NOTES TO STUDENTS

- Exams: July 17, August 22, (and probably September)
- watch out for "Systems Programming Lab" in Fall !!!

TECHNISCHE UNIVERSITÄT DRESDEN

Faculty of Computer Science Institute of Systems Architecture, Operating Systems Group

MODELING DISTRIBUTED SYSTEMS

HERMANN HÄRTIG, DISTRIBUTED OPERATING SYSTEMS, SS2018

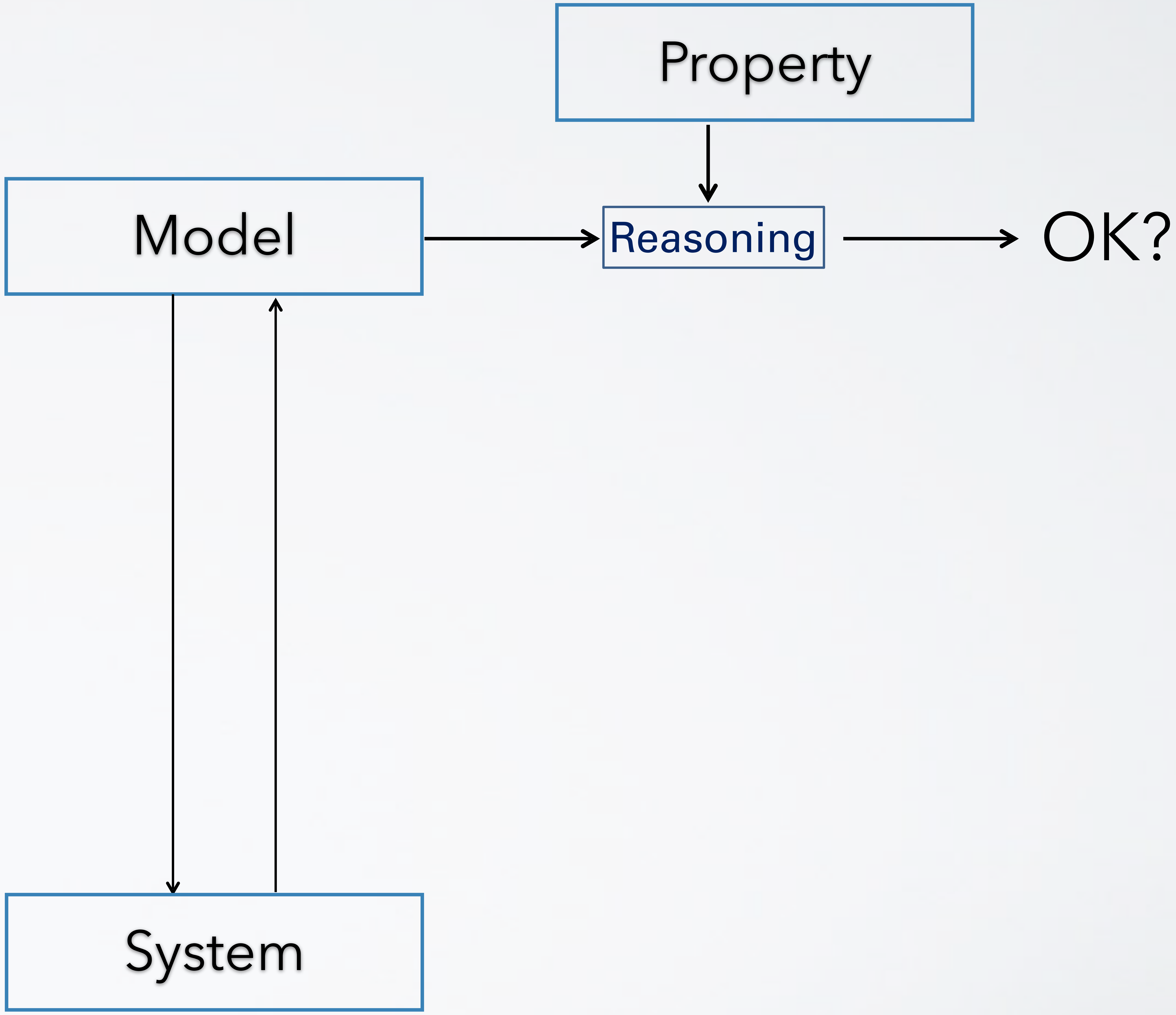
MODELS IN GENERAL

- abstract from details
- concentrate on functionality, properties, … that are considered important for a specific system/application
- use model to analyze, prove, predict, … system properties
  and to establish fundamental insights
- models in engineering disciplines very common, increasingly in CS as well
- we'll see many models in "Real-Time Systems" class

THE GENERAL IDEA

Reasoning:

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

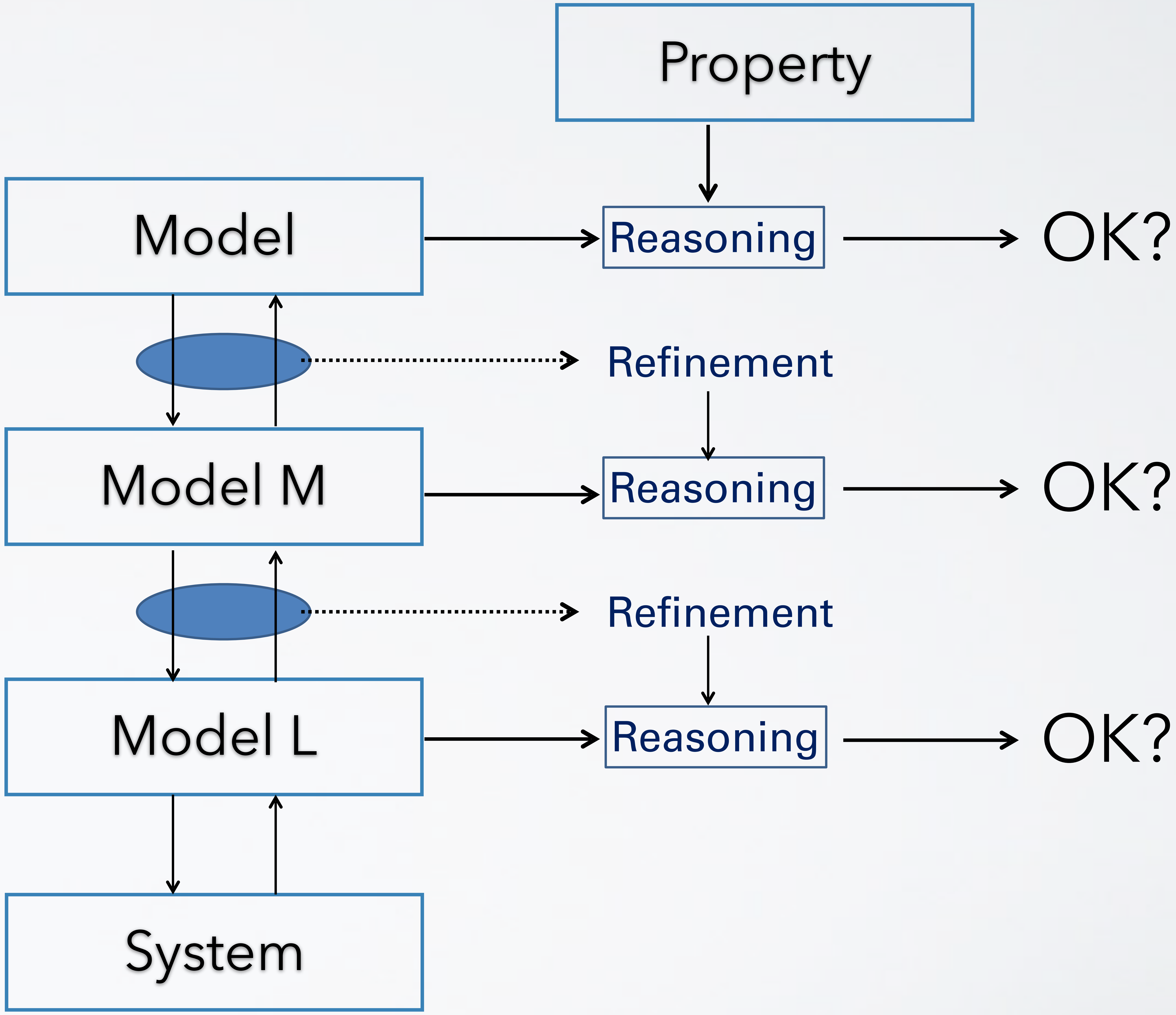


THE GENERAL IDEA

Reasoning:

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

- “Refinement”:
- Abstraction



MODEL EXAMPLES IN GENERAL

Model	Objective/Question
▪ Failure Trees	are all failure combinations taken into account
▪ statics models	does a house eventually fall down what kind of vehicles on a bridge
▪ control laws	stability of controllers
▪ Ohm's Law	behavior of circuits

WELL KNOWN EXAMPLES FOR MODELS

I=V/R

MODEL EXAMPLES COMPUTER SCIENCE

Model	Objective/Question
▪ Turing Machine	Decidability
▪ Amdahl's Law	Scalability
▪ Logic	Correctness, Precision, …
▪ Real-Time "tasks"	can all timing requirements be met
▪ Byzantine Agreement	Consensus
Two Army	Consensus

MODELS IN DOS

- Objective of lecture:
  understand the power of models and the need for their careful understanding
- Intuition, No proofs

THIS LECTURE'S QUESTIONS

- Q1: Is it possible to build arbitrarily reliable Systems out of unreliable components?
- Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?
- Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

2 Models per Question !

All questions/answers/models -> published 1956 - 1982 !!!

LIMITS OF RELIABILITY

Q1: Can we build arbitrarily reliable Systems out of unreliable components ?

- How to build reliable systems from less reliable components
- Fault(Error, Failure, Fault, ….)
  terminology in this lecture synonymously used for "something goes wrong"
  (more precise definitions and types of faults in SE)

DEFINITIONS

Reliability:

- R(t): probability for a system to survive time t

Availability:

- A: fraction of time a system works

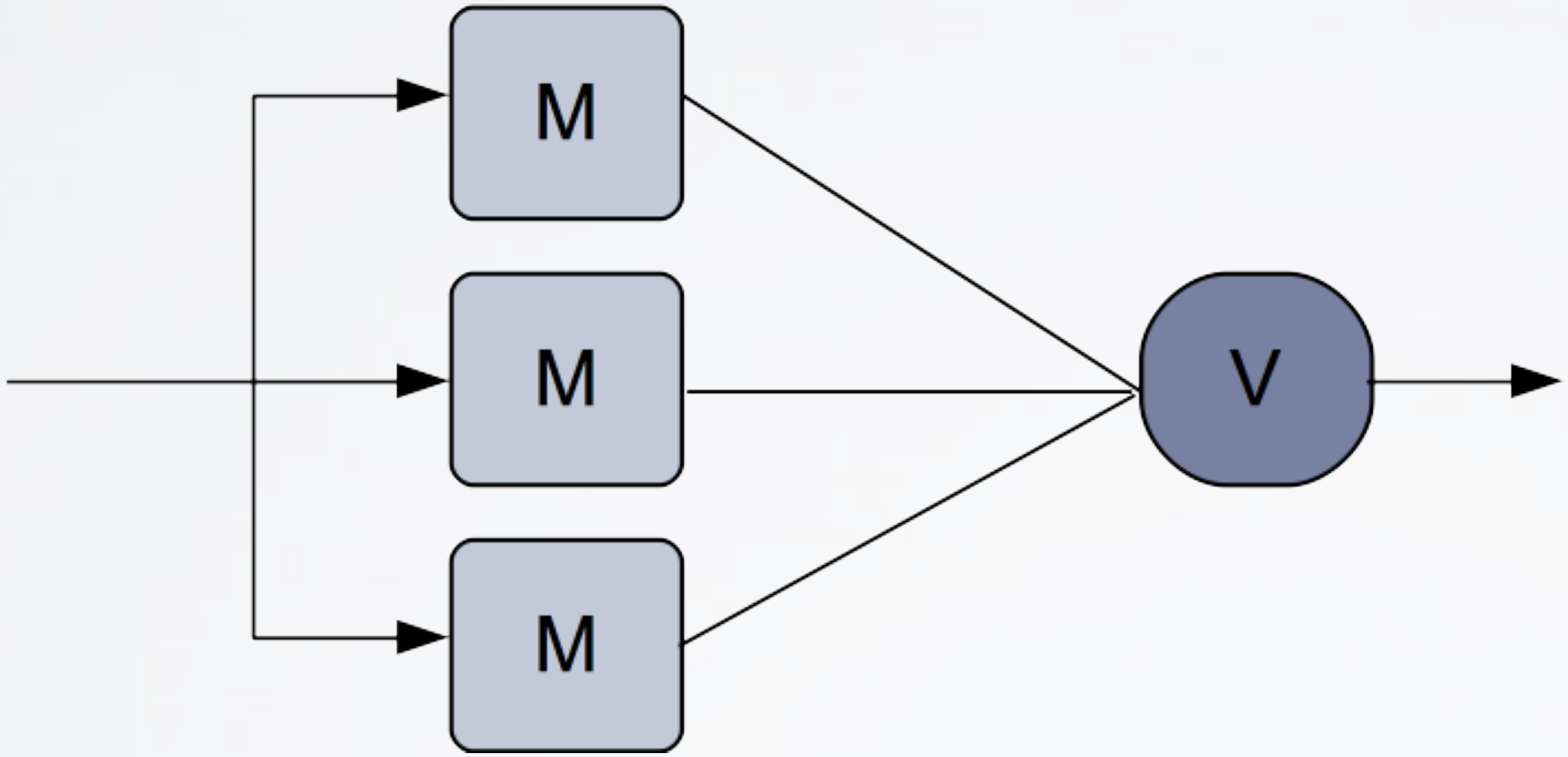
INGREDIENTS

- Fault detection and confinement
- Recovery
- Repair

- Redundancy
  - Information
  - time
  - structural
  - functional

WELL KNOWN EXAMPLE

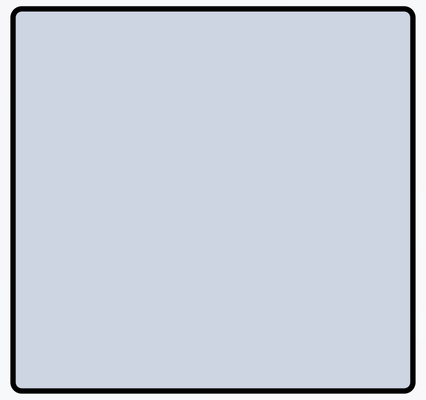
John v. Neumann
Voter: *single point of failure*



Can we do better
→ distributed solutions?

Parallel-Serial-Systems

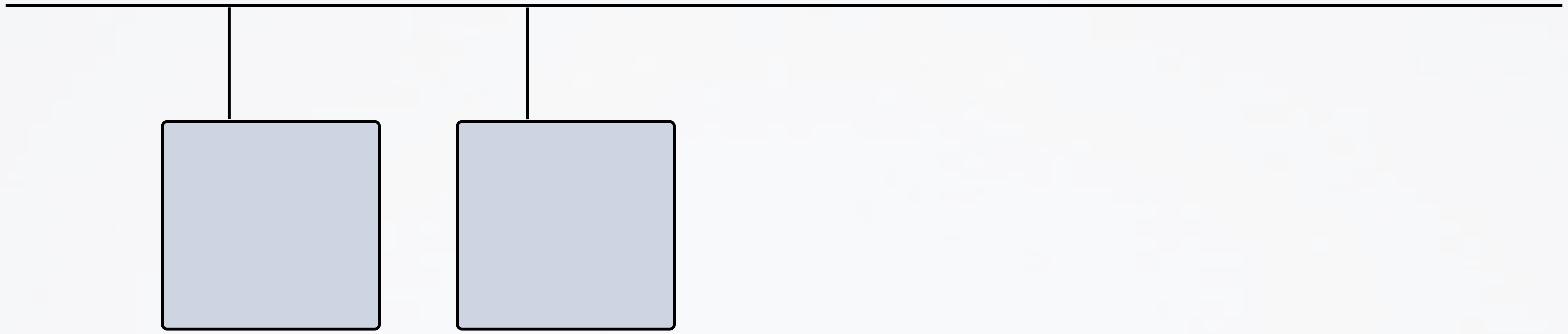
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

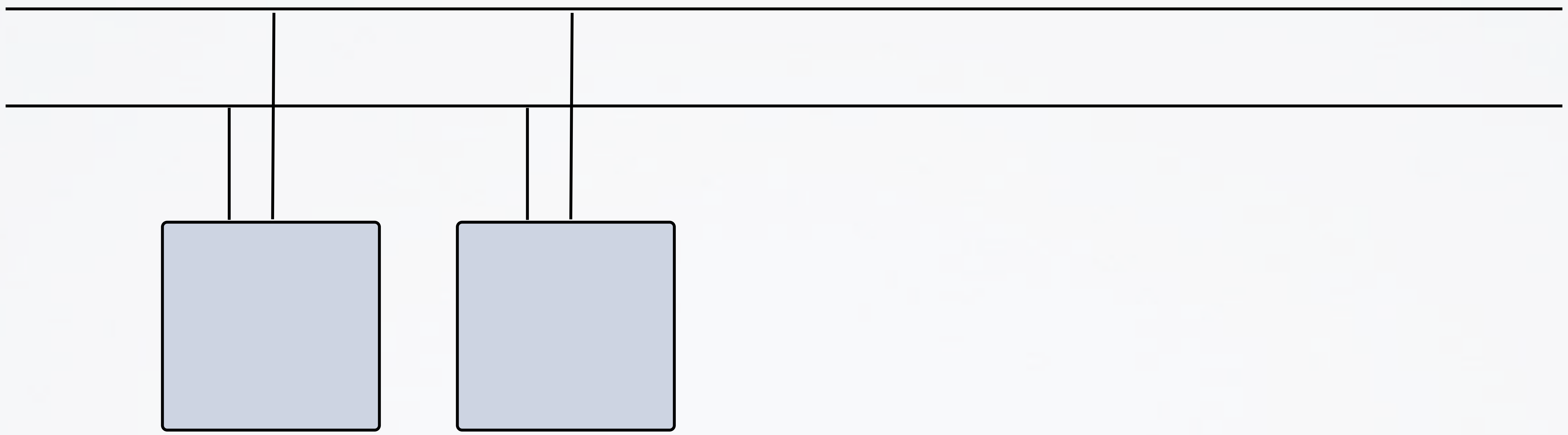
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

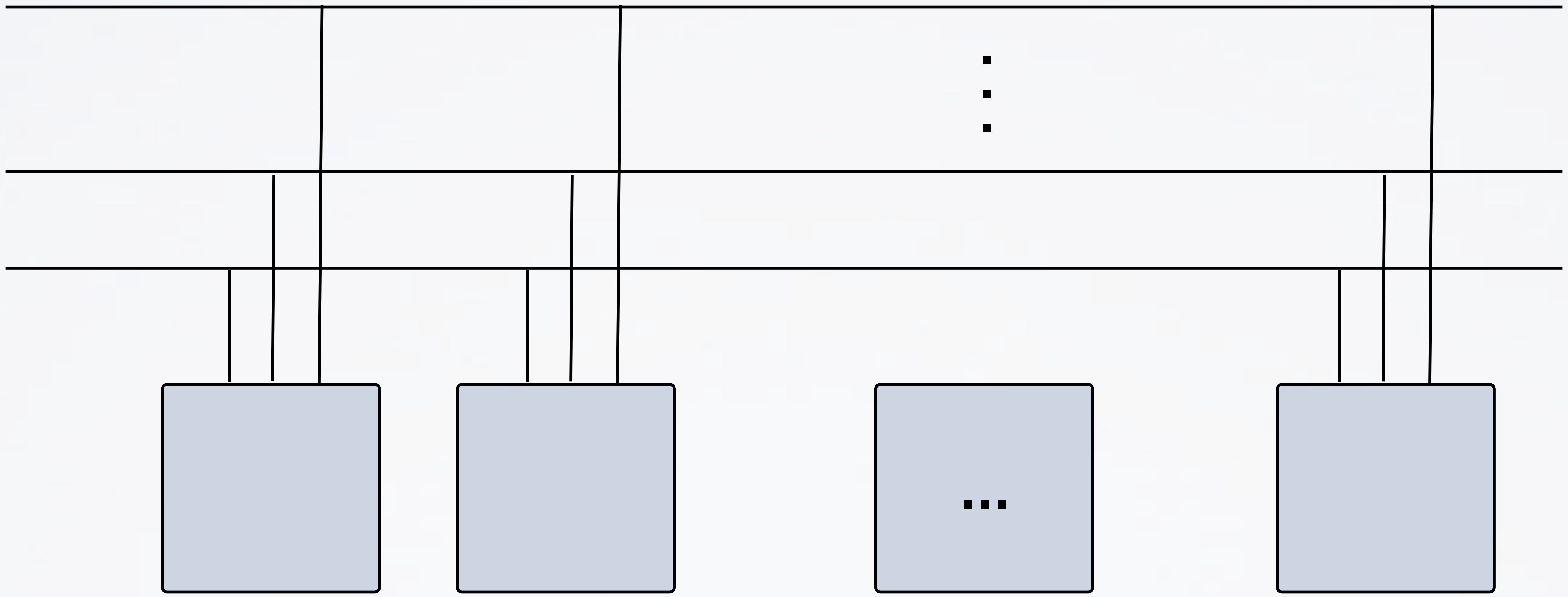
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

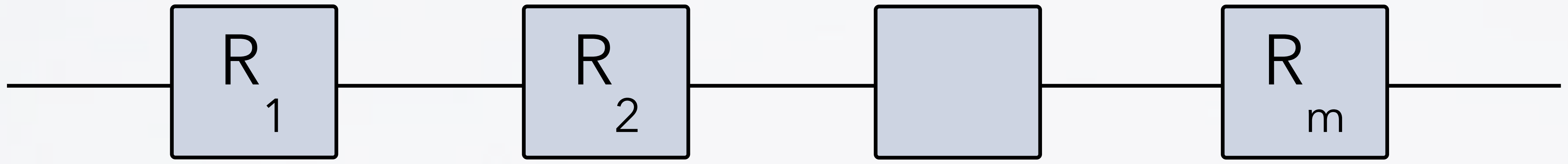
Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)



Q1/MODEL1: ABSTRACT RELIABILITY MODEL

Serial-Systems

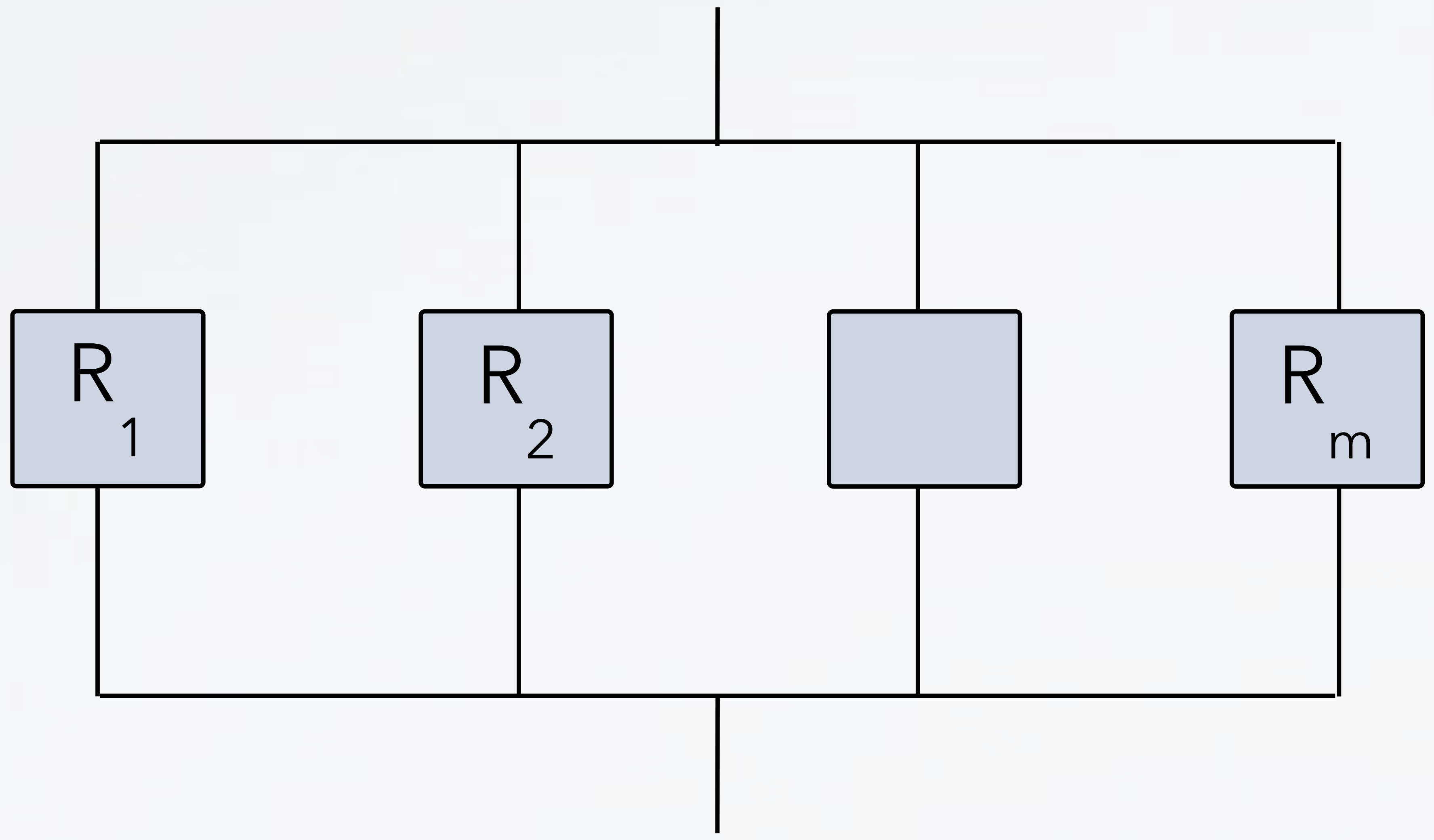


$$R_{whole} = \prod_{j=1}^{m} R_j$$

Each component must work for the whole system to work.

Q1/MODEL1: ABSTRACT MODEL

Parallel-Systems



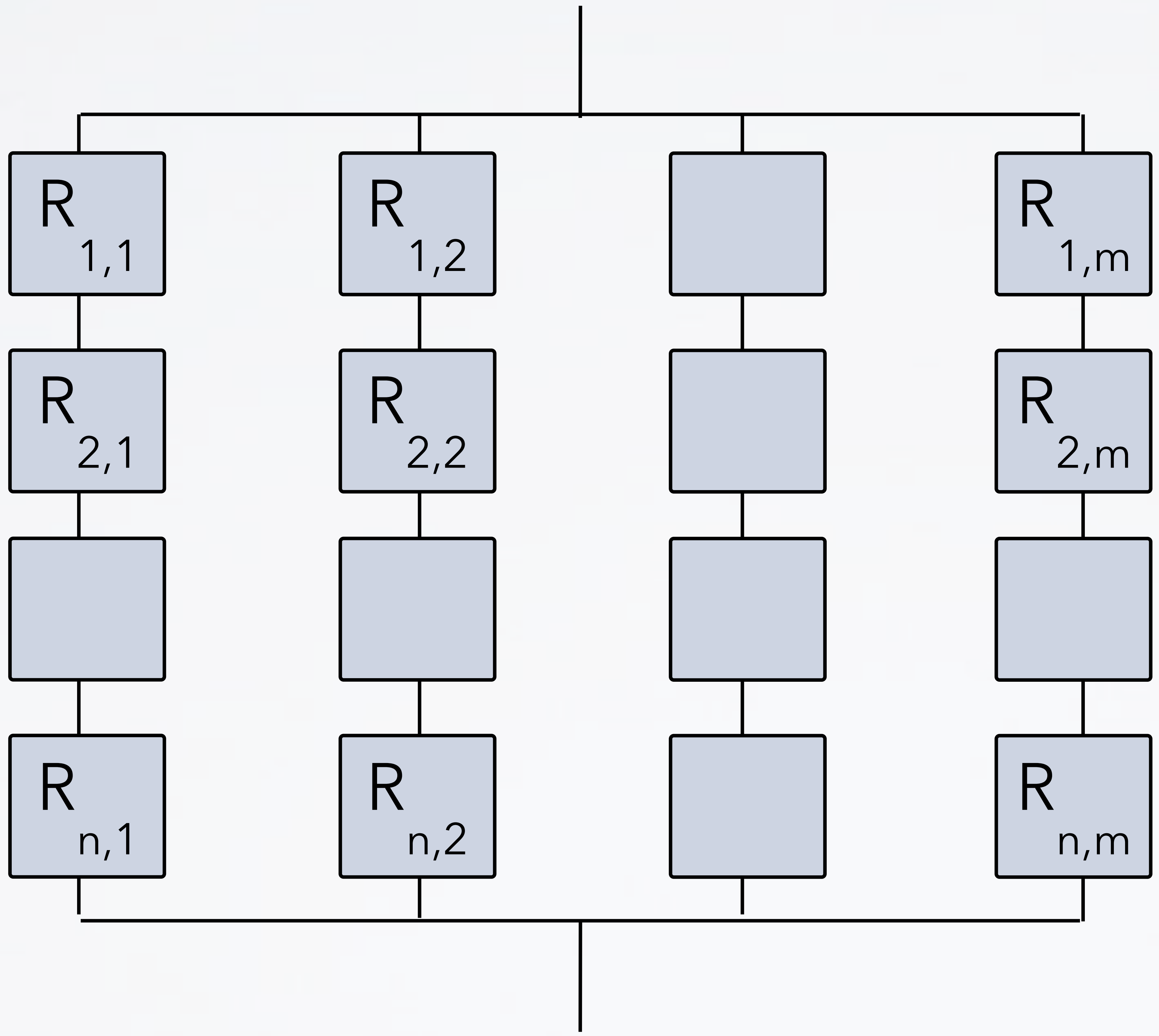
$$R_{whole} = 1 - \prod_{i=1}^{m} (1 - R_i)$$

One component must work for the whole system to work.

Each component must fail for the whole system to fail.

Q1/MODEL1: ABSTRACT MODEL

Serial-Parallel-Systems

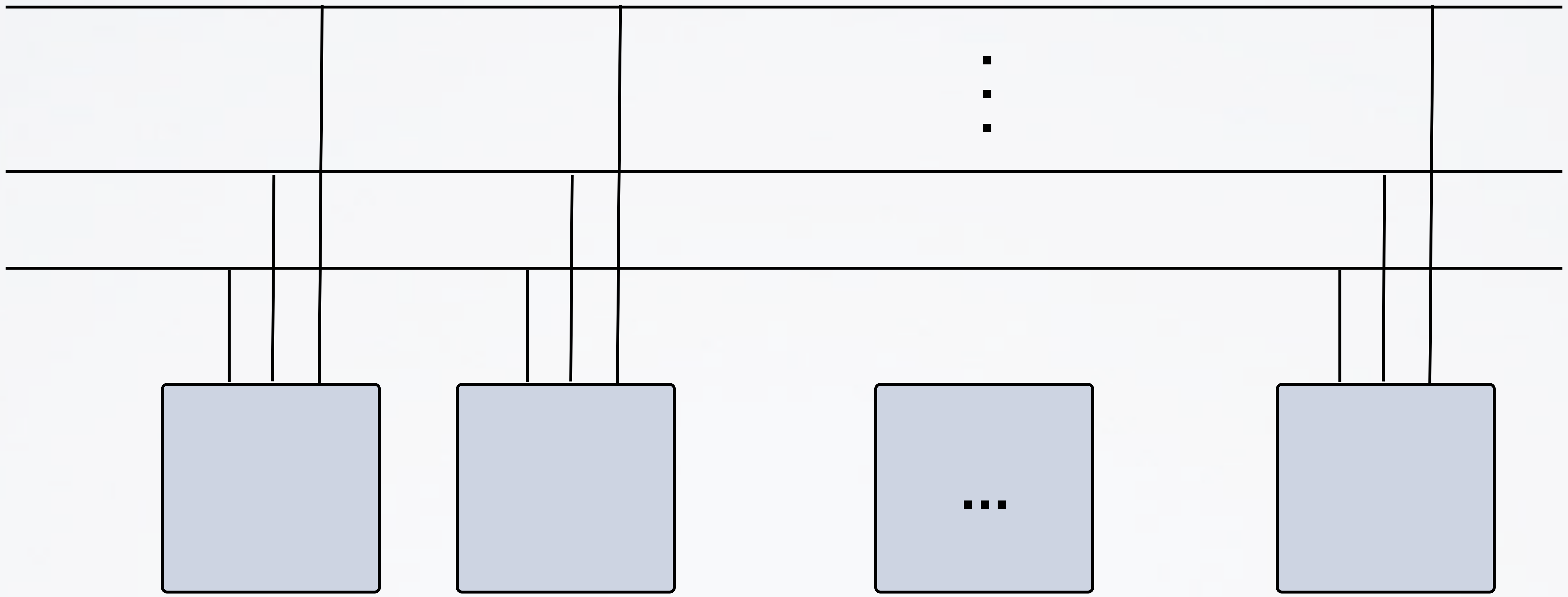


$$R_{whole} = 1 - \prod_{j=1}^{m} \left(1 - \prod_{i=1}^{n} R_{i,j} \right)$$

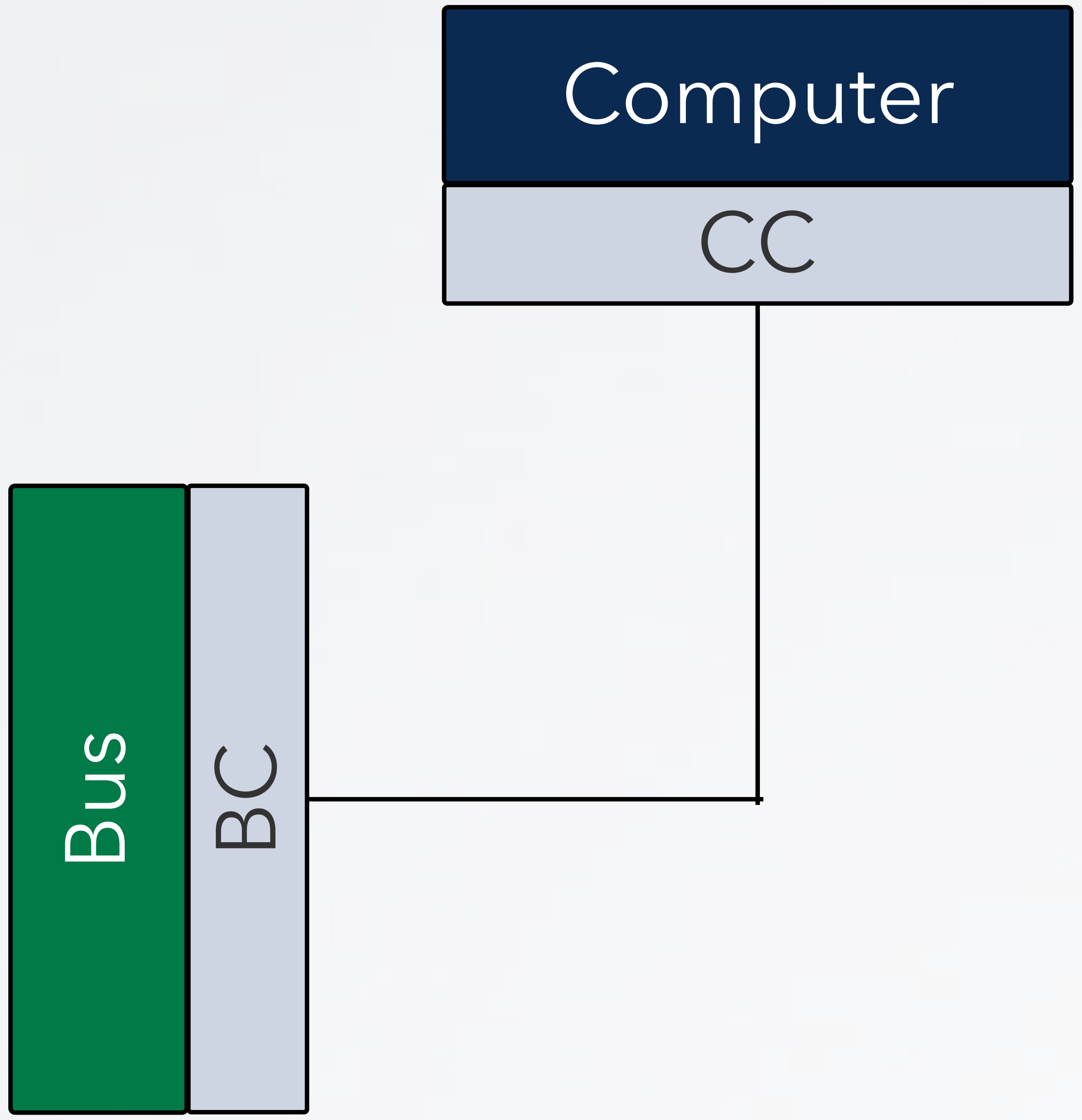
Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)



Q1/MODEL1: CONCRETE MODEL



Fault Model

„Computer-Bus-Connector"
can fail such that Computer and/or Bus also fail

=>

conceptual separation of components into

Computer, Bus: can fail per se

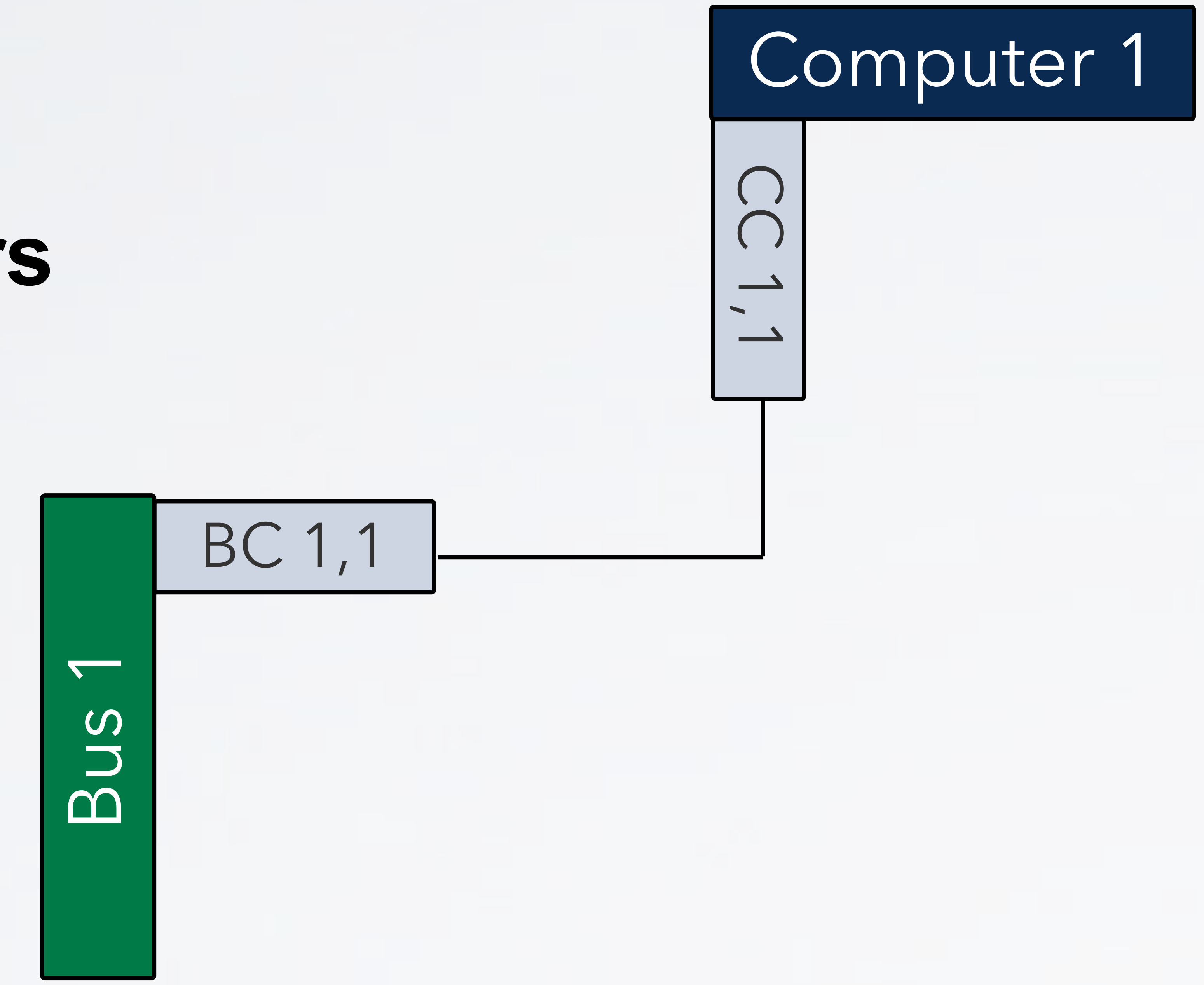
CC: Computer-Connector
fault also breaks the Computer

BC: Bus-Connector
fault also breaks Bus

Q1/MODEL1: CONCRETE MODEL

1 Buses

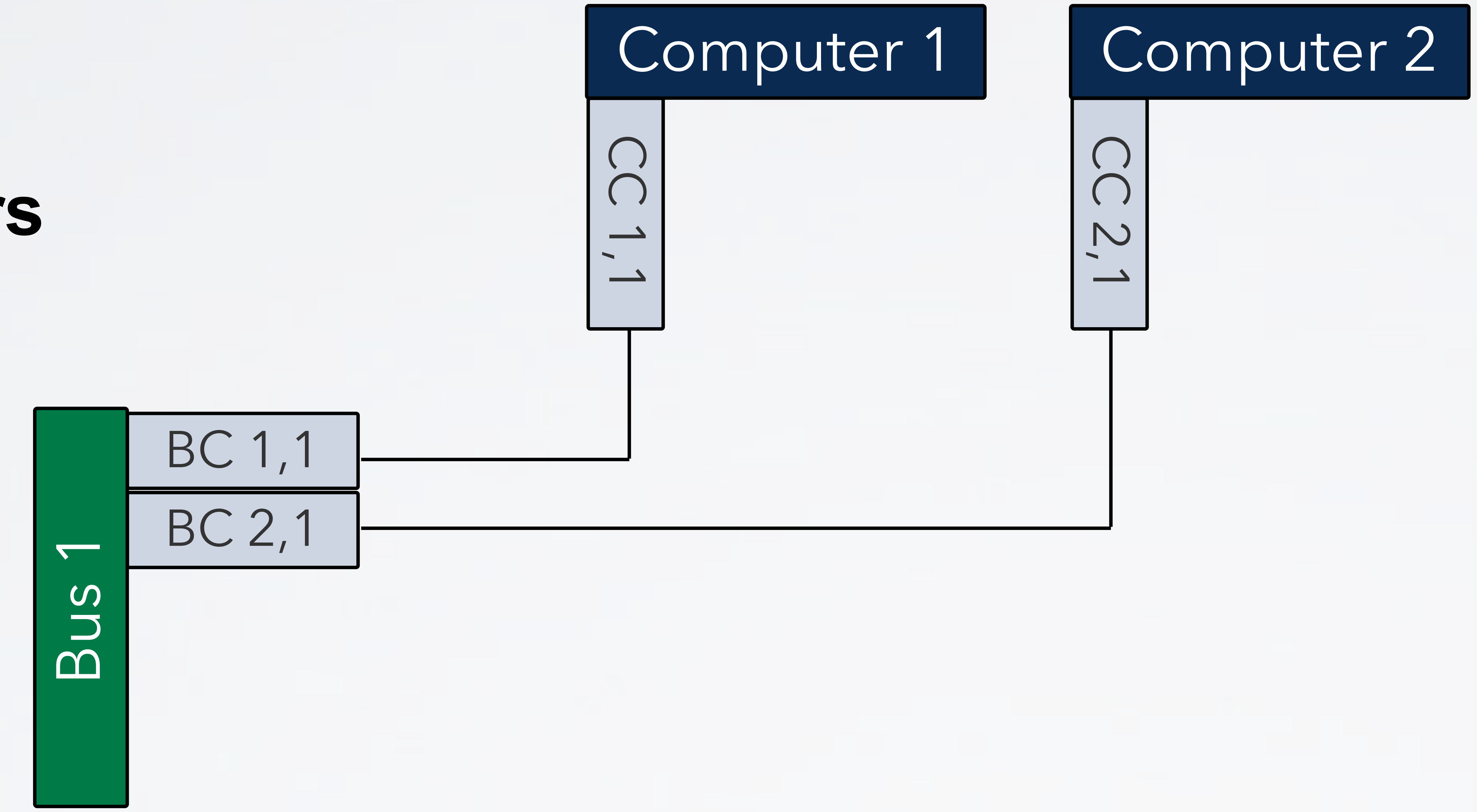
1 Computers



Q1/MODEL1: CONCRETE MODEL

1 Buses

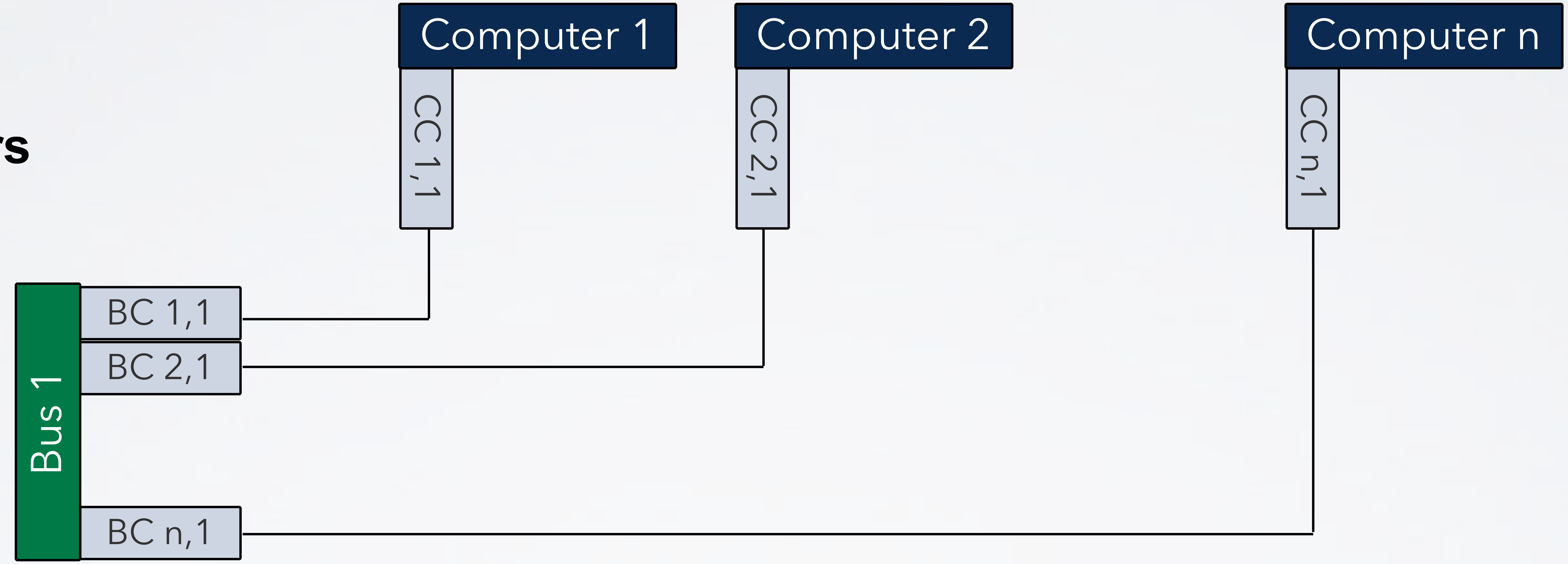
2 Computers



Q1/MODEL1: CONCRETE MODEL

1 Buses

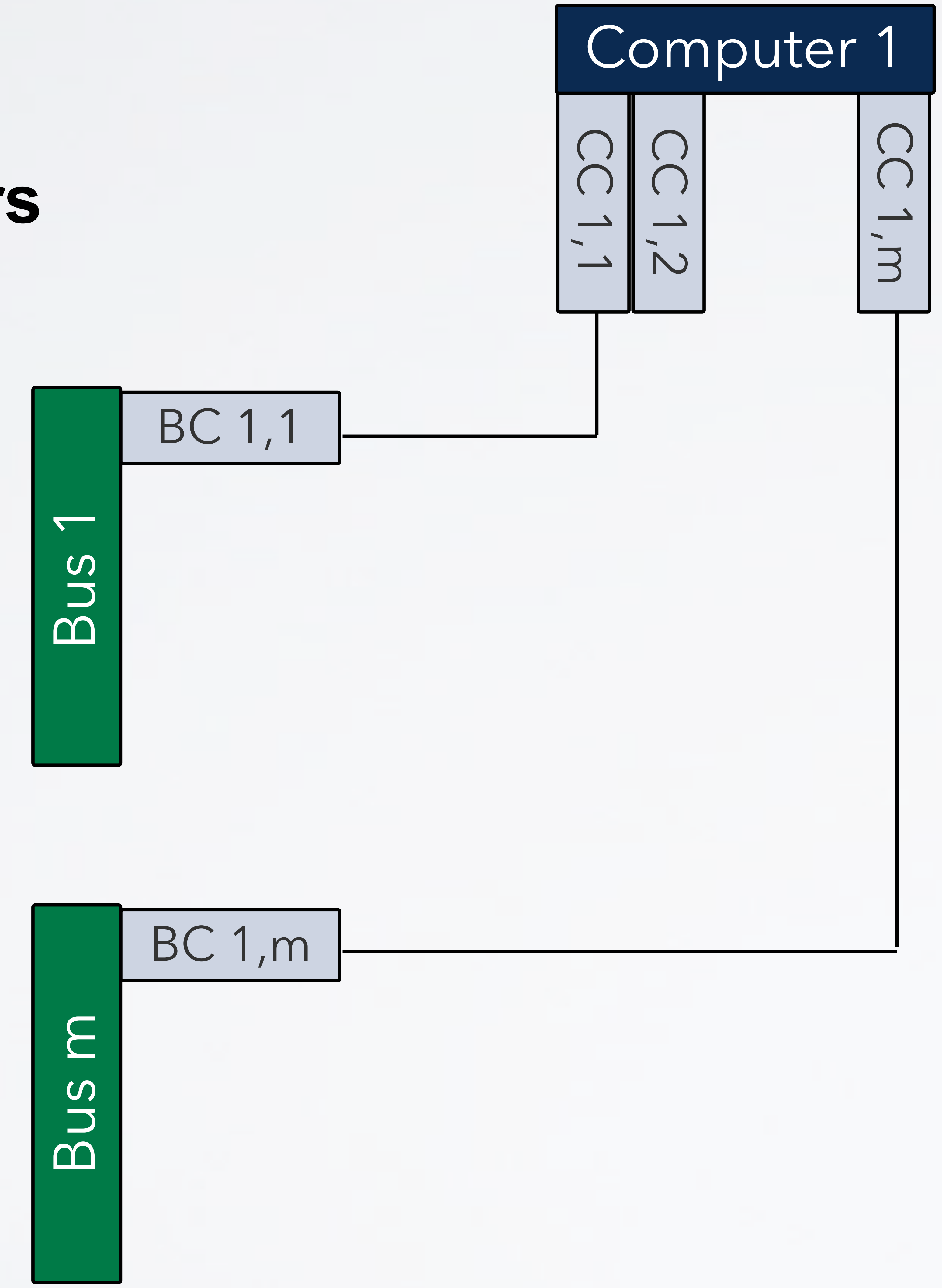
N Computers



Q1/MODEL1: CONCRETE MODEL

M Buses

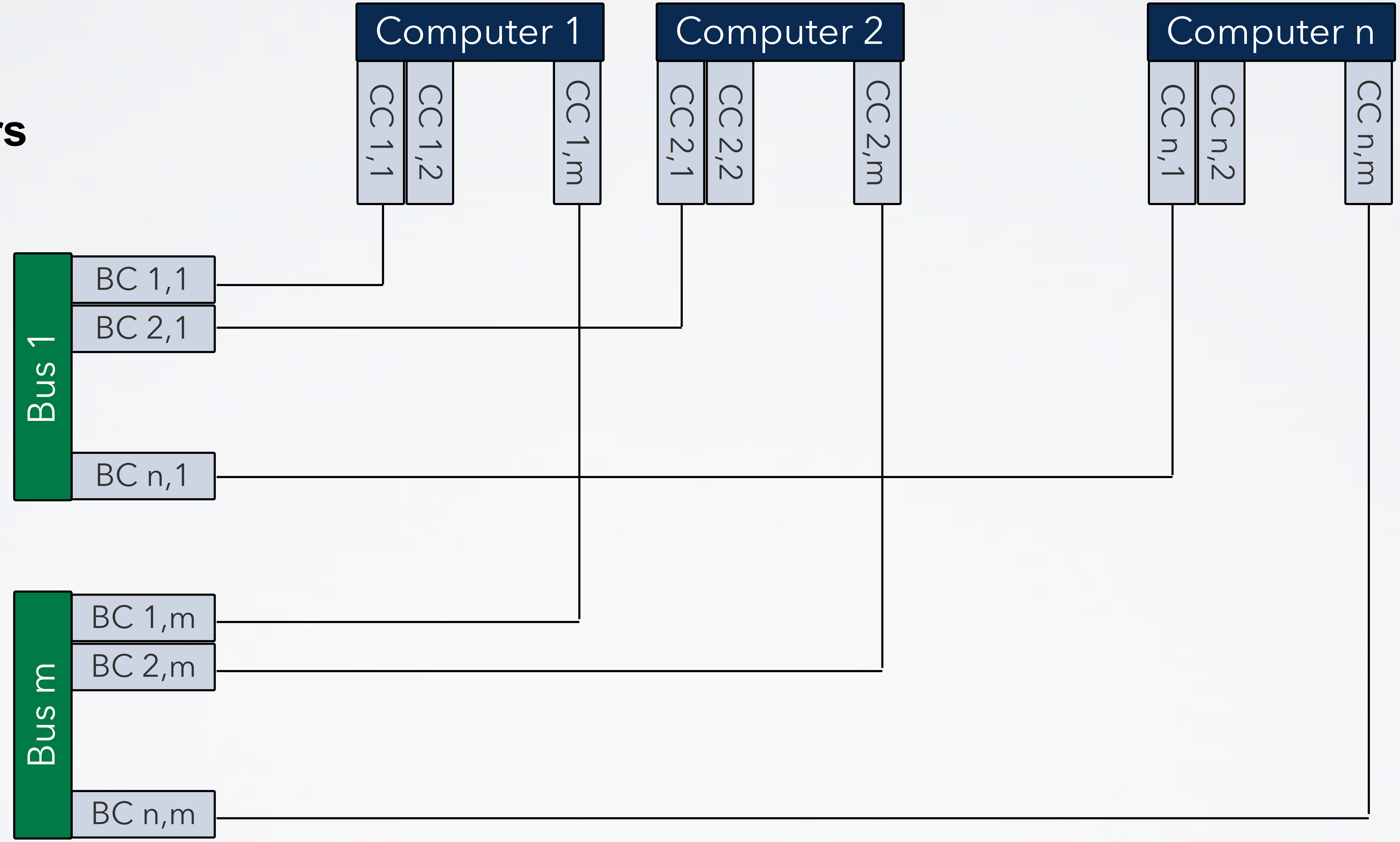
1 Computers



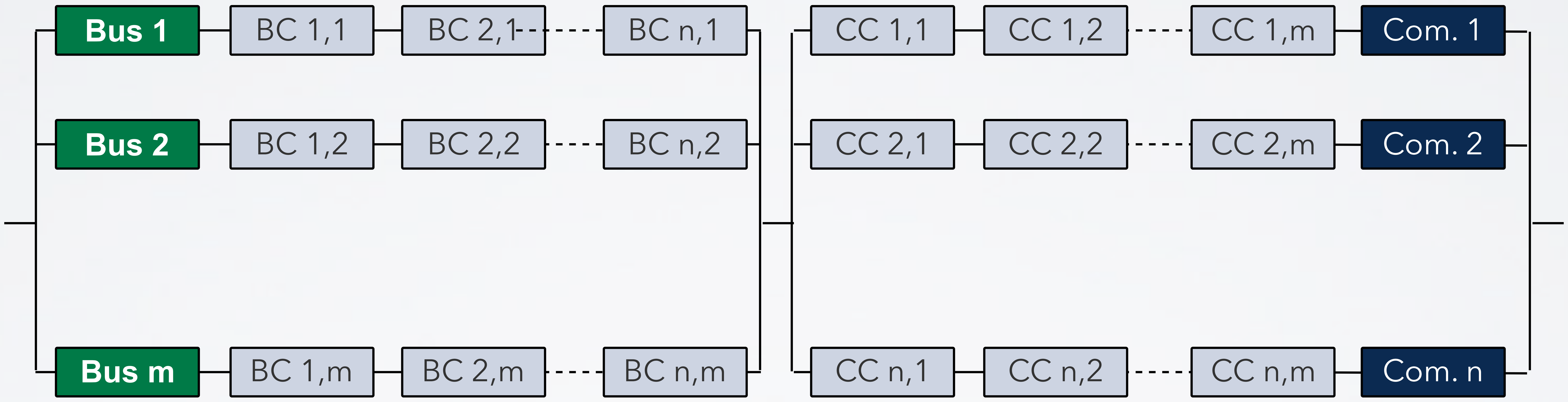
Q1/MODEL1: CONCRETE MODEL

M Buses

N Computers



Q1/MODEL1: CONCRETE MODEL FOR N,M



$$R_{whole}(n,m) = \left(1-\left(1-R_{Bus}\cdot R_{BC}^{n}\right)^{m}\right)\cdot\left(1-\left(1-R_{Computer}\cdot R_{CC}^{m}\right)^{n}\right)$$

$$then:\ R_{CC}, R_{BC} < 1:\ \lim_{n,m\to\infty} R(n,m) =$$

Q1/MODEL2: LIMITS OF RELIABILITY

- System built of Synapses (John von Neumann, 1956)
- Computation and Fault Model :
  - Synapses deliver „0“ or „1”
  - Synapses deliver with R > 0,5:
    - with probability R correct result
    - with (1-R) wrong result
- Then we can build systems that deliver correct result for any (arbitrarily high) probability R

Q2: CONSENSUS

Q2: Can we achieve consensus in the presence of faults
all non-faulty components agree on action?

- all correctly working units agree on result/action
- agreement non trivial (based on exchange of messages)

Q2/MODEL 1: "2 ARMY PROBLEM"

- p,q processes
  - communicate using messages
  - messages can get lost
  - no upper time for message delivery known
  - do not crash, do not cheat
- p,q to agree on action (e.g. attack, retreat, ...)
- how many messages needed ?

- first mentioned: Jim Gray 1978

Q2/MODEL 1: "2 ARMY PROBLEM"

Result: there is no protocol with finite messages

Prove by contradiction:

- assume there are finite protocols (mp--> q, mq --> p)*
- choose the shortest protocol MP,
- last message MX: mp --> q or mq --> p
- MX can get lost
- => must not be relied upon => can be omitted
- => MP not the shortest protocol.
- => no finite protocol

Q2/MODEL 2: "BYZANTINE AGREEMENT"

n processes, f traitors, n-f loyals

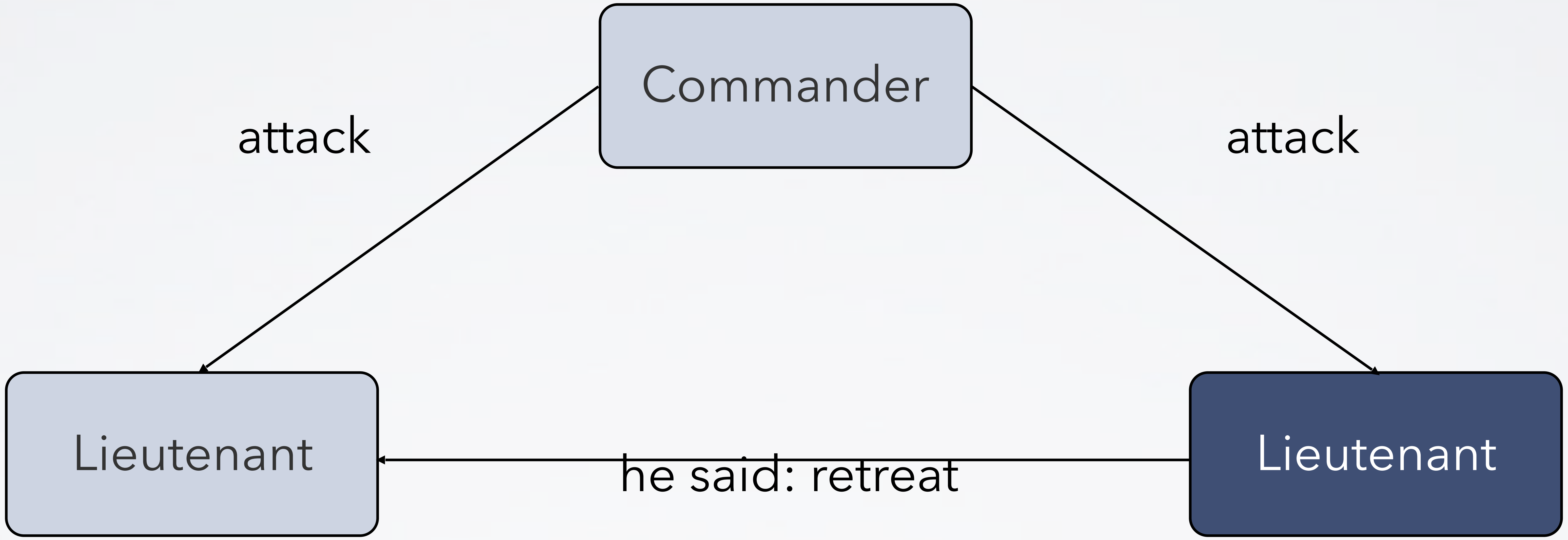
- communicate by reliable and timely messages (synchronous messages)
- traitors lye, also cheat on forwarding messages
- try to confuse loyals

Q2/MODEL 2: "BYZANTINE AGREEMENT"

Goal:

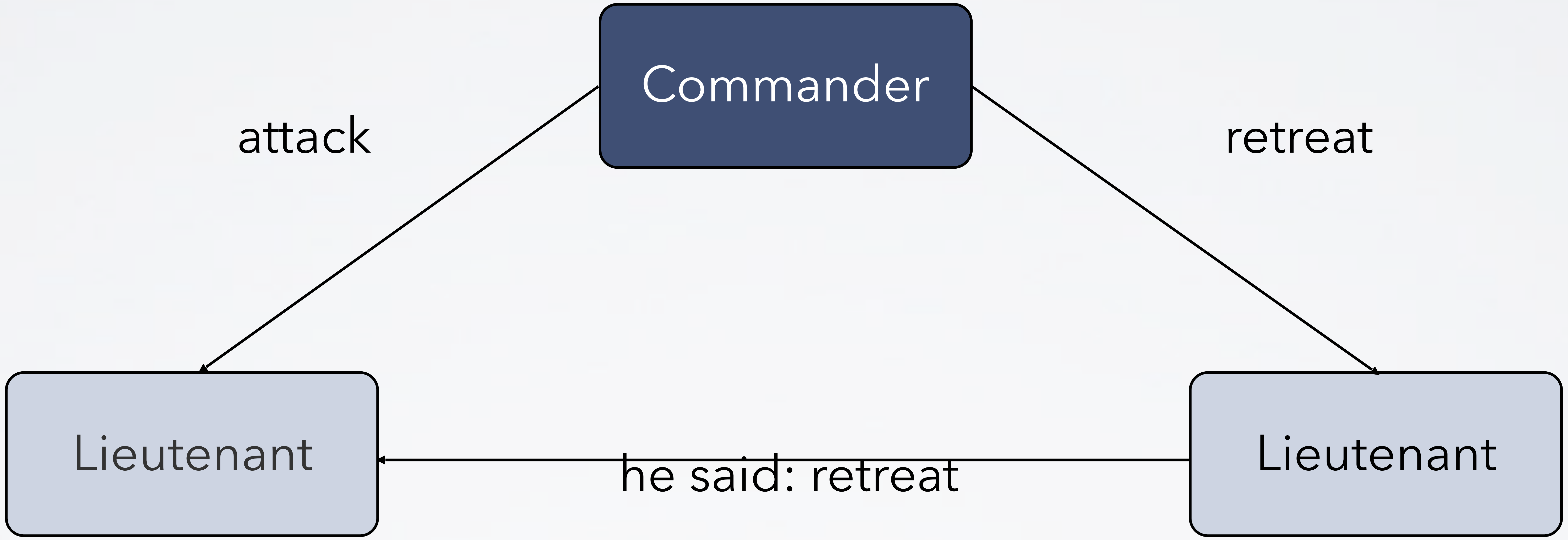
- loyals try to agree on non-trivial action (attack, retreat)
- non-trivial more specific:
  - one process is commander
  - if commander is loyal and gives an order, loyals follow the order otherwise loyals agree on arbitrary action

Q2/MODEL 2: “BYZANTINE AGREEMENT”



3 Processes: 1 traitor, 2 loyals

Q2/MODEL 2: "BYZANTINE AGREEMENT"

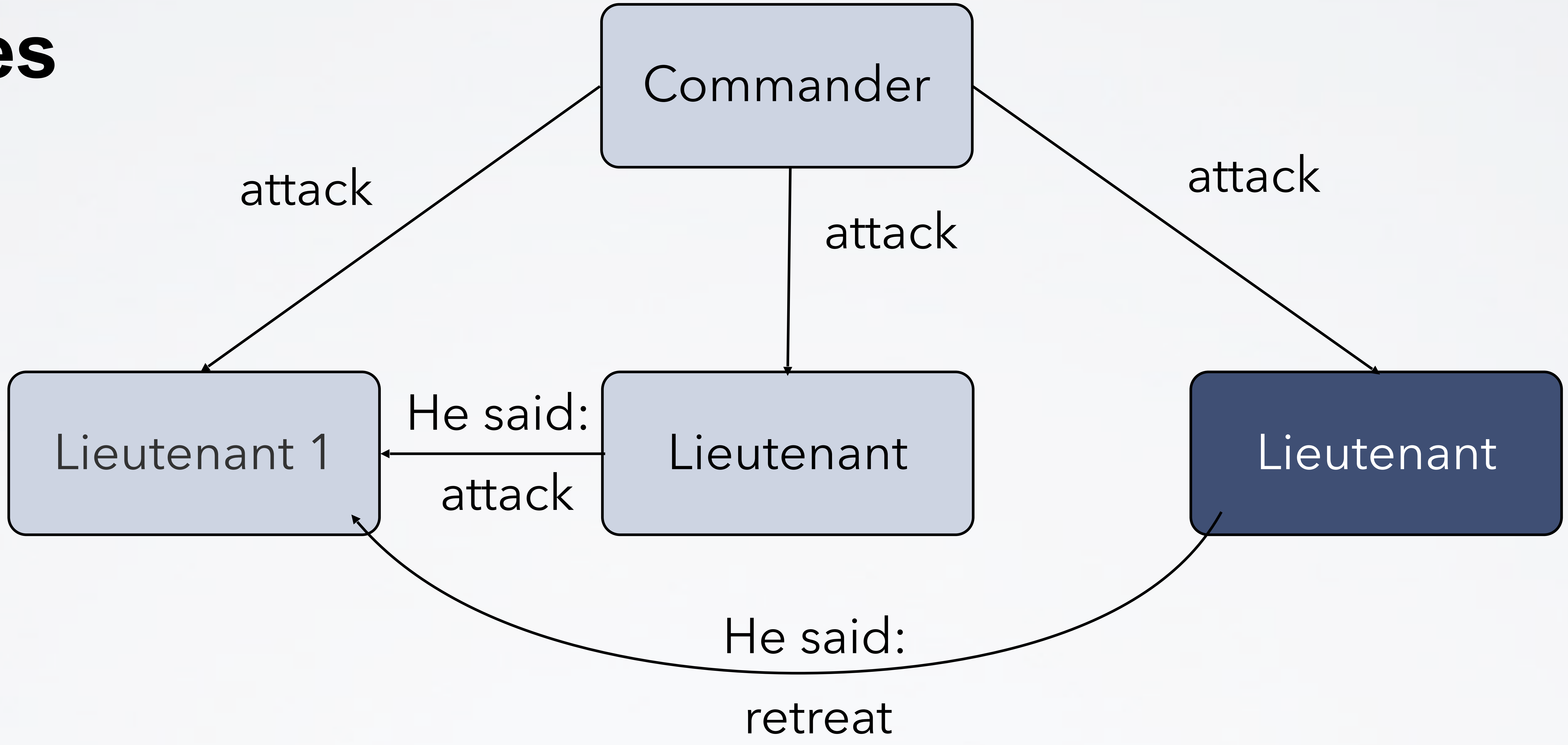


3 Processes: 1 traitor, 2 loyals

=> 3 processes not sufficient to tolerate 1 traitor

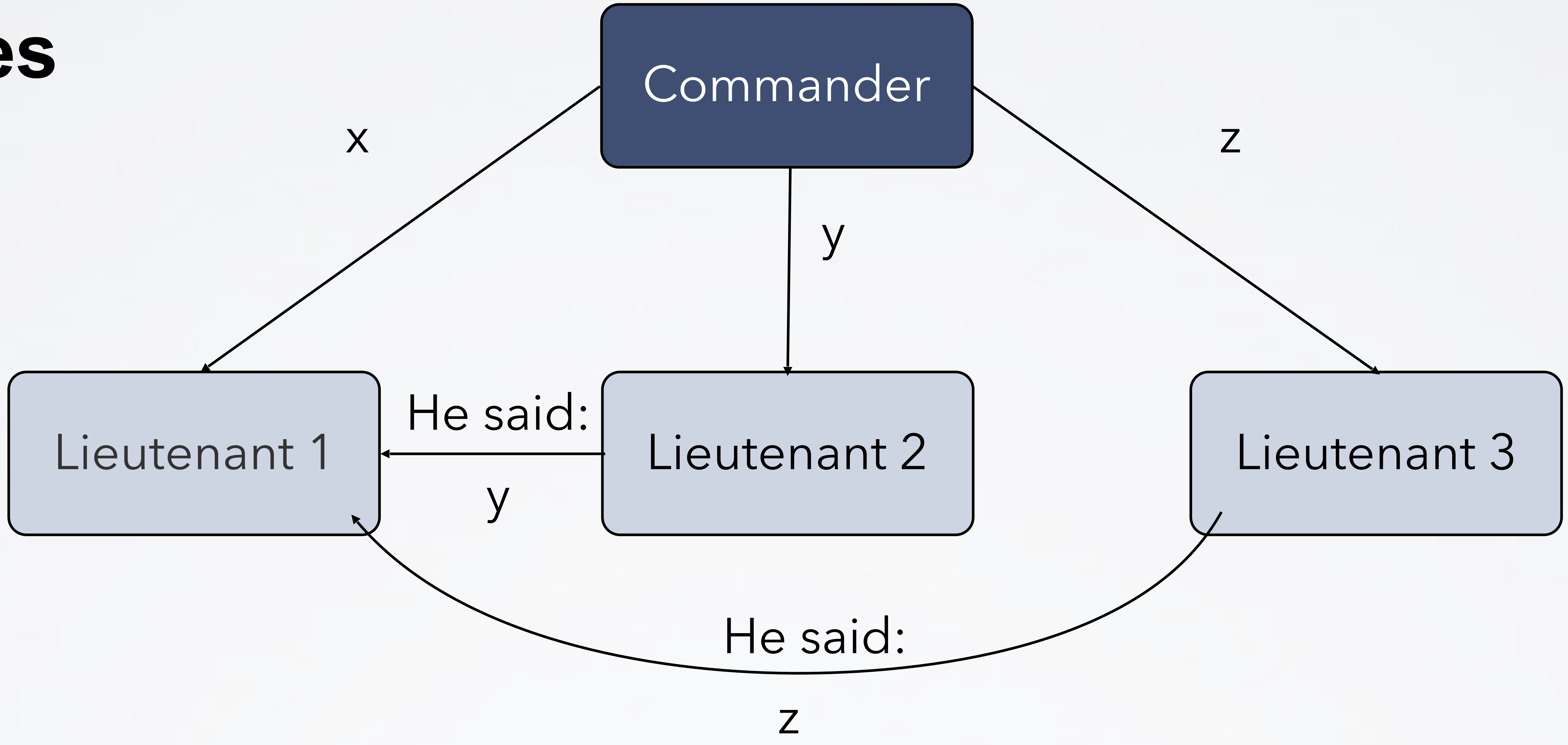
Q2/MODEL 2: “BYZANTINE AGREEMENT”

4 Processes



Q2/MODEL 2: "BYZANTINE AGREEMENT"

4 Processes



all lieutenant receive x,y,z => can decide

General result: 3 f + 1 processes needed to tolerate f traitors

NEXT WEEK

- Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

THE QUESTION

Q3: Is there an algorithm to determine for a system with a given setting of access control permissions, whether or not a Subject A can obtain a right on Object B?

Given a System of Entities (“Objects”) acting as Subjects and/or Objects

- with clearly-defined limited access rights among themselves
- can we achieve clearly-defined Security Objectives ?

TOPICS OF LECTURE

- Definition and Example of “higher-Level” Security Policies (Security Policy Models) (Bell La Padula, Chinese Wall)
- Mechanisms to express/set clearly-defined access rights: Access Control Matrix, ACL, and Capabilities
- Q3 “formalized” in 2 Models: “ACM-based” & “Take Grant”
- Decidable ?
- No proofs (in 2018)

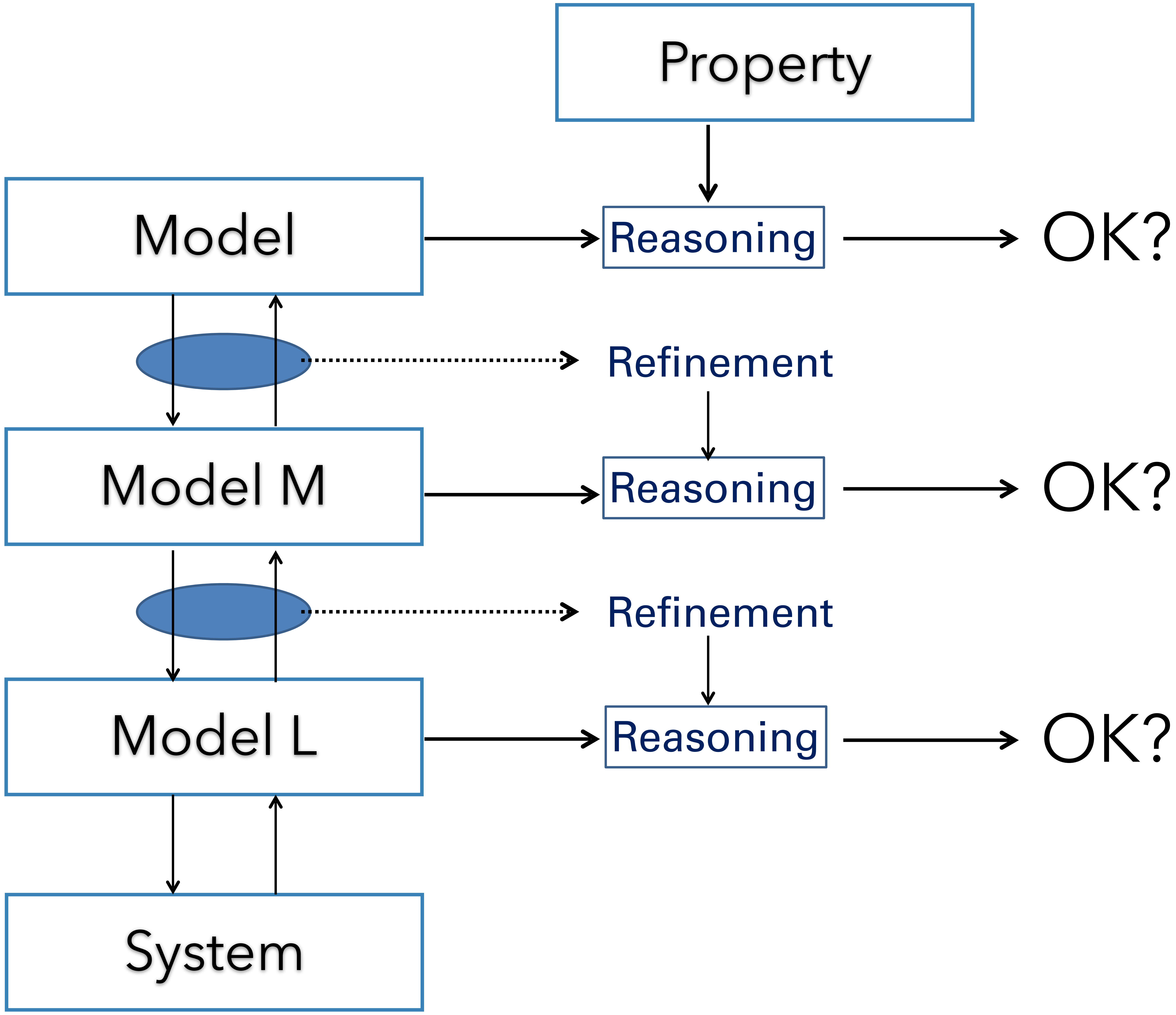
THE GENERAL APPROACH

"Reasoning":

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

"Refinement":

- Abstraction
- Implementation



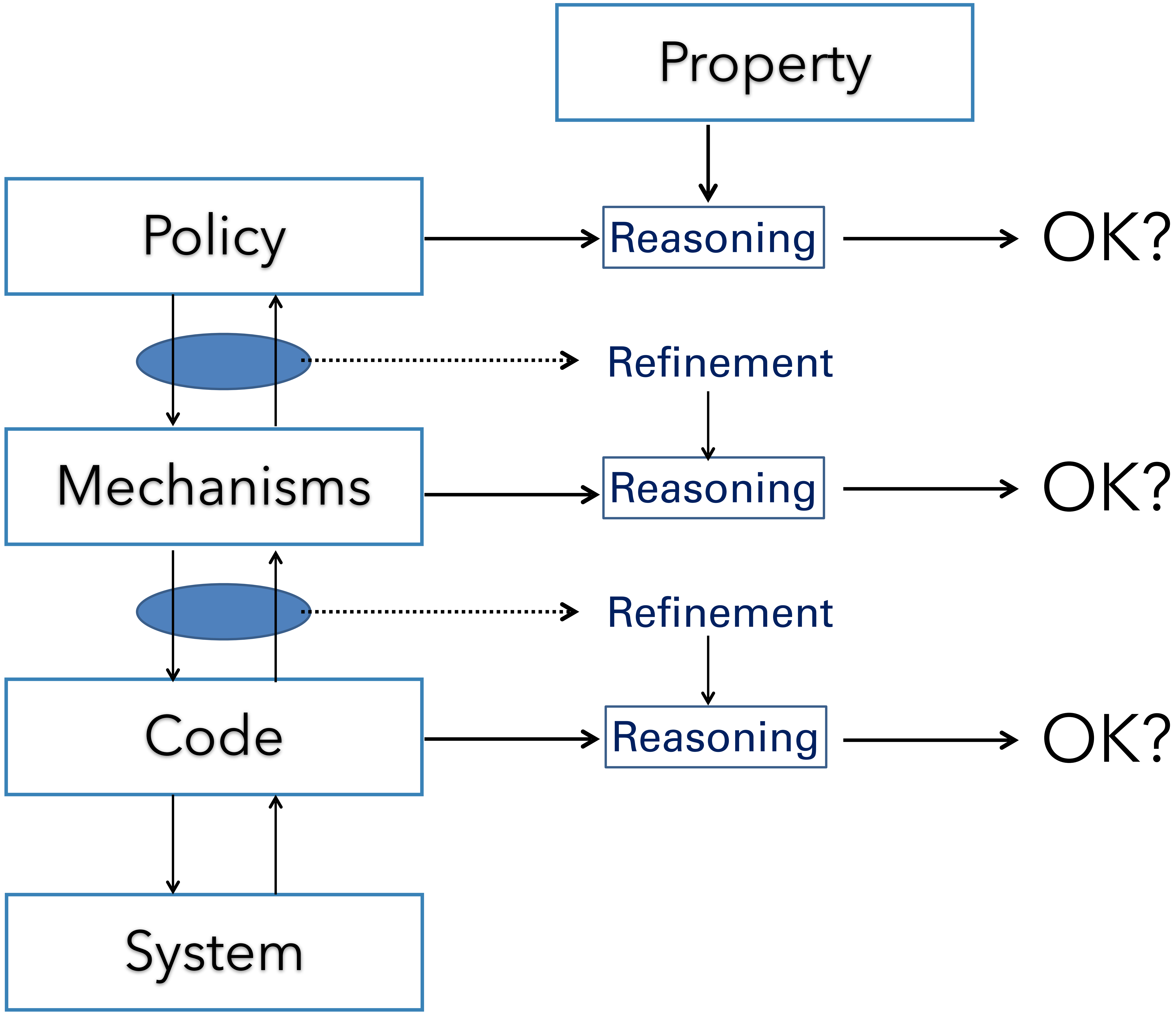
SECURITY MODELS

“Reasoning”:

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics ...
- “Common Criteria Assurance”

“Refinement”:

- Abstraction
- Implementation



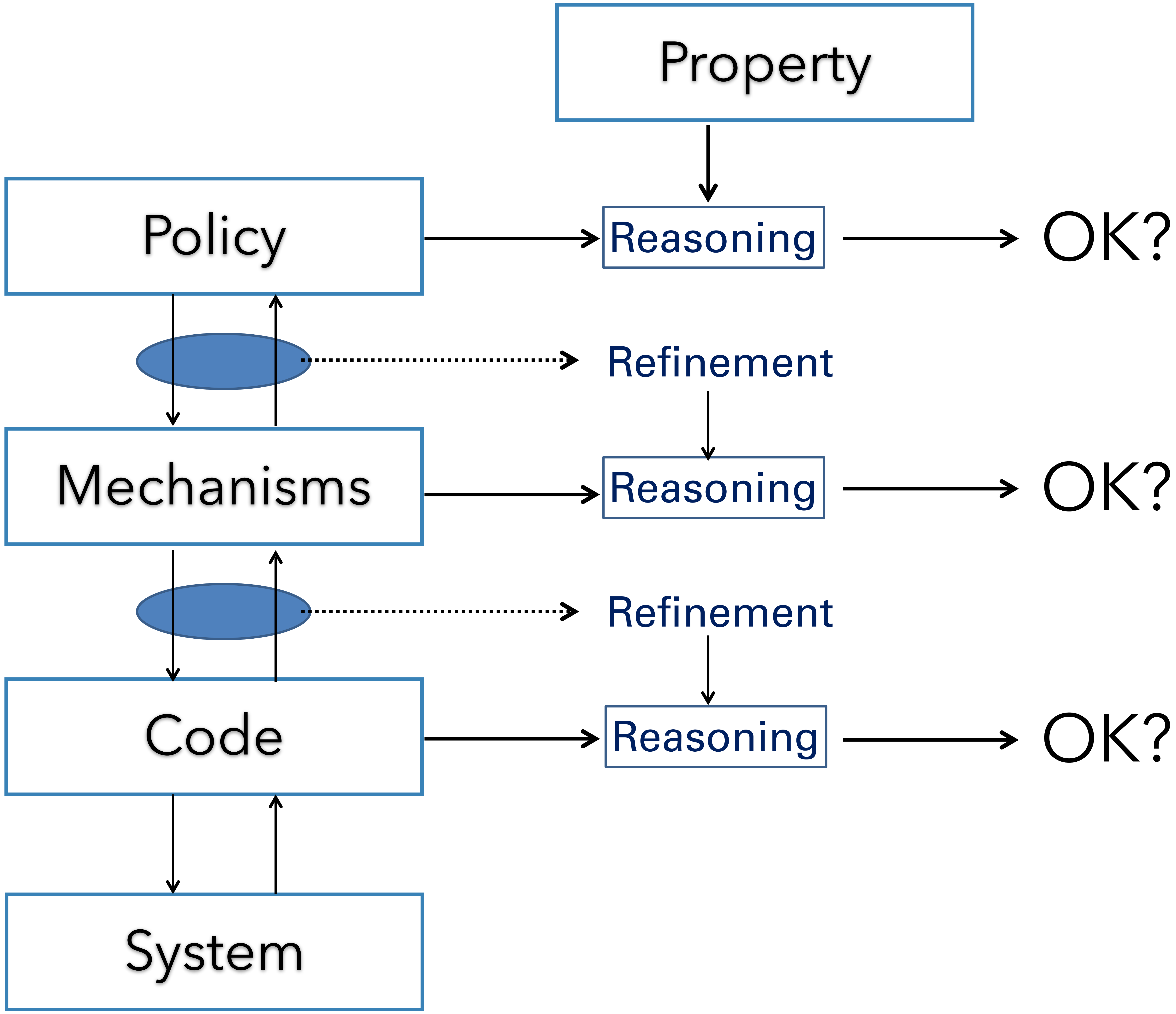
IN THIS LECTURE

Definiton: Policy

Examples:

Higher-Level Policies (very short):

- Bell La Padula
- Chinese Wall



IN THIS LECTURE

Operating Sys. Mechanisms:

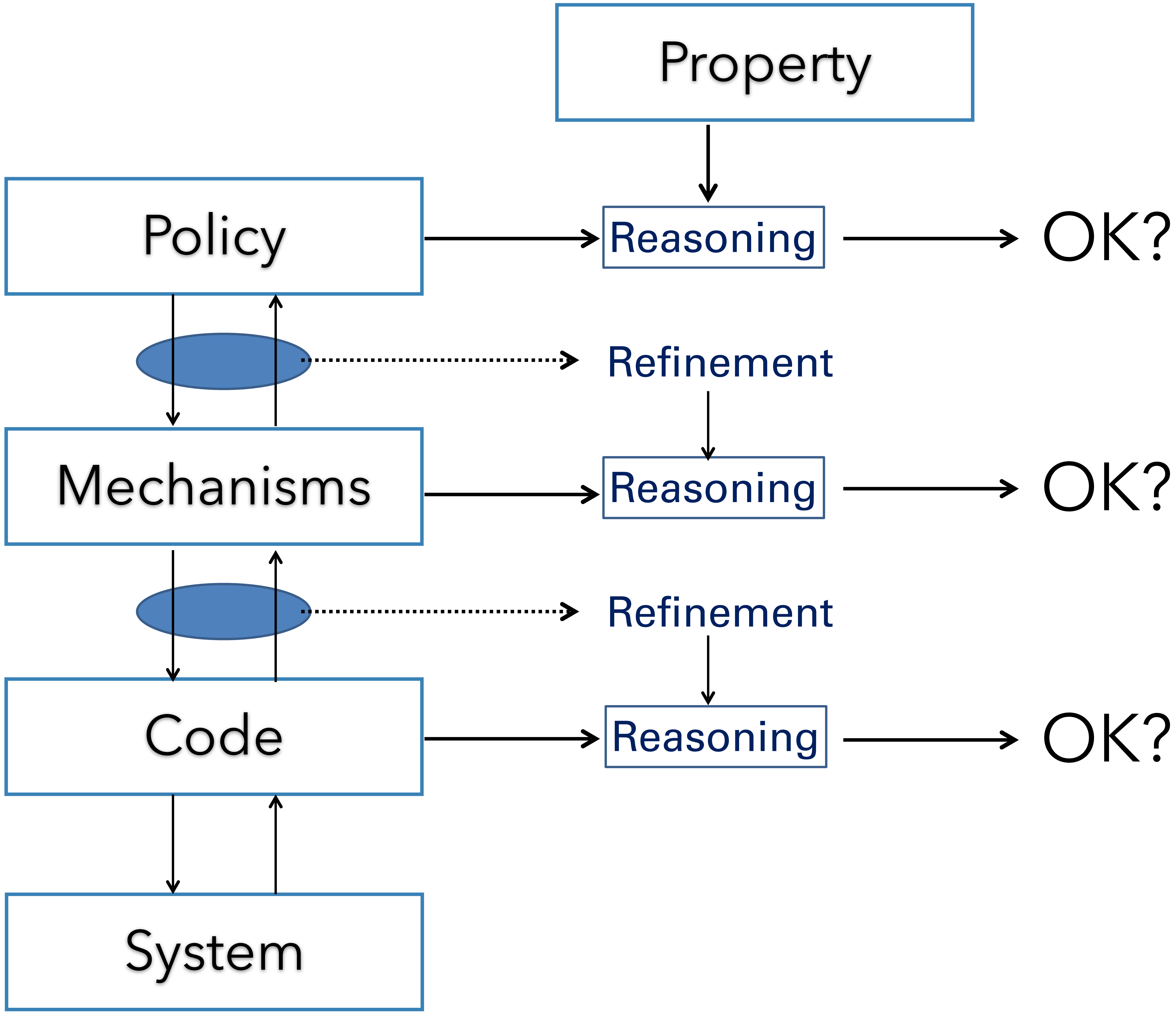
- Access Control List
- Capabilities

...

Explain Q3 and formalize per model!

Models:

- *based on Access Control Matrix*



Security Policy

A security policy P is a statement that partitions the states S of a system into a set of authorized (or secure) states (e.g., $\Sigma sec := \{ \sigma \in \Sigma \mid P(\sigma) \}$) and a set of unauthorized (or non-secure) states.

Secure System

A secure system is a system that starts in an authorized state and that cannot enter an unauthorized state (i.e., $\Sigma reachable \subseteq \Sigma sec$)

Reference: Matt Bishop: Computer Security Art and Science

ref MB: page 95

<u>Definitions:</u>

Information or data I is **confidential**

- with respect to a set of entities X if no member of X can obtain information about I.

Information I or data is **integer** if (2 definitions in text books)

- (1) it is current, correct and complete
- (2) it is either is current, correct, and complete or it is

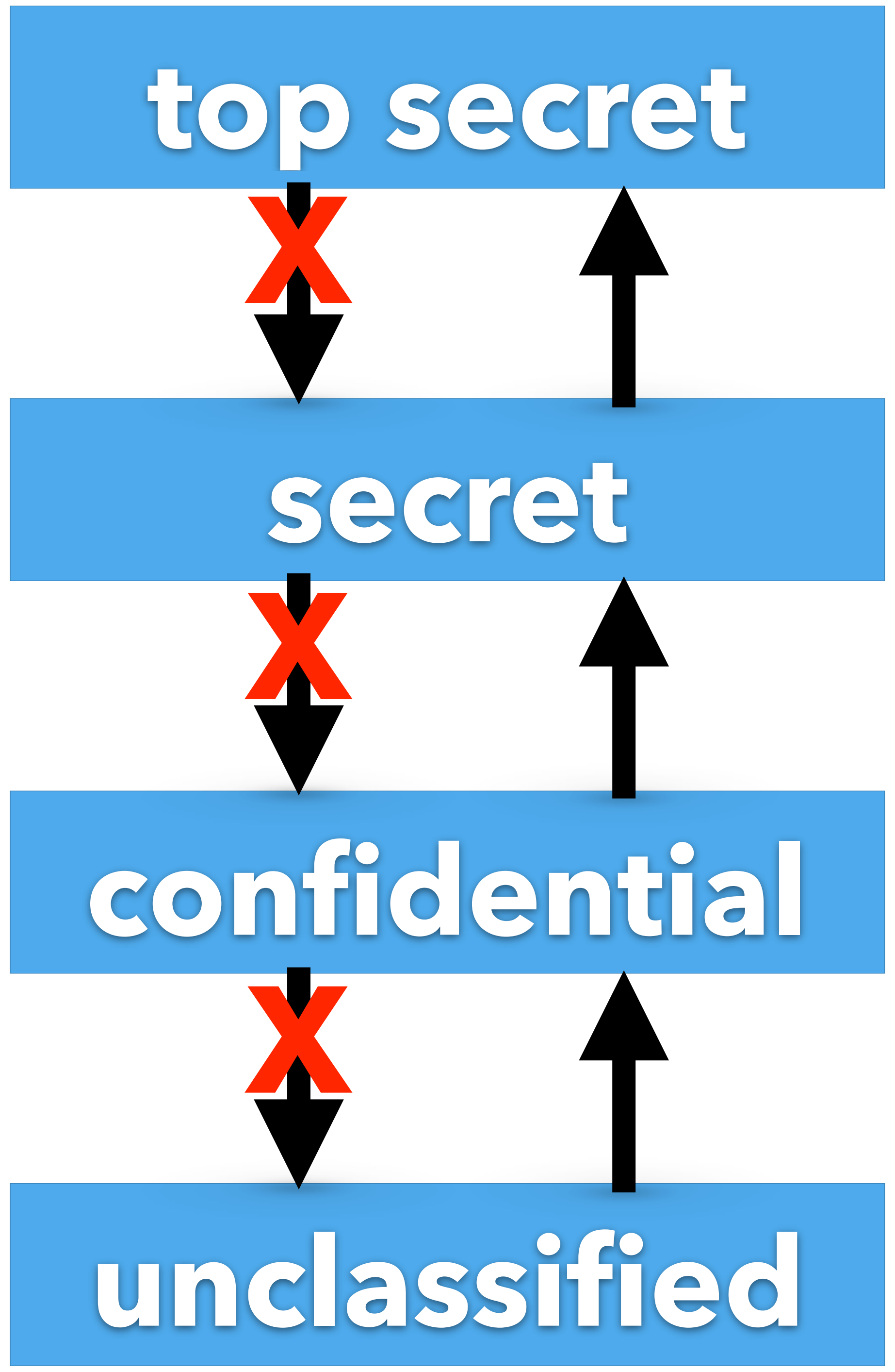
INFORMAL BELL LAPADULA

Model for Confidentiality

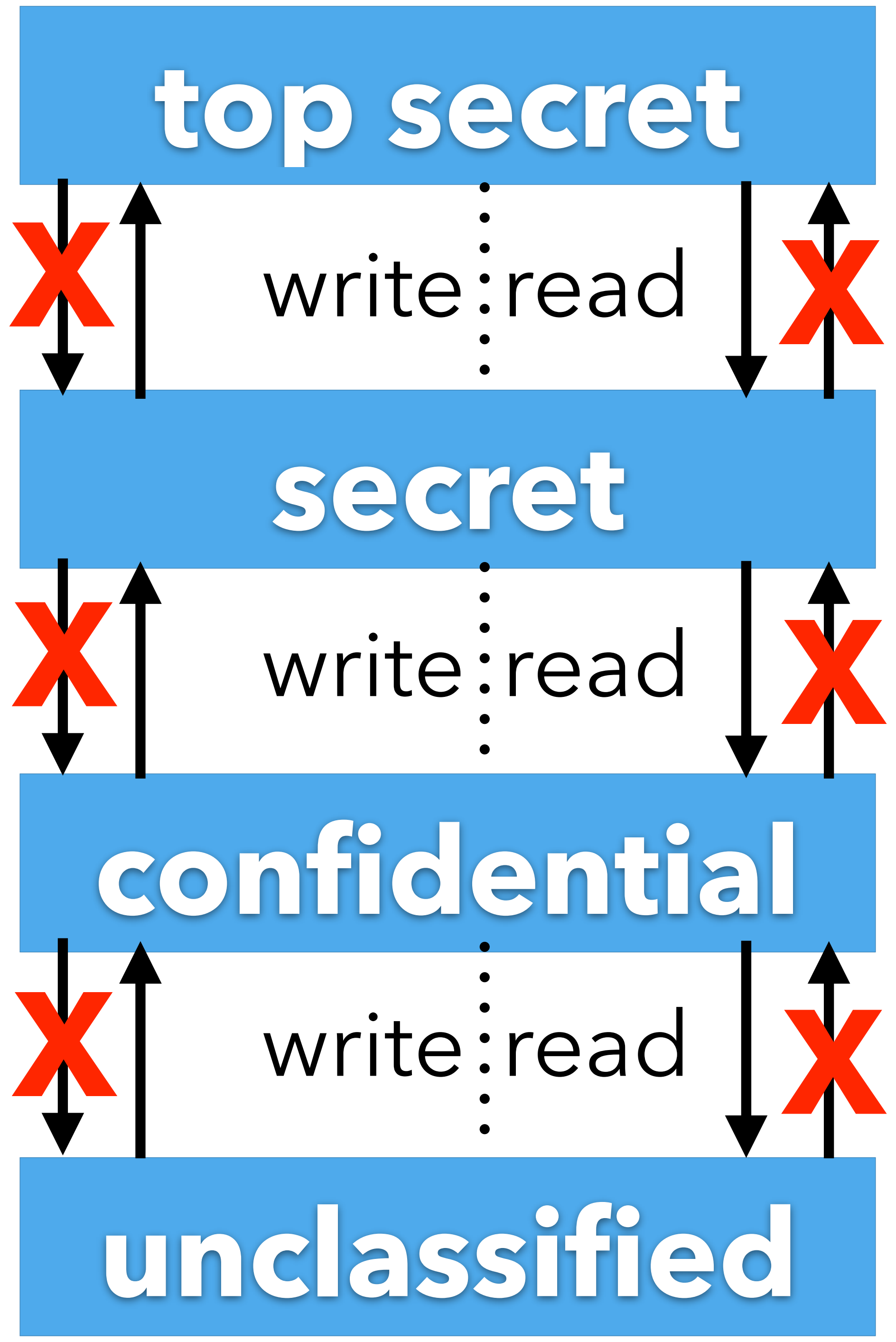
Secrecy Levels:

- Classification (documents)
- Clearance (persons)
- The higher the level the more sensitive the data
- totally ordered

information



operations



EXAMPLES BLP(TANENBAUM)

- categories: NATO, Nuclear
  levels/clearance: top secret, secret, confidential, unclassified
- document: Nato, secret
- person clearance: read
  secret, Nato -> allowed
  secret, Nuclear -> not allowed
  confidential, Nato -> not allowed

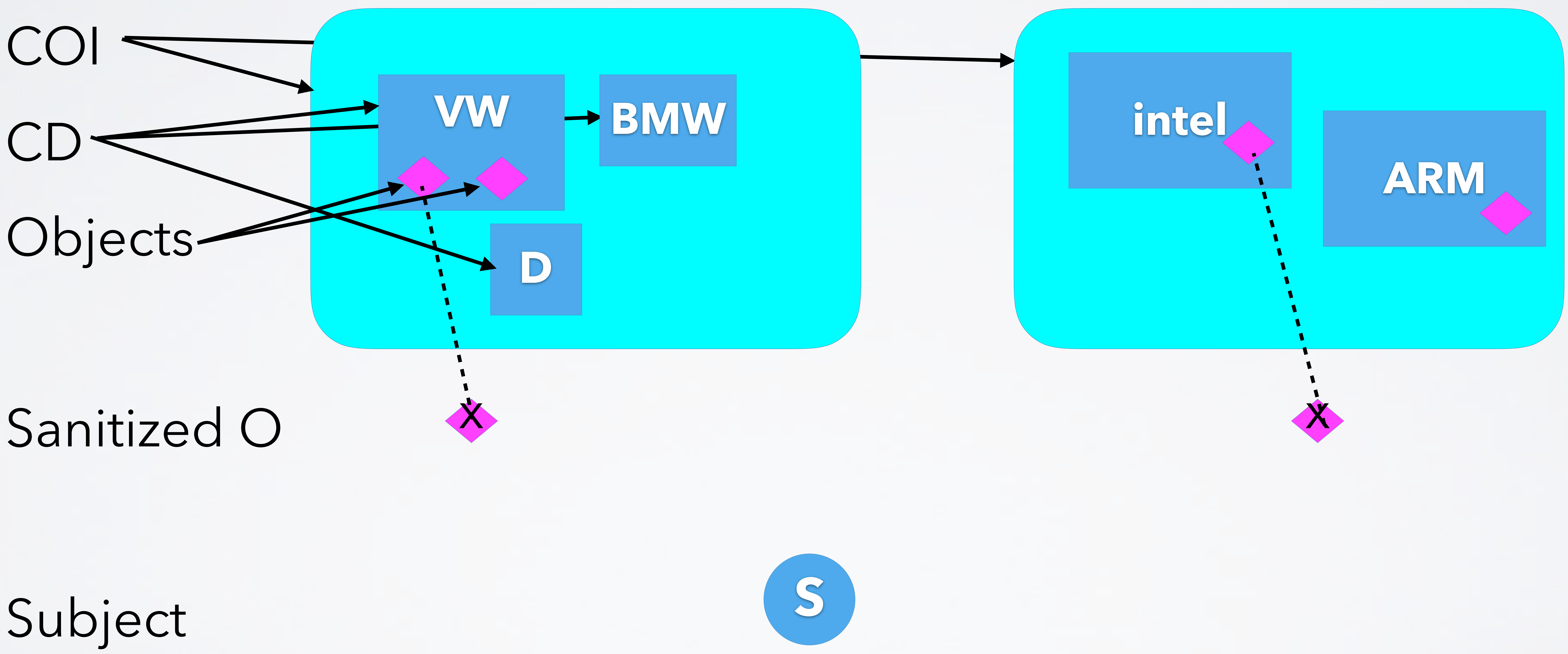
CHINESE WALL POLICY

Confidentiality & Integrity

- Subjects
- Objects: pieces of information of a company
- CD: Company Data Sets
  objects related to single company
- COI: Conflict of Interest class
  data sets of competing companies
- Sanitized Objects
  version of object that does not contain critical information

Ref MB: Chapter 7.1

CHINESE WALL, EXAMPLE



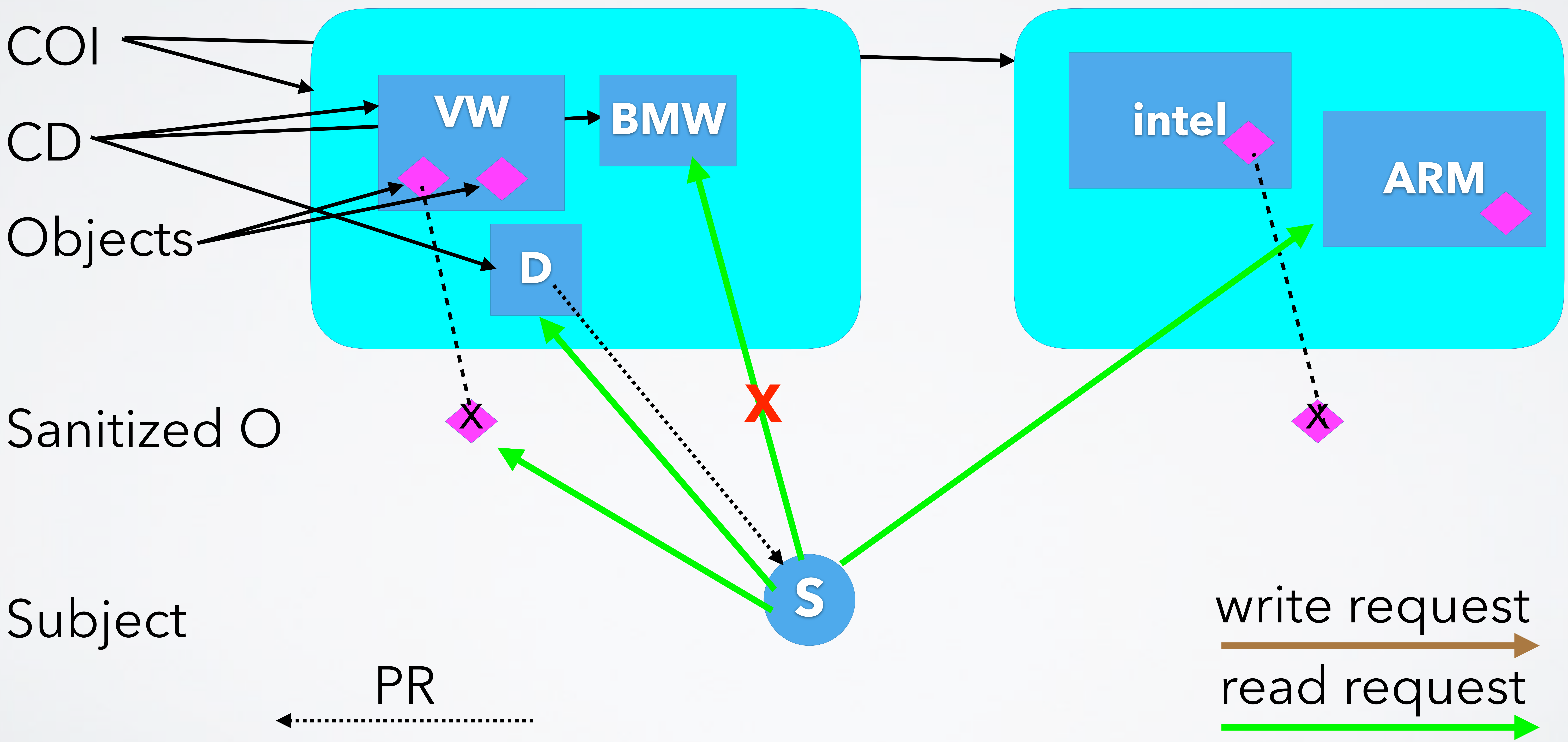
CHINESE WALL, RULES

PR(S): set of Objects previously read by S

S can read O, if any of the following holds

- first-time read
- $\forall$ O, O' $\in$ PR(S) => COI(O) $\neq$ COI(O')
- O is a sanitized Object

CHINESE WALL, EXAMPLE



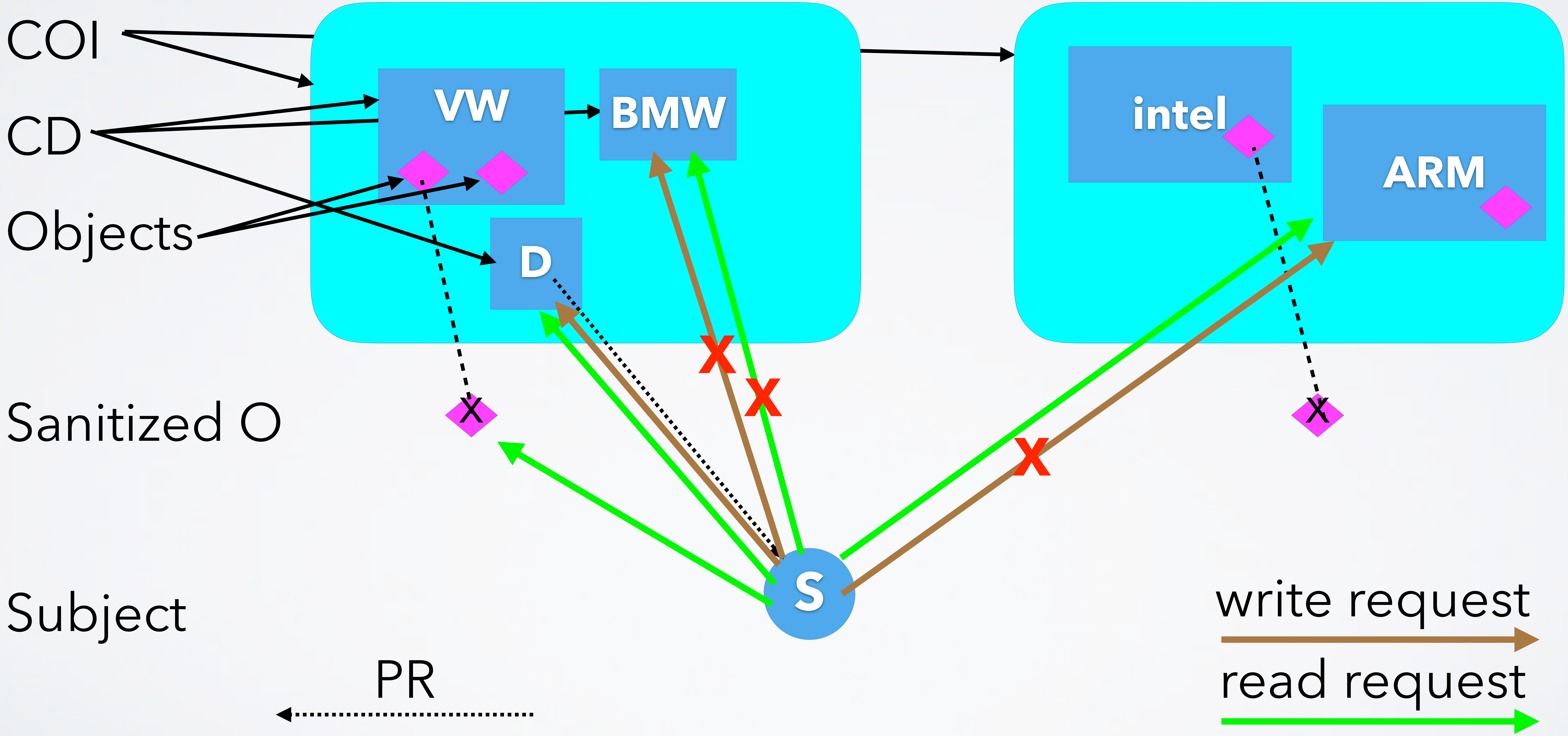
CHINESE WALL, RULES

PR(S): set of Objects read by S

S can write O, if

- "S can read O"
- ∀ unsanitized O', "S can read O'" => CD(O) = CD(O')

CHINESE WALL, EXAMPLE



MECHANISMS

Operating Sys. Mechanisms:

- Access Control List
- Capabilities

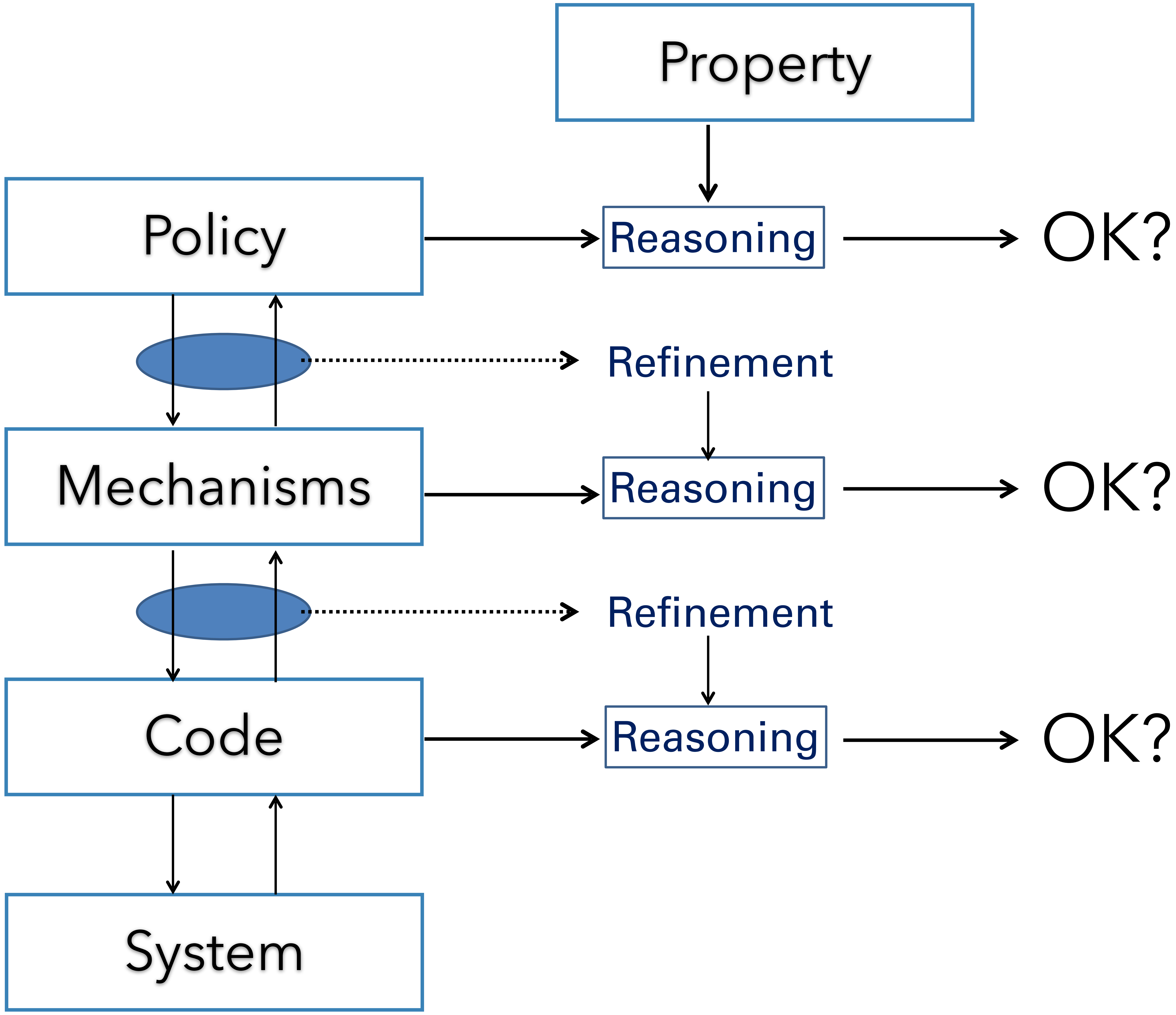
...

Explain Q3 and

formalize per model!

Models:

- *based on Access Control Matrix*
- *"take grant" model*



MECHANISMS: ACCESS CONTROL MATRIX

Subjects: S

Objects: O

Entities: E = S ∪ O

Rights: {read, write, own,...}

Matrix: S x E x R

	O1	O2	S1	S2	S3
S1	r,w,own	r,w	r,w,own	–-	r,w
S2	r,w	r,w,own	–	r,w,own	r
S3	r,w	r	w	–-	r,w,own

Simple AC**M** Operations:

create subject / object

destroy subject / object

enter / delete R into cell (s,o)

ref MB: chapter 2.2

OS MECHANISMS: ACL & CAPS

ACM

- Access Control List (ACL)

	O1	O2	S1	S2	S3
S1	r,w,own	r,w	r,w,own	–-	r,w
S2	r,w	r,w,own	–	r,w,own	r
S3	r,w	r	w	–-	r,w,own

- Capabilities

	O1	O2	S1	S2	S3
S1	r,w,own	r,w	r,w,own	–-	r,w
S2	r,w	r,w,own	–	r,w,own	r
S3	r,w	r	w	–-	r,w,own

ref MB: chapter 2.2

Q3/MODEL 1: ACL & "LEAKAGE"

- Define Protection Mechanisms of an Operating System in terms of primitive ACM operations
- only the defined mechanism provided by the OS can used

ref MB: chapter 2.2

Q3/MODEL 1: ACL & "LEAKAGE"

- "Leakage":
  an access right is placed into S/O that has not been there before
  it does not matter whether or not that is allowed
- Is leakage decidable ?

ref MB: chapter 3

Q3/MODEL 1: ACL & "LEAKAGE"

Examples for OS-Mechanisms defined by ACM-Operations:

UNIX create file (S1,F)

create object

enter own into A(S1,F)

enter read into A(S1,F)

enter write into A(S1,F)

	O1	O2	S1	S2	F
S1	r,w,own	r,w	r,w,own	–-	r,w, own
S2	r,w	r,w,own	–	r,w,own	–
S3	r,w	r	w	–-	–

ref MB: chapter 2.2

Q3/MODEL 1: ACL & "LEAKAGE"

Examples for OS-Mechanisms defined by ACM-Operations:

UNIX chmod -w (S2,F)

if own ∊ A(caller,F)

then delete w in A(S2,F)

	O1	O2	S1	S2	F
S1	r,w,own	r,w	r,w,own	–-	r,w, own
S2	r,w	r,w,own	–	r,w,own	r,-
S3	r,w	r	w	–-	–

Q3:

Given an OS with a ACM-based description of protection mechanisms is "Leakage" decidable for any R in A(x,y) ?

ref MB: chapter 2.2

Q3/MODEL 1: DECIDABILITY OF LEAKAGE

Decidable

- no subjects/objects can be created

or
- only **one** primitive ACM operation per OS-Mechanism

by exhaustive search !

Q3 in general:

- undecidable (proof: reduction to Turing machine)

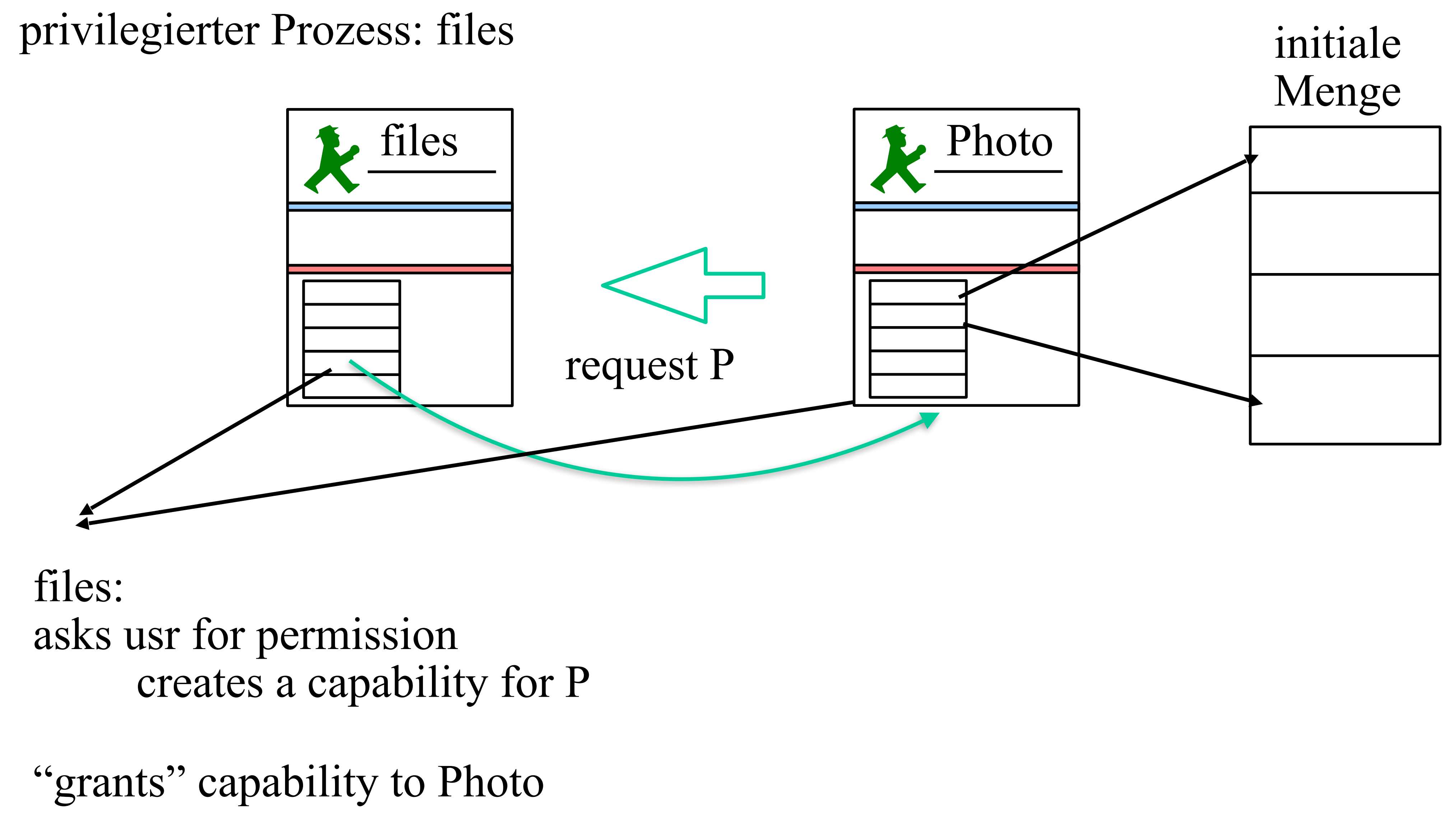
ref MB: chapter 3

Q3/MODEL 2: "TAKE GRANT"

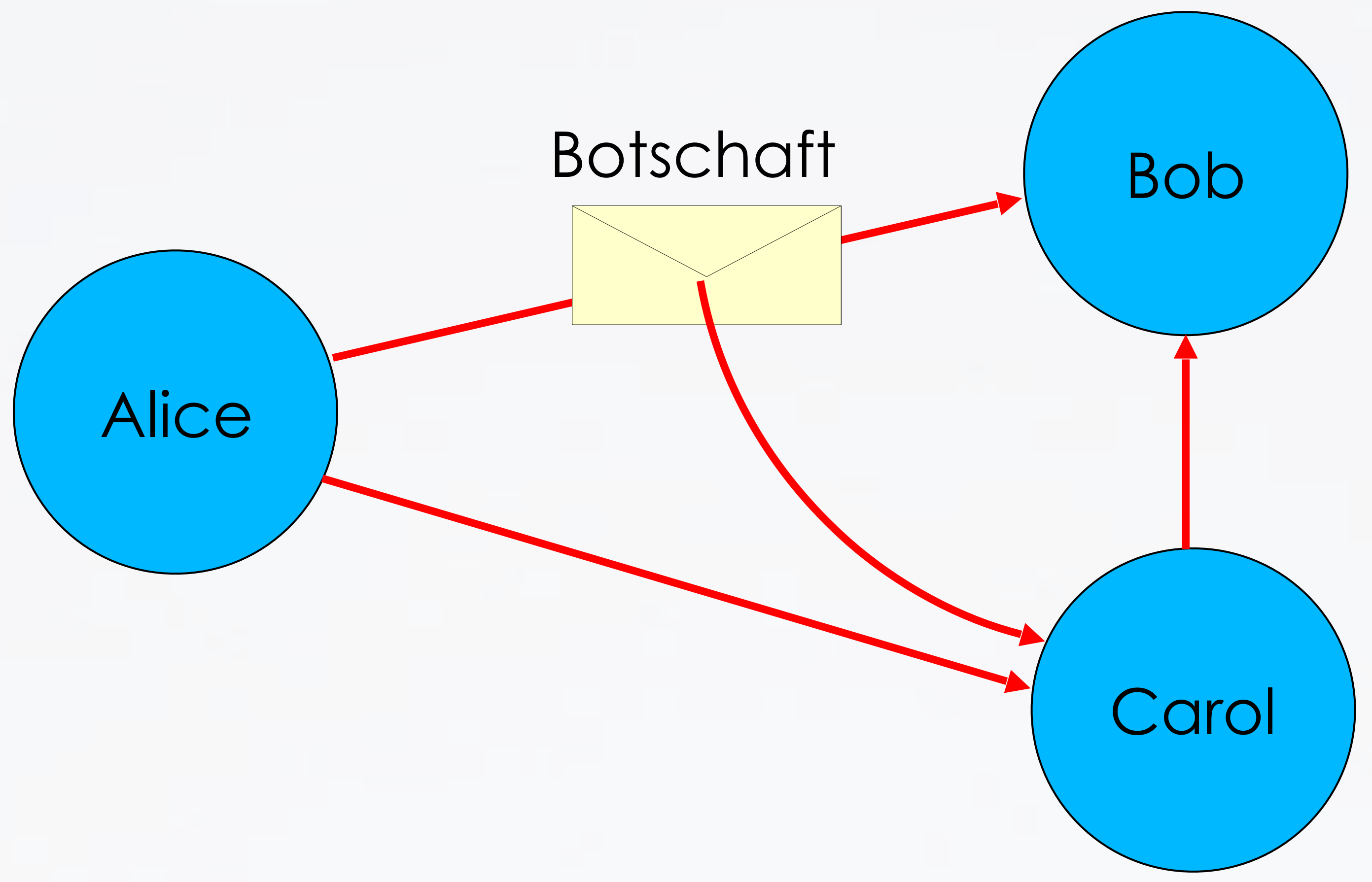
"Capabilities"

an intuitive example

- files: a privileged process
- Photo: an untrusted process
- Photo brings a small initial set of "capabilities" on installation
- needs permission to edit a specific photo P



L4 CAPABILITIES



Q3/MODEL 2: “TAKE GRANT”

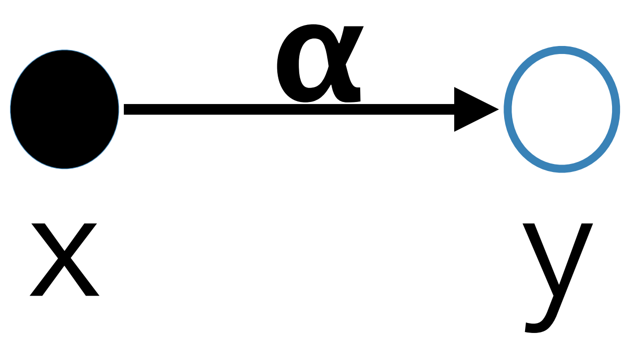
<u>Directed Graph:</u>

Subjects: ●

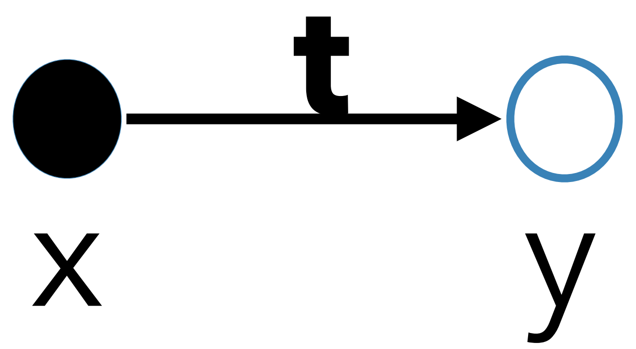
Objects: ○

Either S or O: ⊗

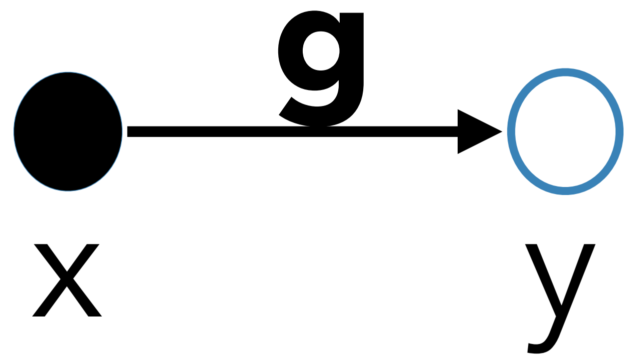
x has capability
with set of rights **α** on y:



t <u>take right</u>
x has cap with set of rights
τ that includes t



g <u>grant right</u>
x has cap with set of rights
γ that includes **g**

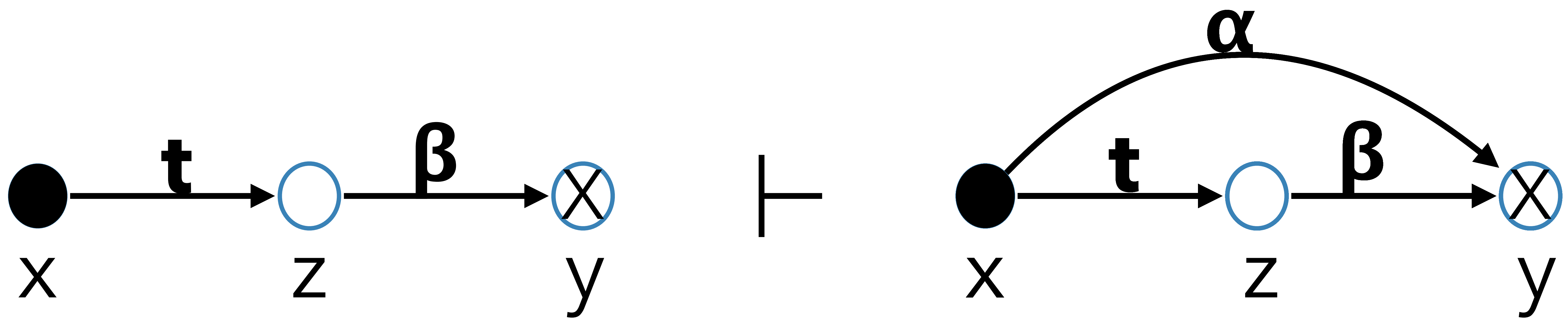


TECHNISCHE UNIVERSITÄT DRESDEN

Q3/ 2: TAKE GRANT RULES

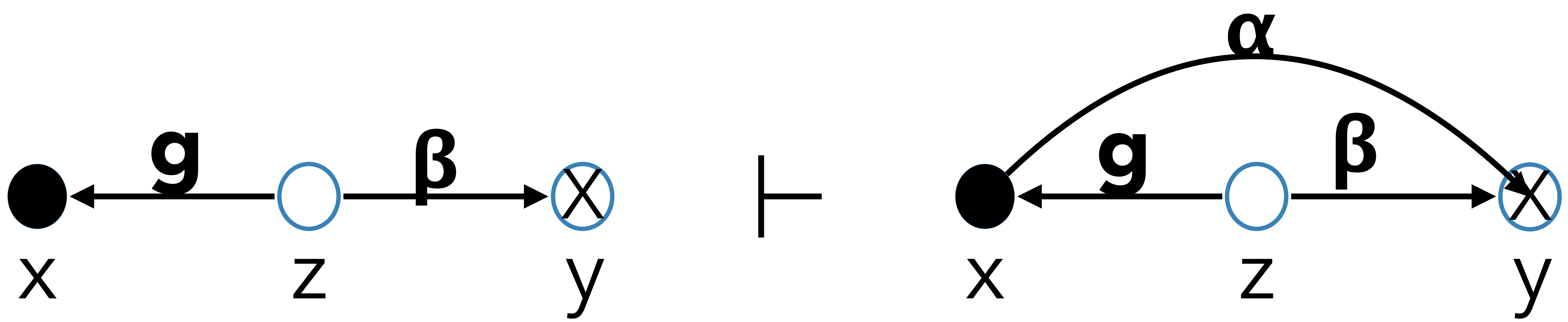
<u>Rules:</u>

take rule (**α**⊆**β**)



a takes (**α** to y) from z

grant rule (**α**⊆**β**)



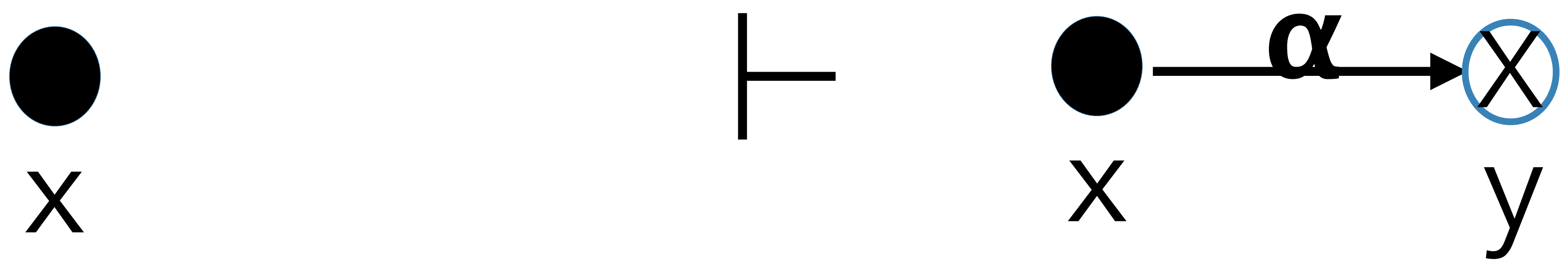
z grants (**α** to y) to x

ref MB: chapter 3.3

Q3/ 2: TAKE GRANT RULES

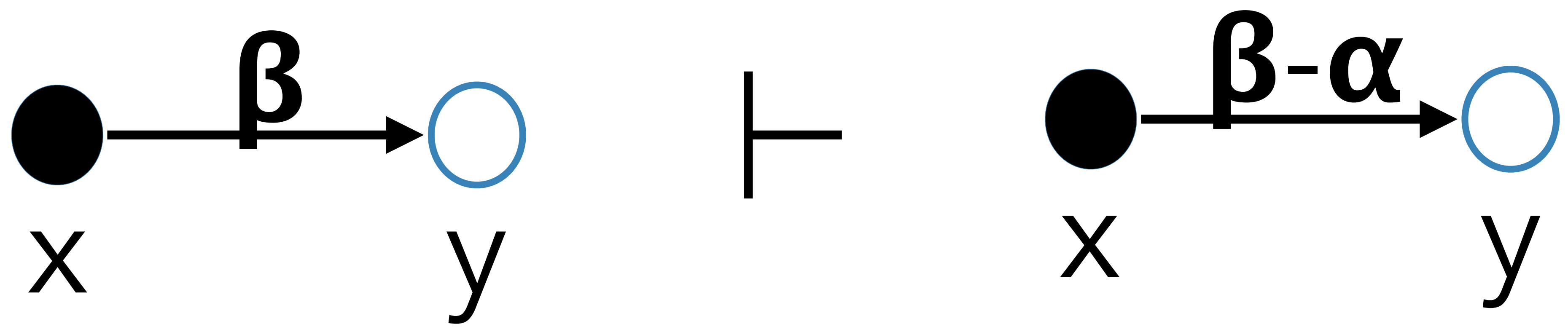
Rules:

create rule



x create (**α** to new vertex) y

remove rule



x removes (**α** to) y

ref MB: chapter 3.3

Q3/M2: FORMALIZED

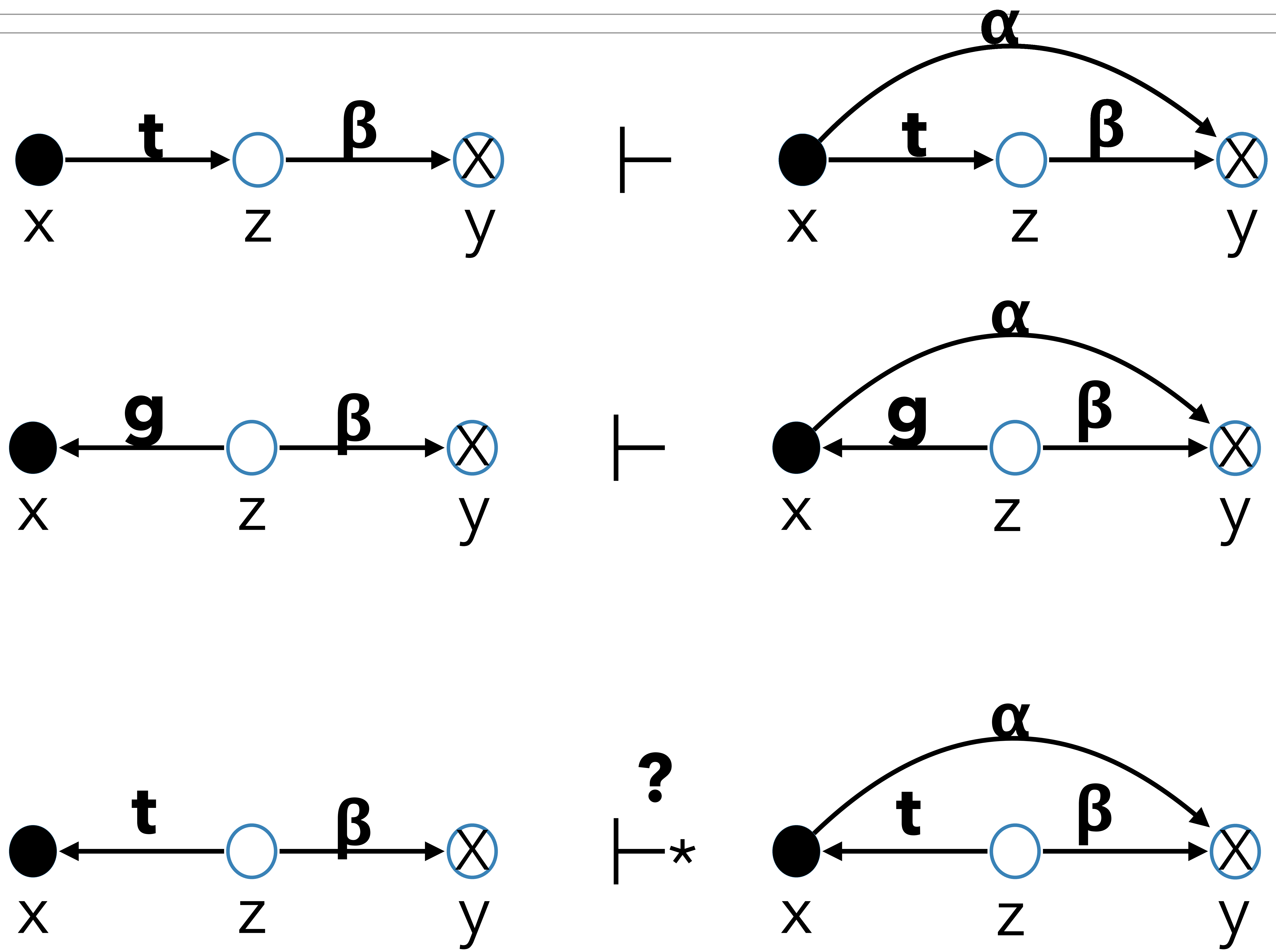
CanShare(α, x, y, G_0):

there exists a sequence of G_0 … G_n with $G_0 \vdash^* G_n$

and there is an edge in G_n: $x \xrightarrow{\alpha} y$

ref MB: chapter 3.3

Q3/ 2: CAREFUL: LEMMA



take rule (α⊆β)

a takes (α to y) from z

grant rule (α⊆β)

z grants (α to y) to x

ref MB: chapter 3.3

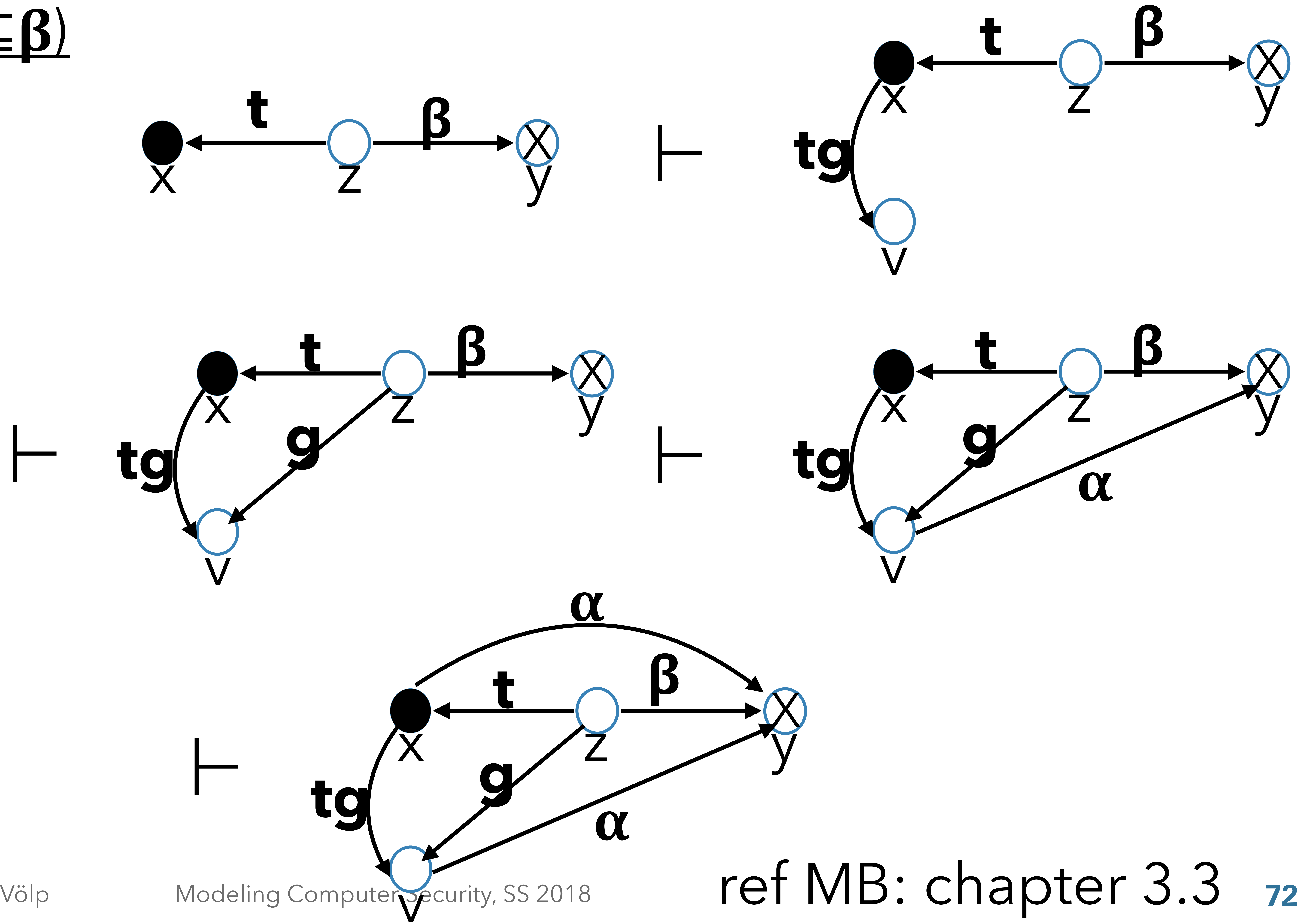
Q3/ 2: CAREFUL: LEMMA

<u>(α⊆β)</u>

create rule

z takes (g to v) from x

z grants (**α** to y) to v



ref MB: chapter 3.3

Q3/M2: FORMALIZED

<u>CanShare(**α**, x, y, G_0):</u>

there exists a sequence of $G_0 \ldots G_n$ with $G_0 \vdash^* G_n$

and there is an edge: $x \xrightarrow{\boldsymbol{\alpha}} y$

CanShare decidable in linear time !

ref MB: chapter 3.3

TAKE AWAY

- three questions, 2 models per question, different answers !!!
- modeling is powerful
- need to look extremely carefully into understanding models !!!

REFERENCES

- Q1/M1:

  Pfitzmann A., Härtig H. (1982) Grenzwerte der Zuverlässigkeit von Parallel-Serien-Systemen. In: Nett E., Schwärtzel H. (eds) Fehlertolerierende Rechnersysteme. Informatik-Fachberichte, vol 54. Springer, Berlin, Heidelberg (in German only)

  Q1/M2:

  John v. Neuman, PROBABILISTIC LOGICS AND THE SYNTHESIS OF RELIABLE. ORGANISMS FROM UNRELIABLE COMPONENTS.

- Q2: most textbooks on distributed systems
- Q3: textbook: Matt Bishop, Computer Security, Art and Science, Addison Wesley 2002