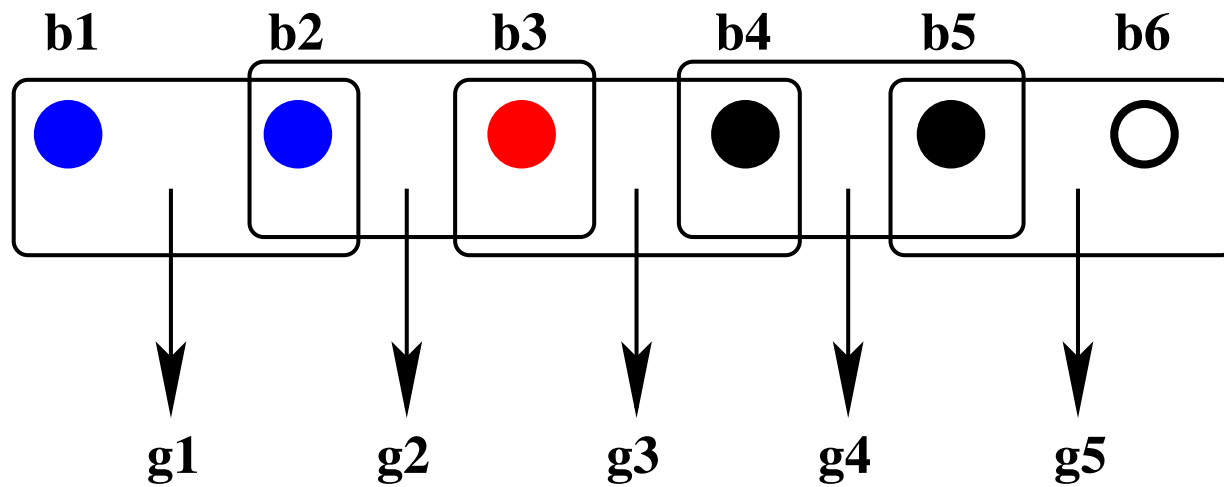








Abstract
(card)

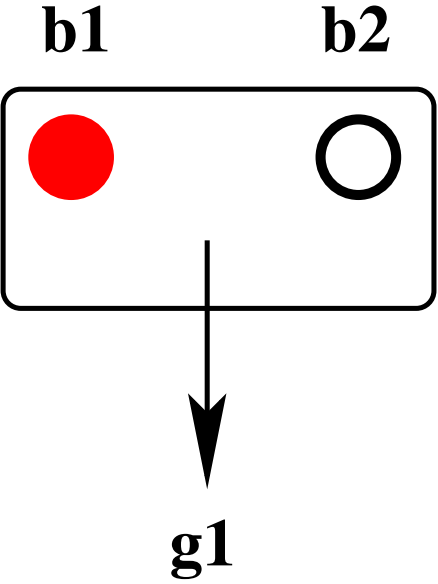


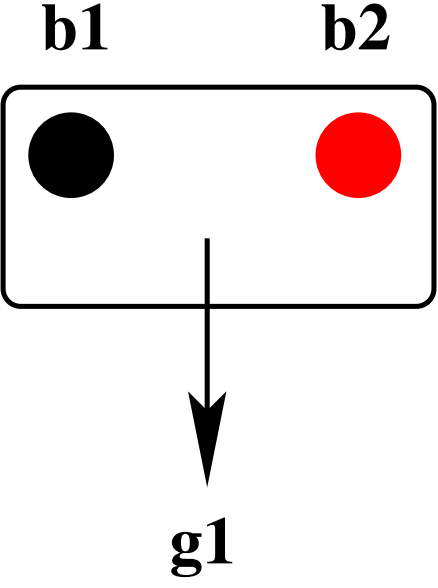
Concrete
(card and guest)

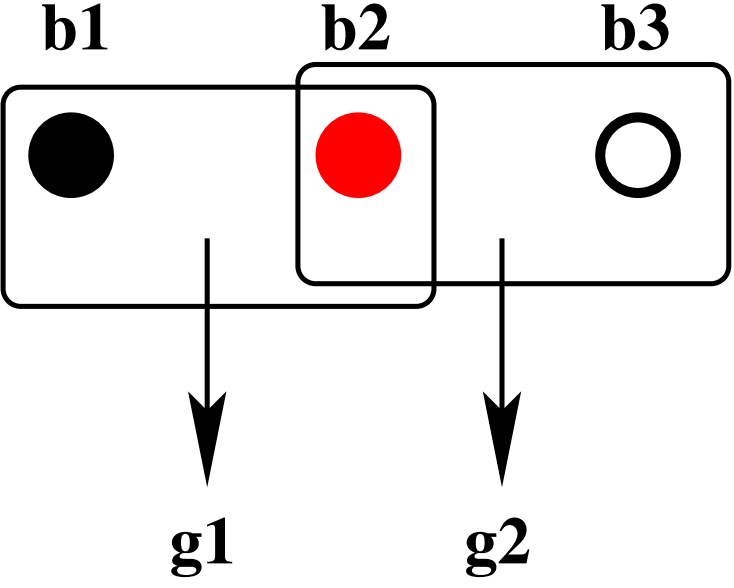
b1

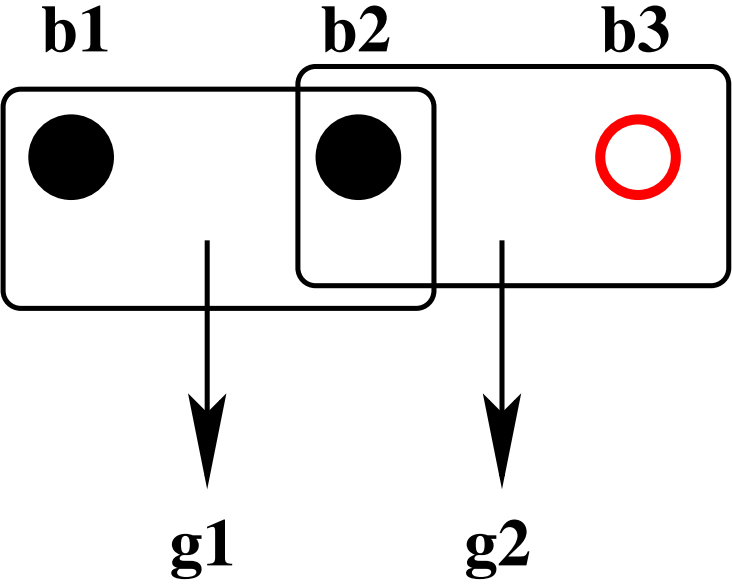


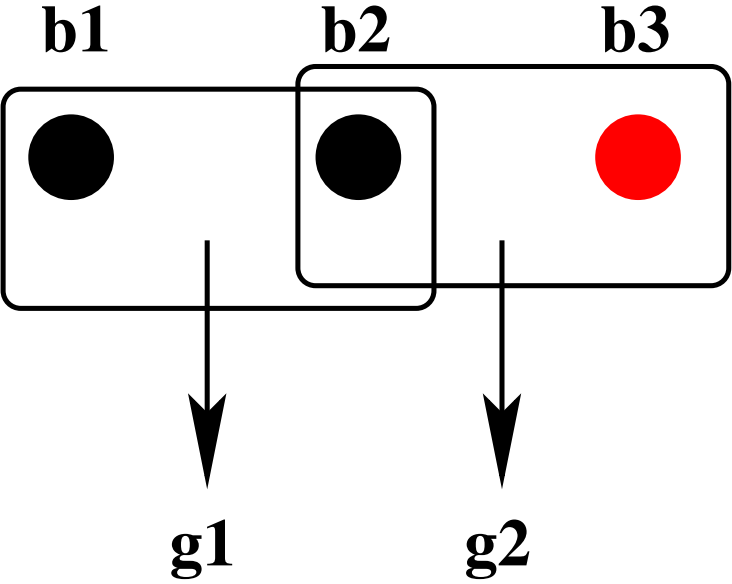
.

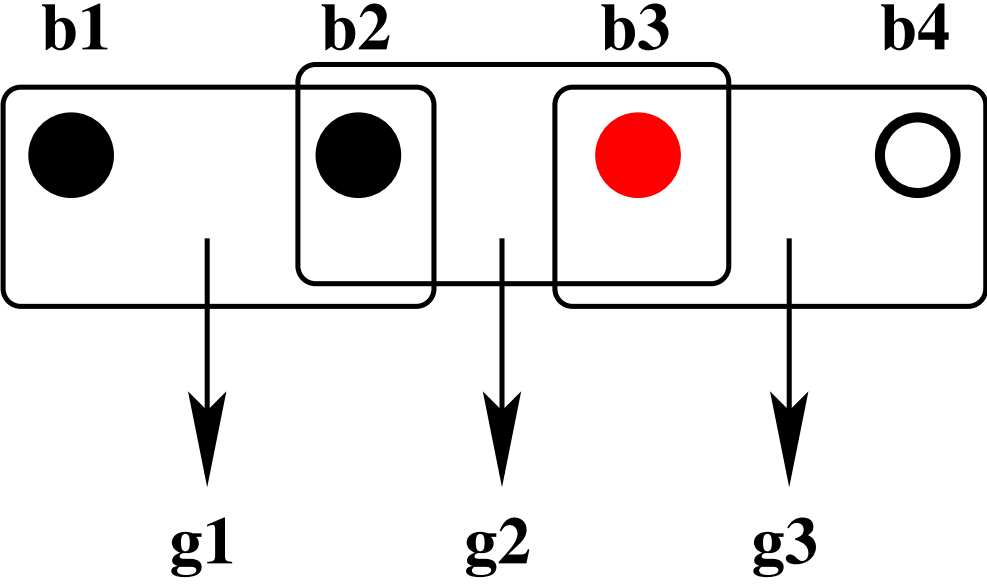


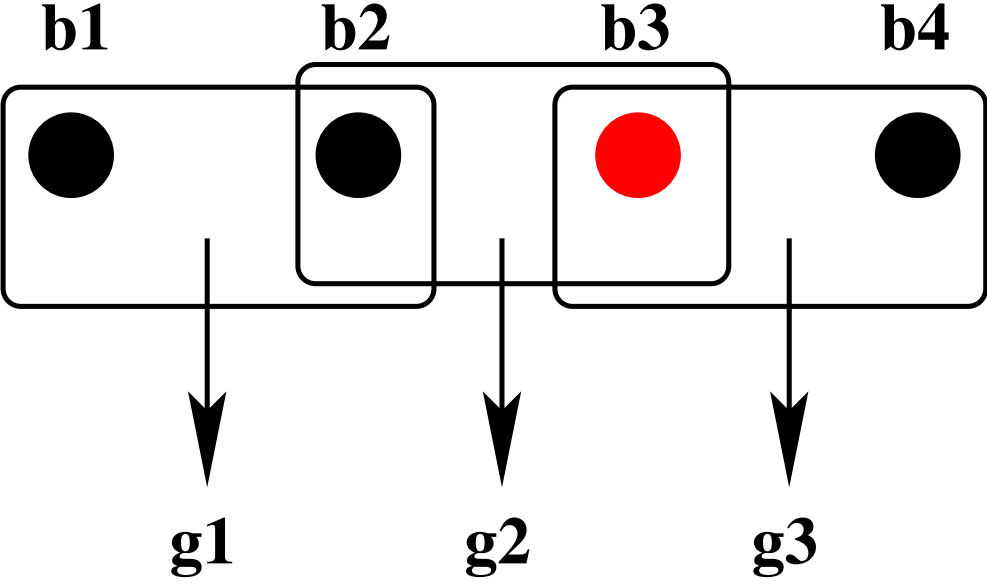


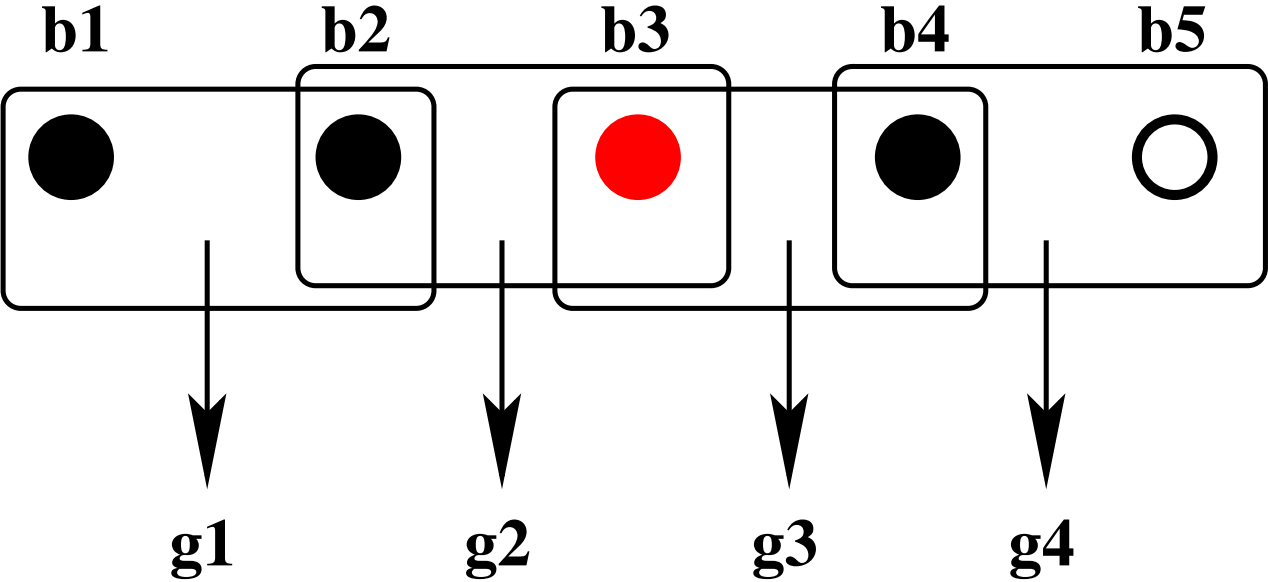


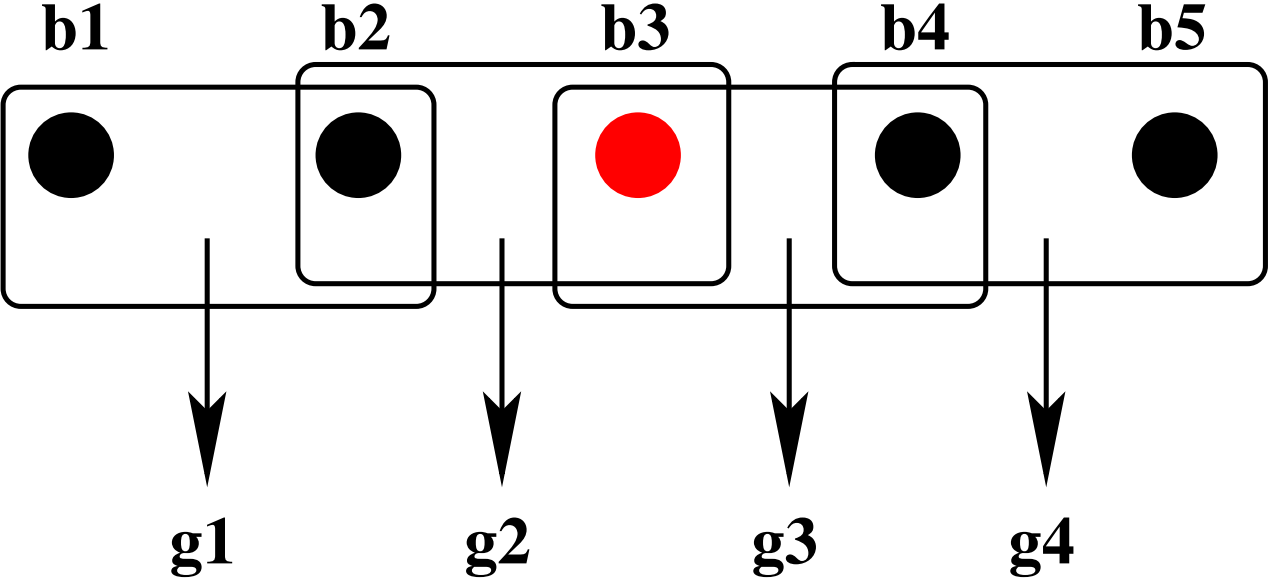


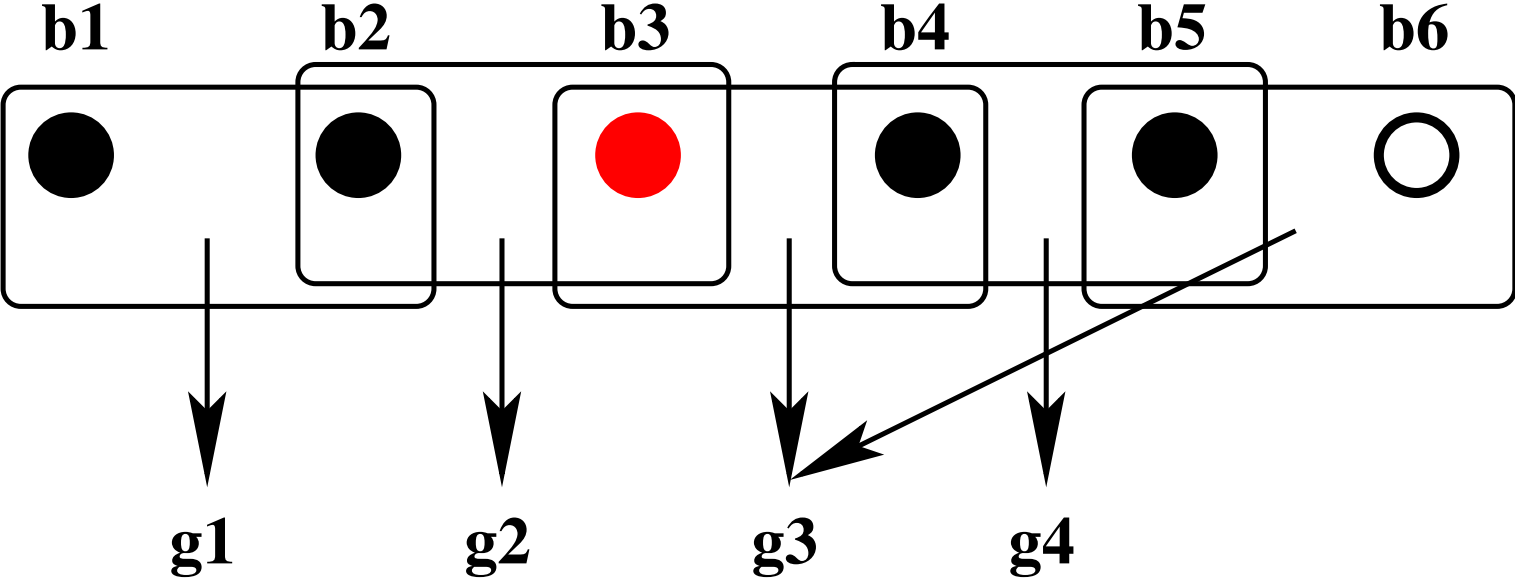


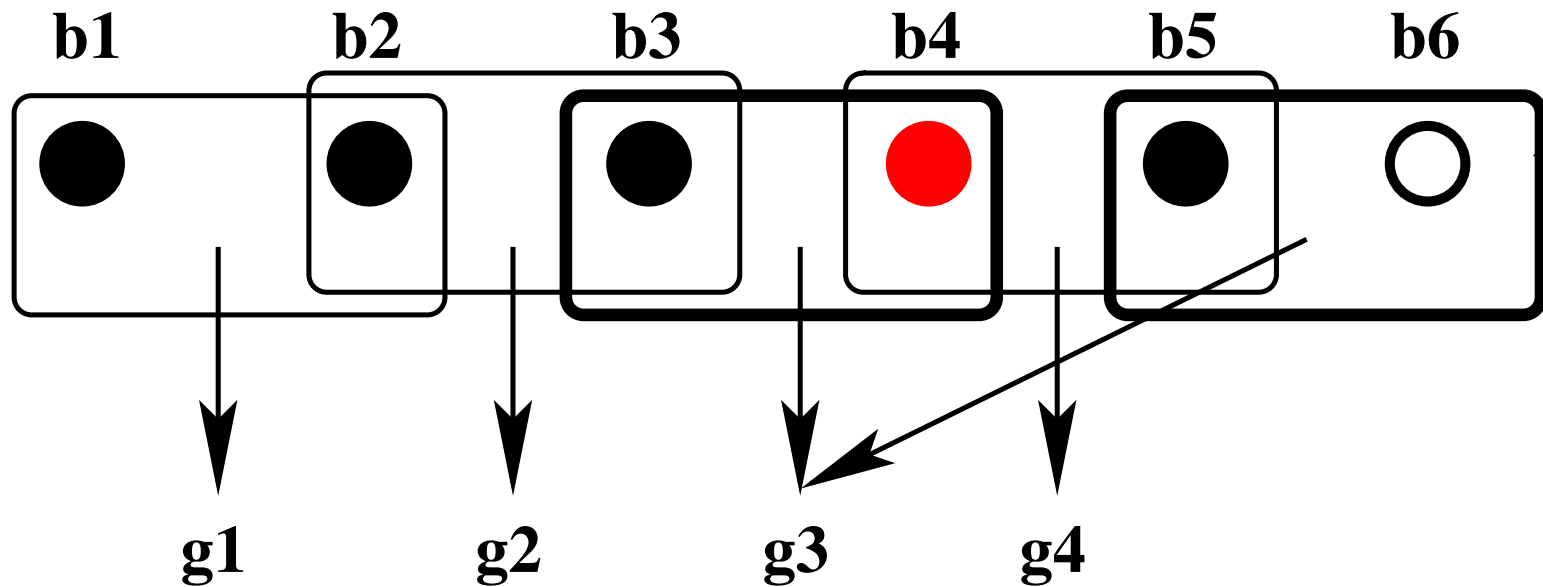












Here, we have a **serious problem**

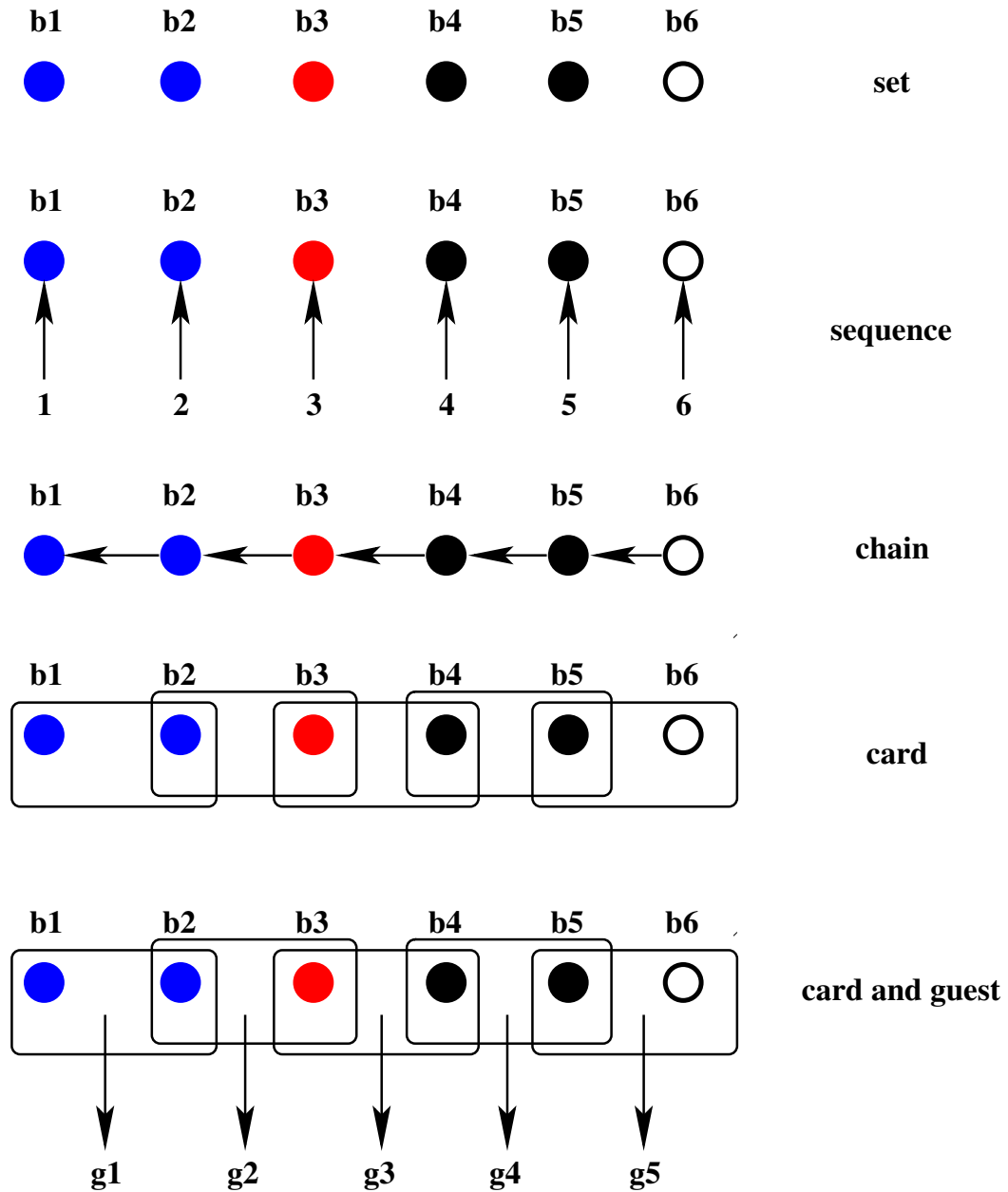
Guest g_3 has got **two cards** (he never used the old one)

He figures out that **the new one does not open the room**

So, **he tries the old one**, which allows him to enter into the room he booked

But he is **not safe**: **guest g_4 can enter the room**

- Never use an old card, event if it opens the room
- Go back to the lobby
- Ask someone to open the door with a master-card



- We've **done our best** to build a **professional tool**
- Use it, **it's fun**
- We are in the process of constructing a **Rodin Platform web-site**
- Give us **any feedback**

Thanks for listening