TECHNISCHE UNIVERSITÄT DRESDEN

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

MODELING DISTRIBUTED SYSTEMS

HERMANN HÄRTIG, DISTRIBUTED OPERATING SYSTEMS, SS2022

SYSTEM MODELS IN GENERAL

use models to analyze, prove, predict, ... properties of concrete systems AND
to **establish fundamental insights**

- abstract from details
- concentrate on functionality, properties, ... considered important for a specific system/application/question
- models in engineering disciplines very common, increasingly in CS as well

MODELS IN REAL-TIME SYSTEMS

Purpose

- describe the timing requirements of an application
- describe available resources
- question:
  can the application run on/use these resources such that timing requirements are fulfilled

Model elements:

- periodic tasks, deadlines, worst-case exec time, ...

Hopefully RTS class is offered in future (by my successor)

MODEL EXAMPLES IN GENERAL

Model	Objective/Question
■ Failure Trees	are all failures and their combinations taken into account
■ statics models	does a house fall down (snow, quake) 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

THIS LECTURE'S QUESTIONS

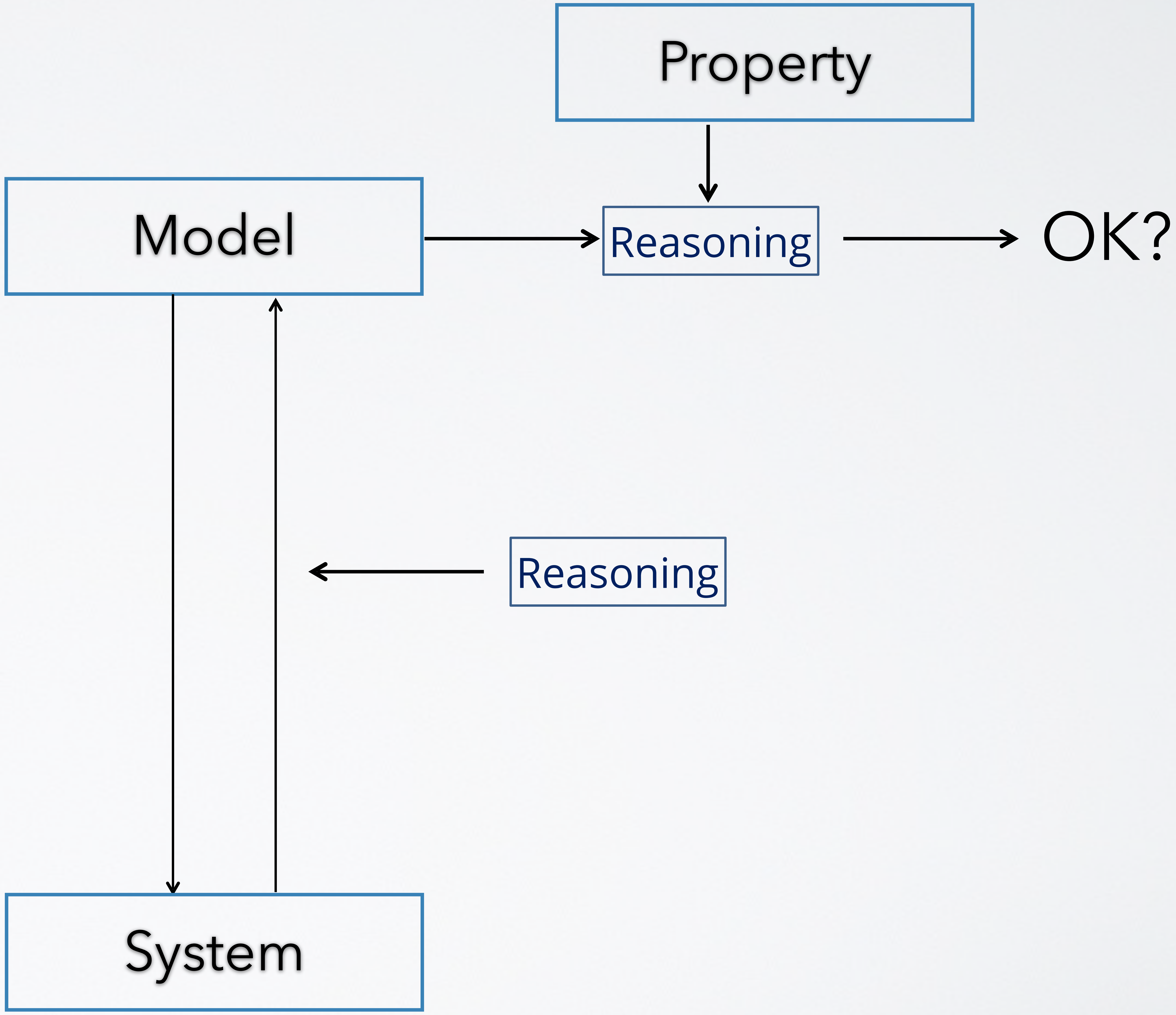
- Q1: Is it possible to build arbitrarily reliable Systems out of unreliable components?
- Q2: Can we achieve consensus in the presence of faults (consensus: all non-faulty components agree on action)?
- Q3: Is there an algorithm to determine for a system with a given setting of access control permissions,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

2 Models per Question !

SYSTEMS MODELS: GENERAL APPROACH

Reasoning:

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

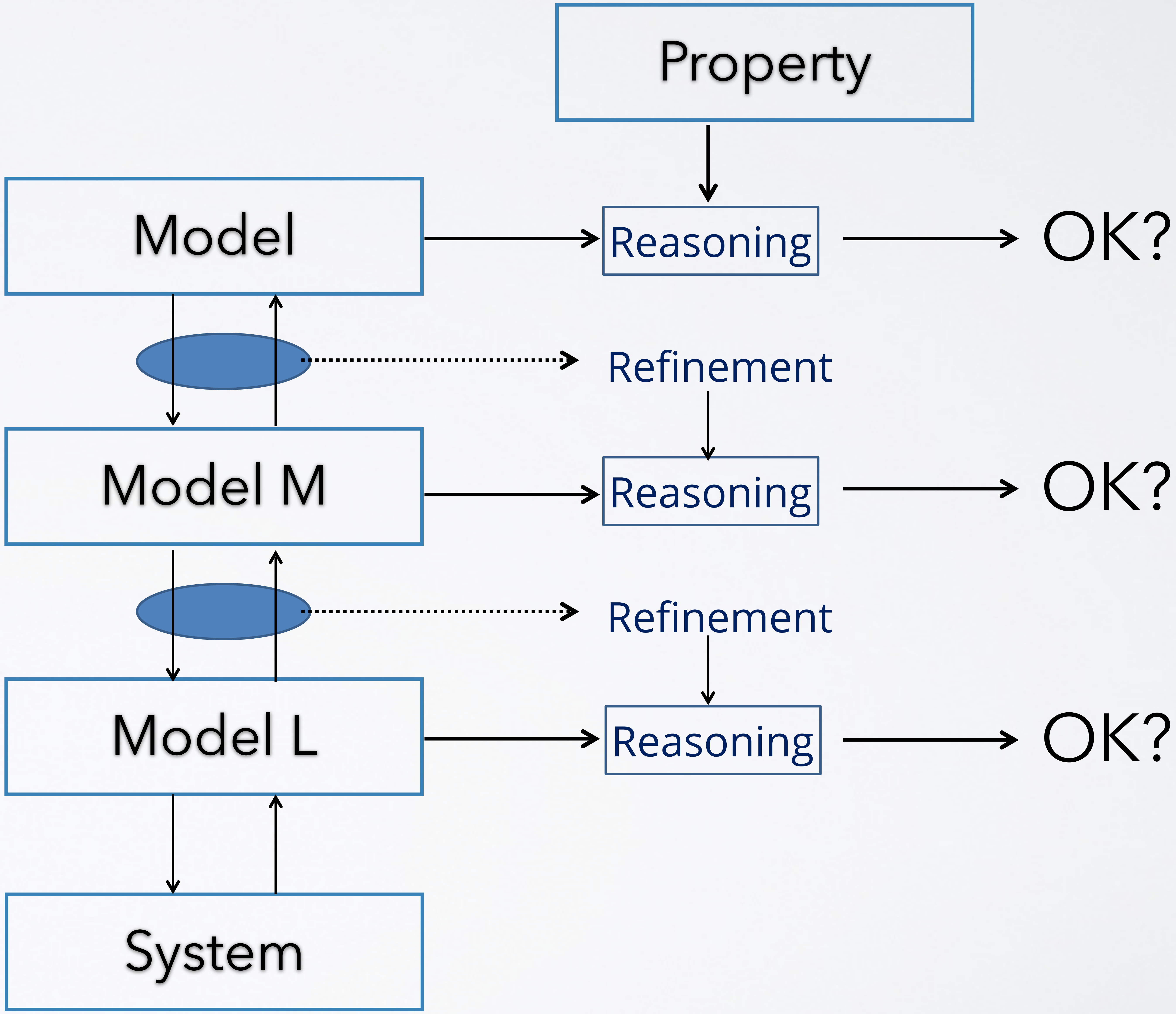


SYSTEMS MODELS: GENERAL APPROACH

Reasoning:

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics …

- "Refinement":
  - Abstraction
  - Implementation
  - Formal Refinement



MODEL EXAMPLES COMPUTER SCIENCE

Model	Objective/Question
▪ Amdahl's Law	Scalability
▪ Turing Machine	Halting problem, Decidability
▪ Logic	Correctness, Precision, …

AMDAHL'S LAW

$$\text{Speedup:} \frac{\text{original execution time}}{\text{enhanced execution time}}$$

- P: section that can be parallelized
- 1-P: serial section
- N: number of CPUs

$$\text{Speedup}(P,N) = \frac{1}{\left(1 - P + \frac{P}{N}\right)}$$

- if N becomes VERY large, speedup approaches: 1/(1-P)

MODELS IN THIS CLASS(DOS)

- Objective of lecture:
  understand the power of models and the need for their careful understanding
- models in detail, but math results by intuition not proofs
- Try to find answers to question Q1 … Q3
  BEFORE viewing the other pieces of the lecture and the full slide set

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 !

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)

Reliability:

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

Availability:

- A: fraction of time a system works

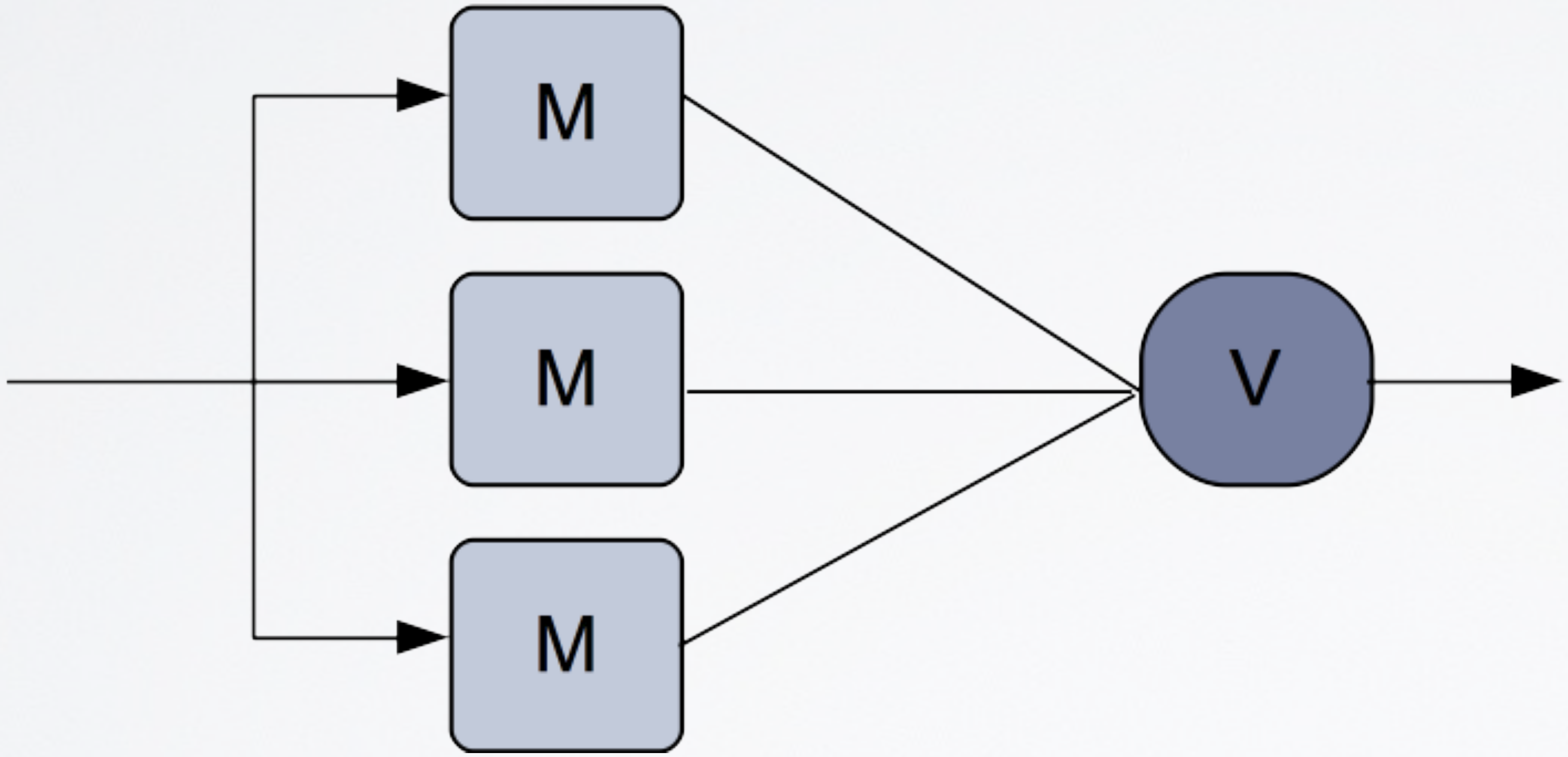
INGREDIENTS OF FT

- Fault detection and confinement
- Recovery
- Repair

- Redundancy
  - Information
  - time
  - structural
  - functional

WELL KNOWN EXAMPLE

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

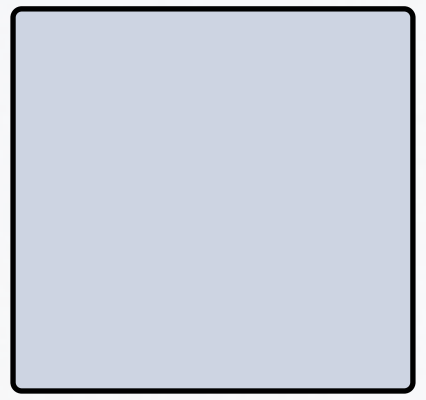


Can we do better
→ distributed solutions?

Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

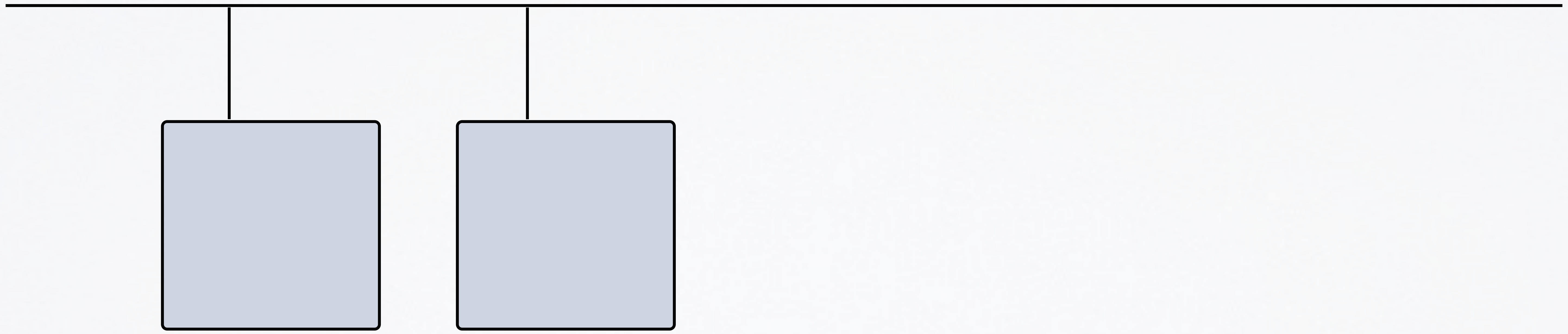
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

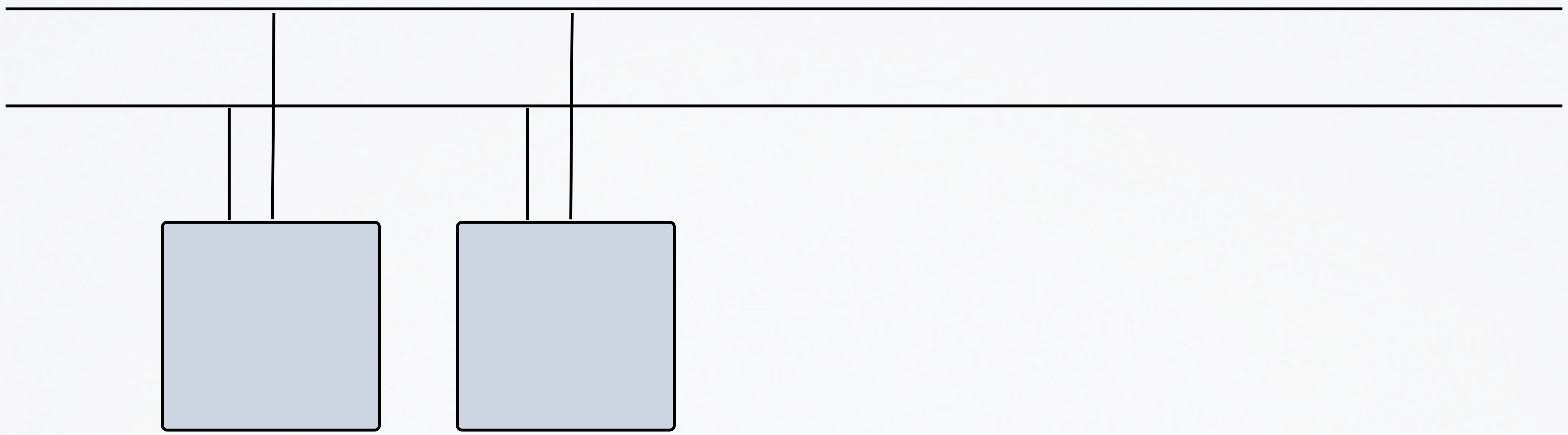
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

Parallel-Serial-Systems

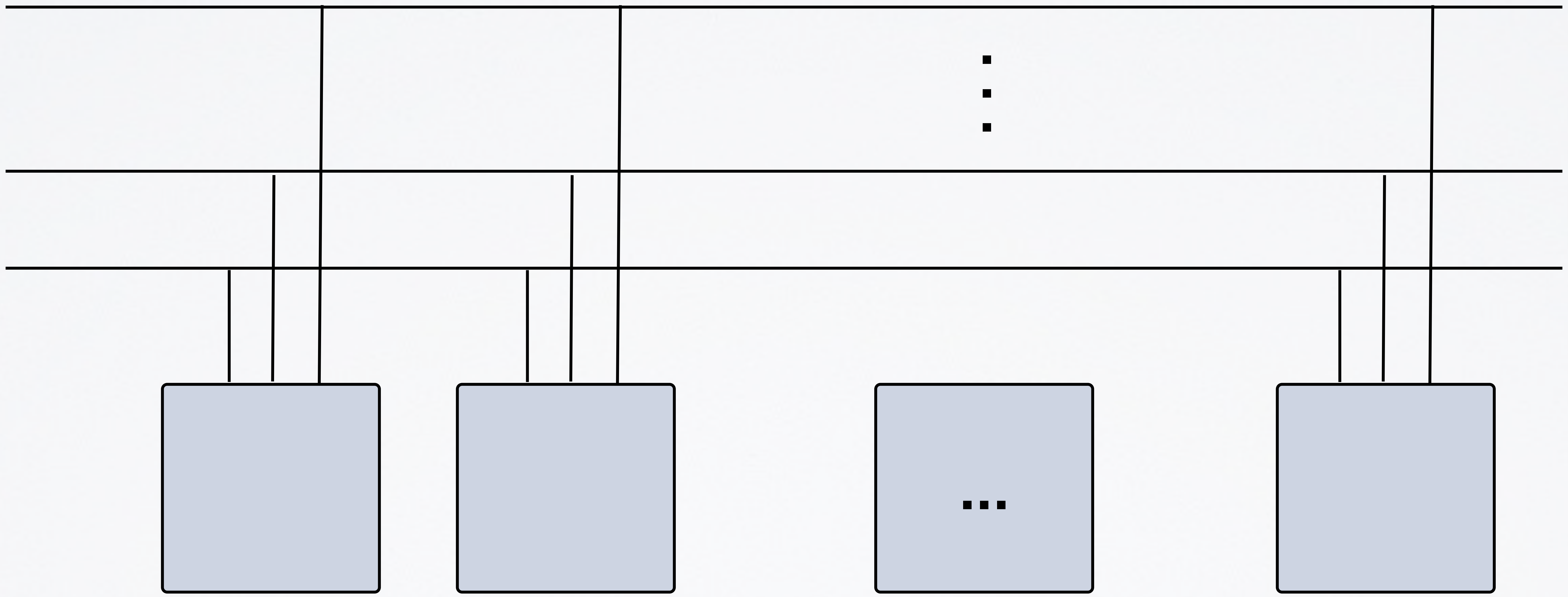
(Pfitzmann/Härtig 1982)



Q1/MODEL1: LIMITS OF RELIABILITY

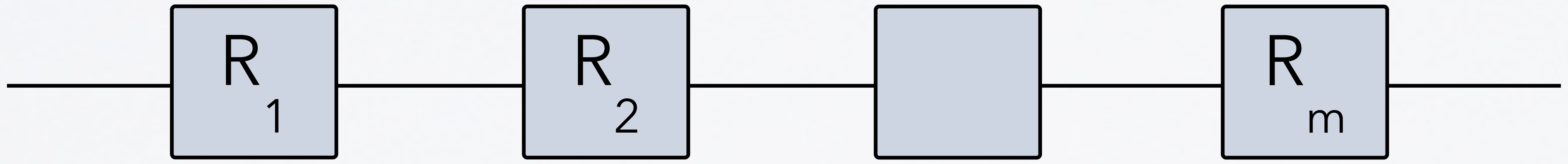
Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)



Q1/MODEL1: ABSTRACT RELIABILITY MODEL

Serial-Systems

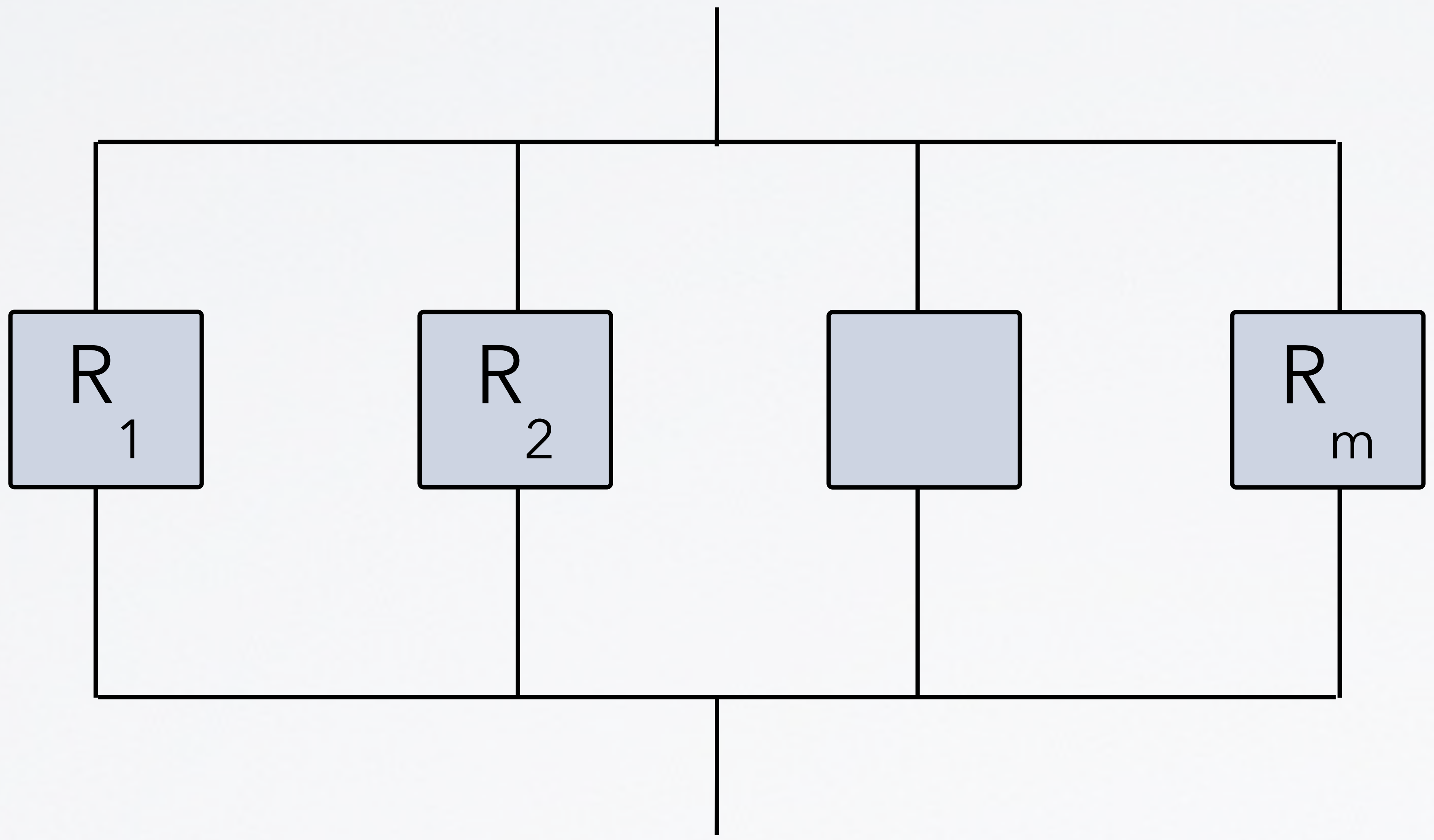


$$R_{whole} = \prod_{j=1}^{m} R_j$$

Each component must work for the whole system to work.

Q1/MODEL1: ABSTRACT MODEL

Parallel-Systems



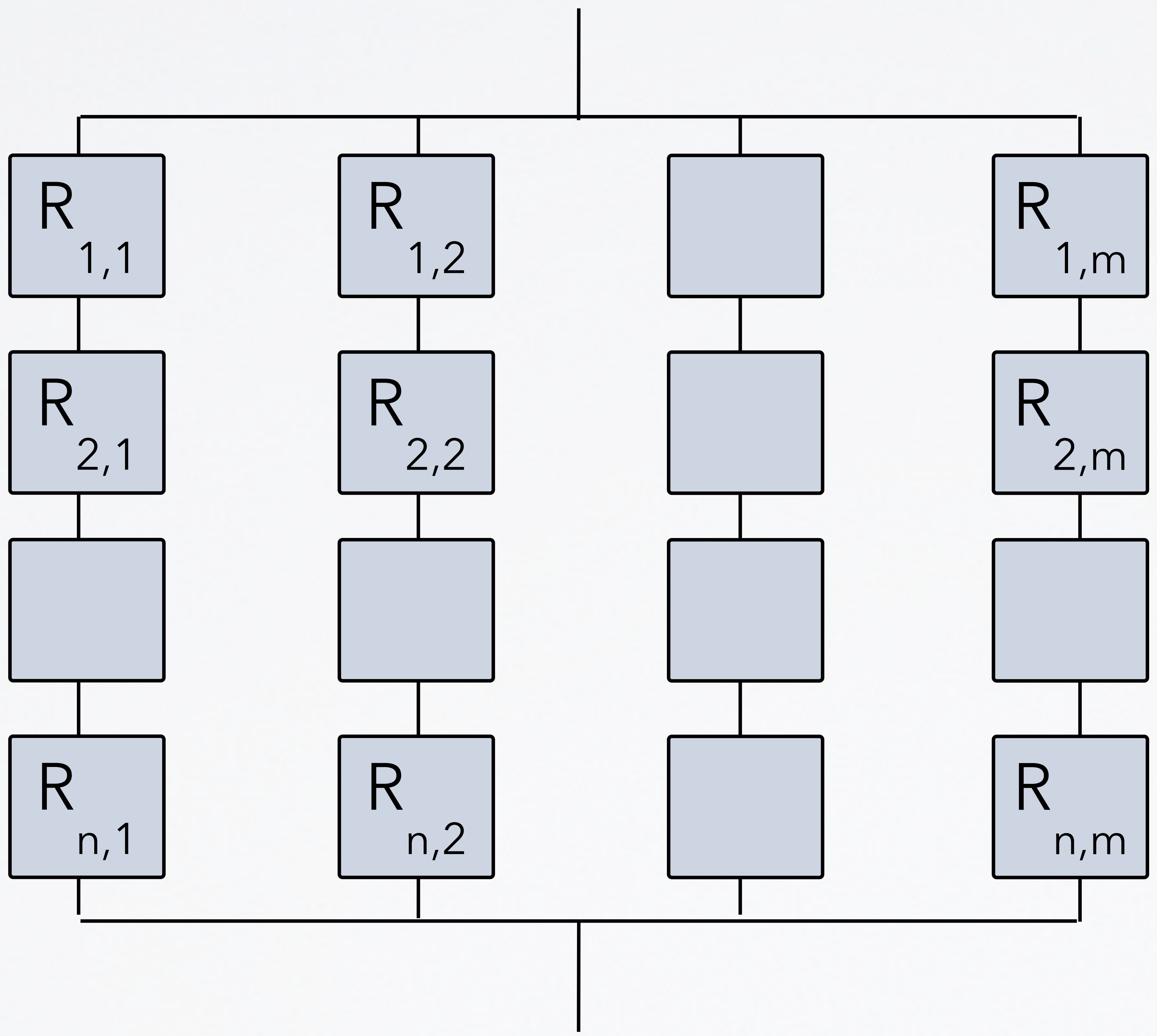
$$R_{whole} = 1 - \prod_{i=1}^{m} (1 - R_i)$$

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

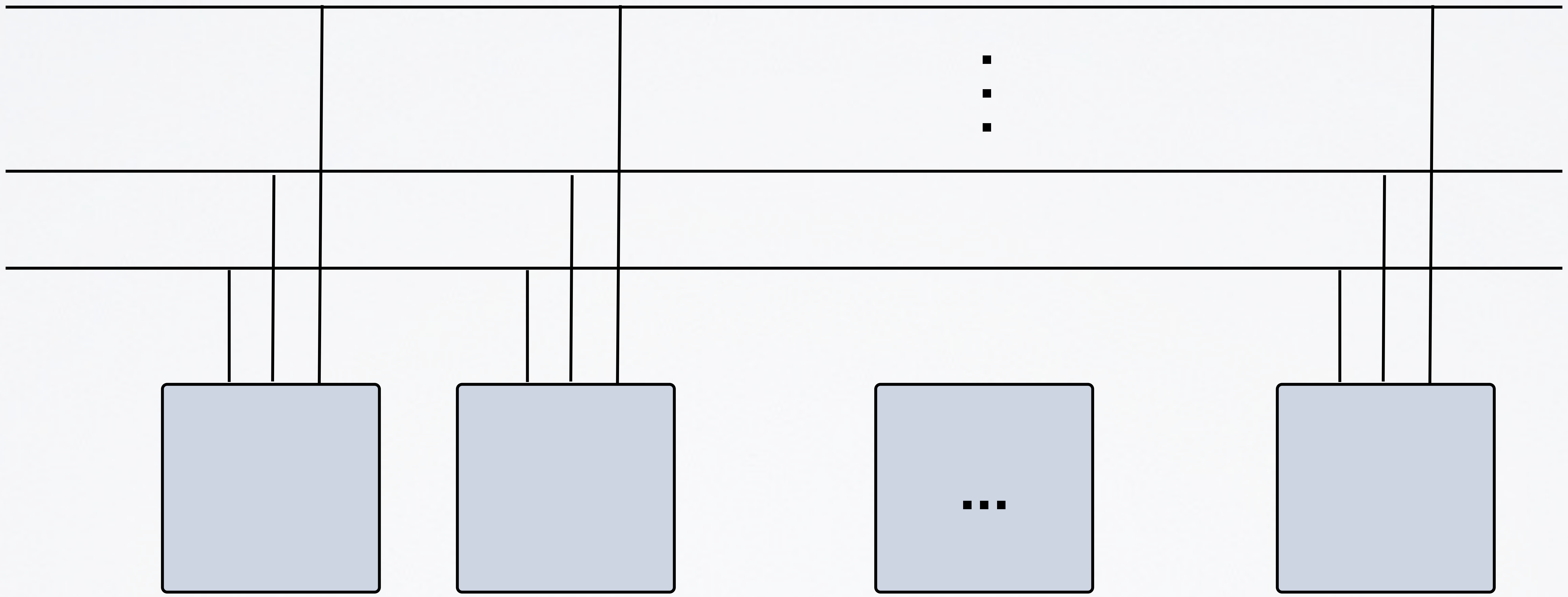


$$R_{whole} = 1 - \prod_{j=1}^{m} \left(1 - \prod_{i=1}^{n} R_{i,j} \right)$$

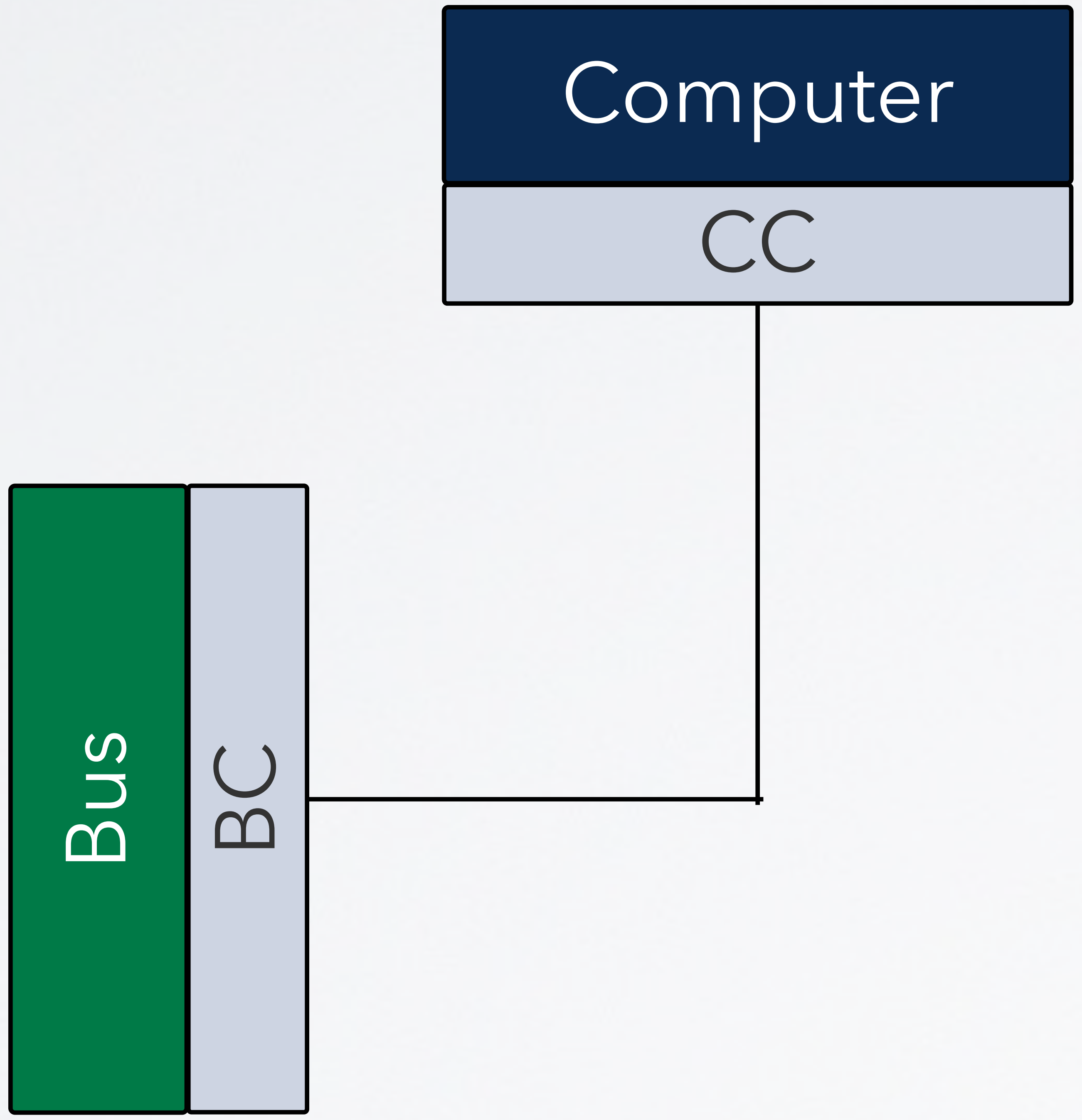
Q1/MODEL1: LIMITS OF RELIABILITY

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

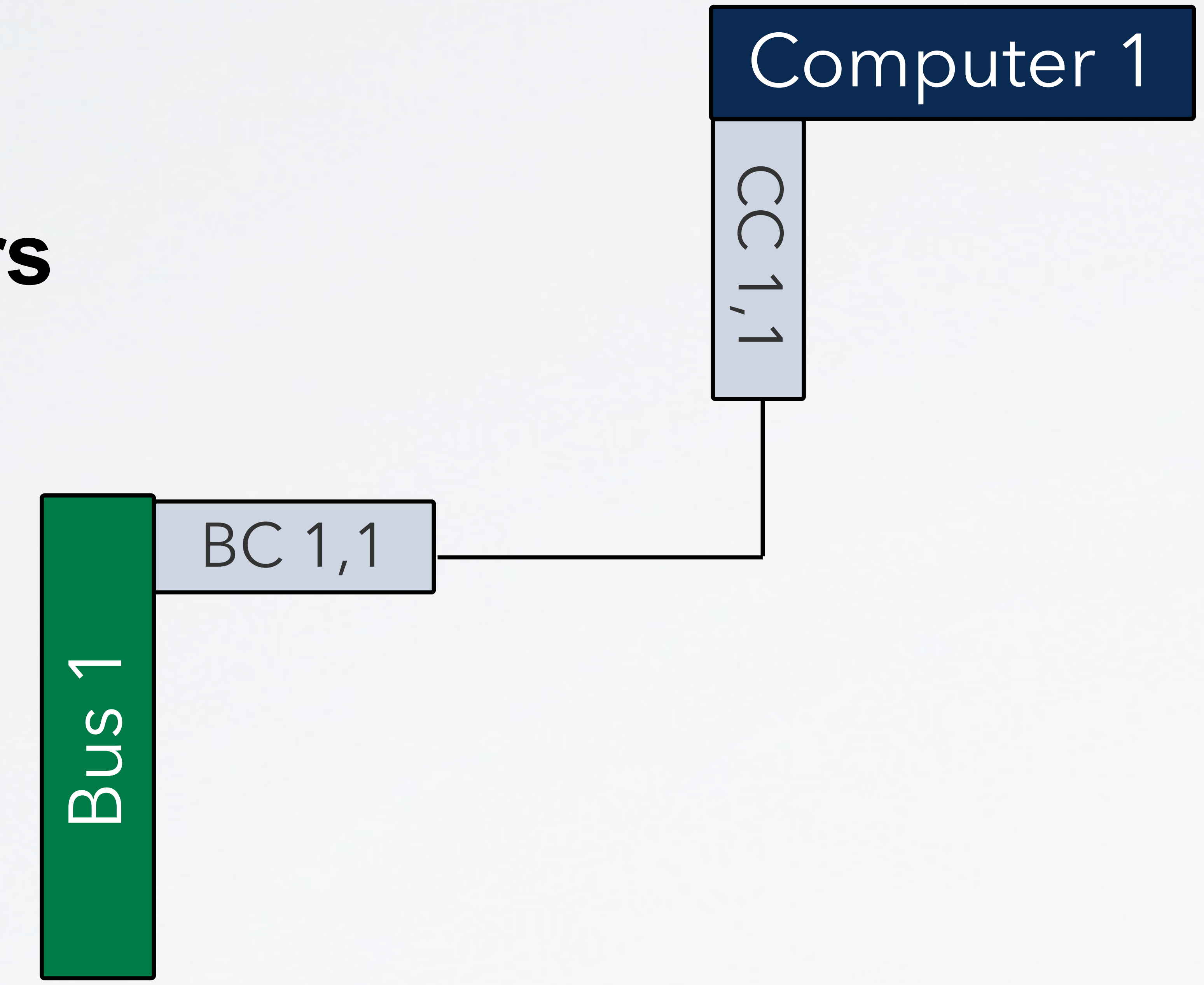
CC: Computer-Connector
fault also breaks the Computer

BC: Bus-Connector
fault also breaks Bus

Q1/MODEL1: CONCRETE MODEL

1 Buses

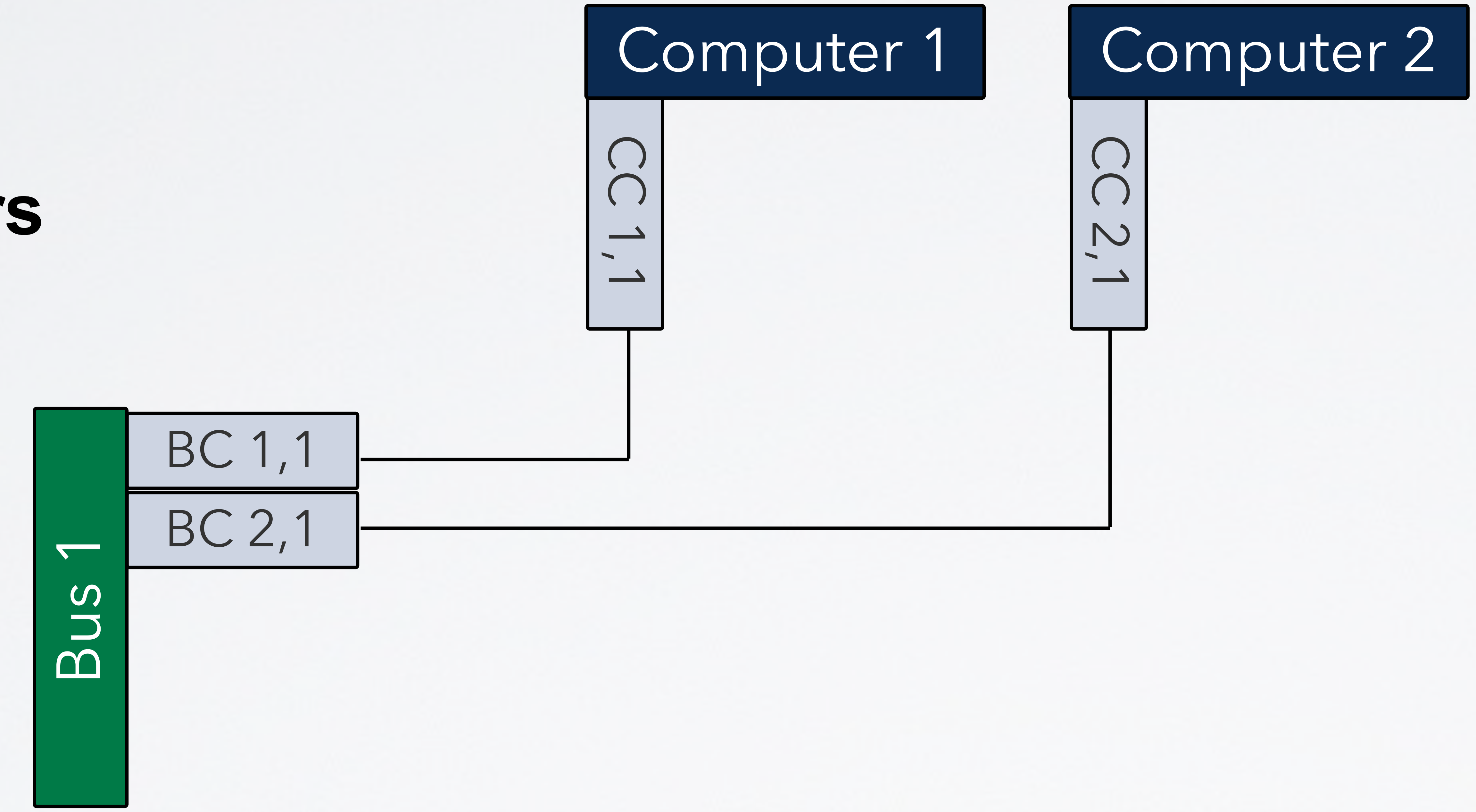
1 Computers



Q1/MODEL1: CONCRETE MODEL

1 Buses

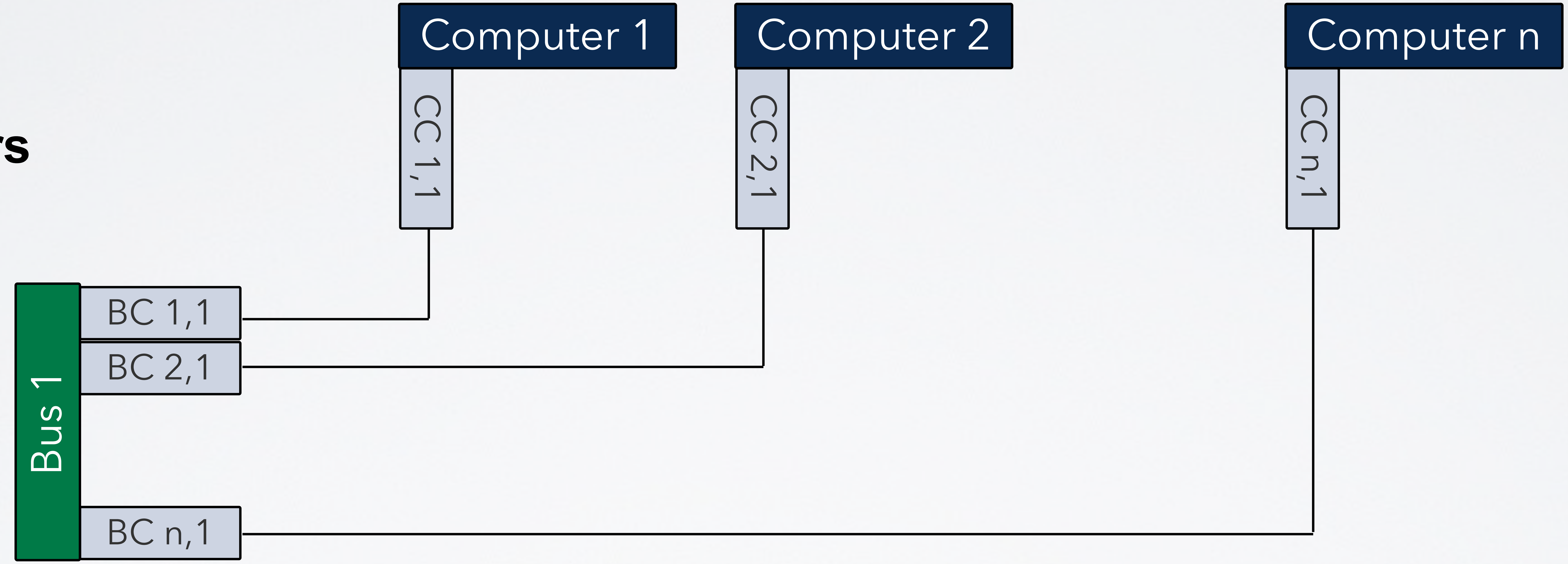
2 Computers



Q1/MODEL1: CONCRETE MODEL

1 Buses

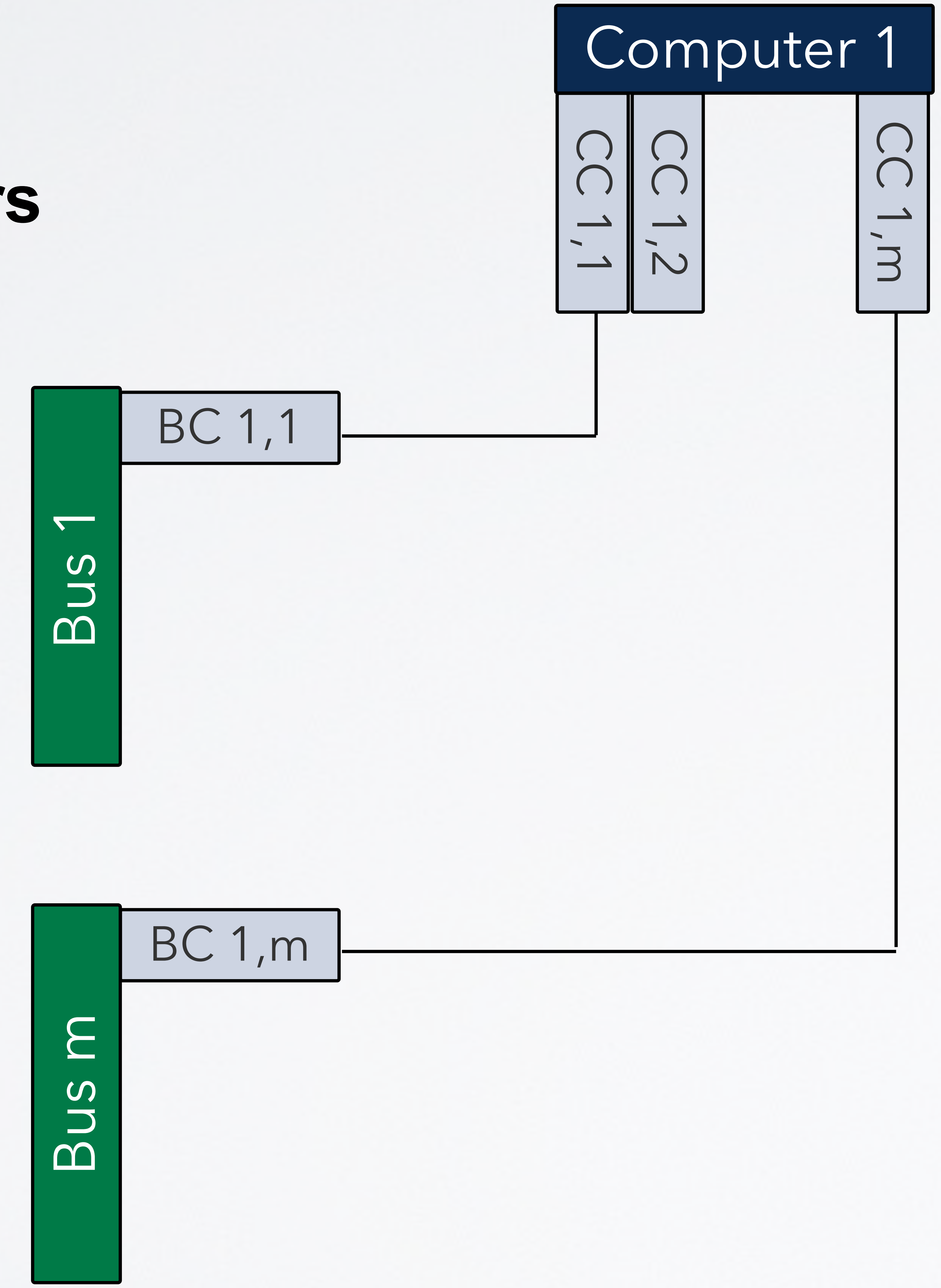
N Computers



Q1/MODEL1: CONCRETE MODEL

M Buses

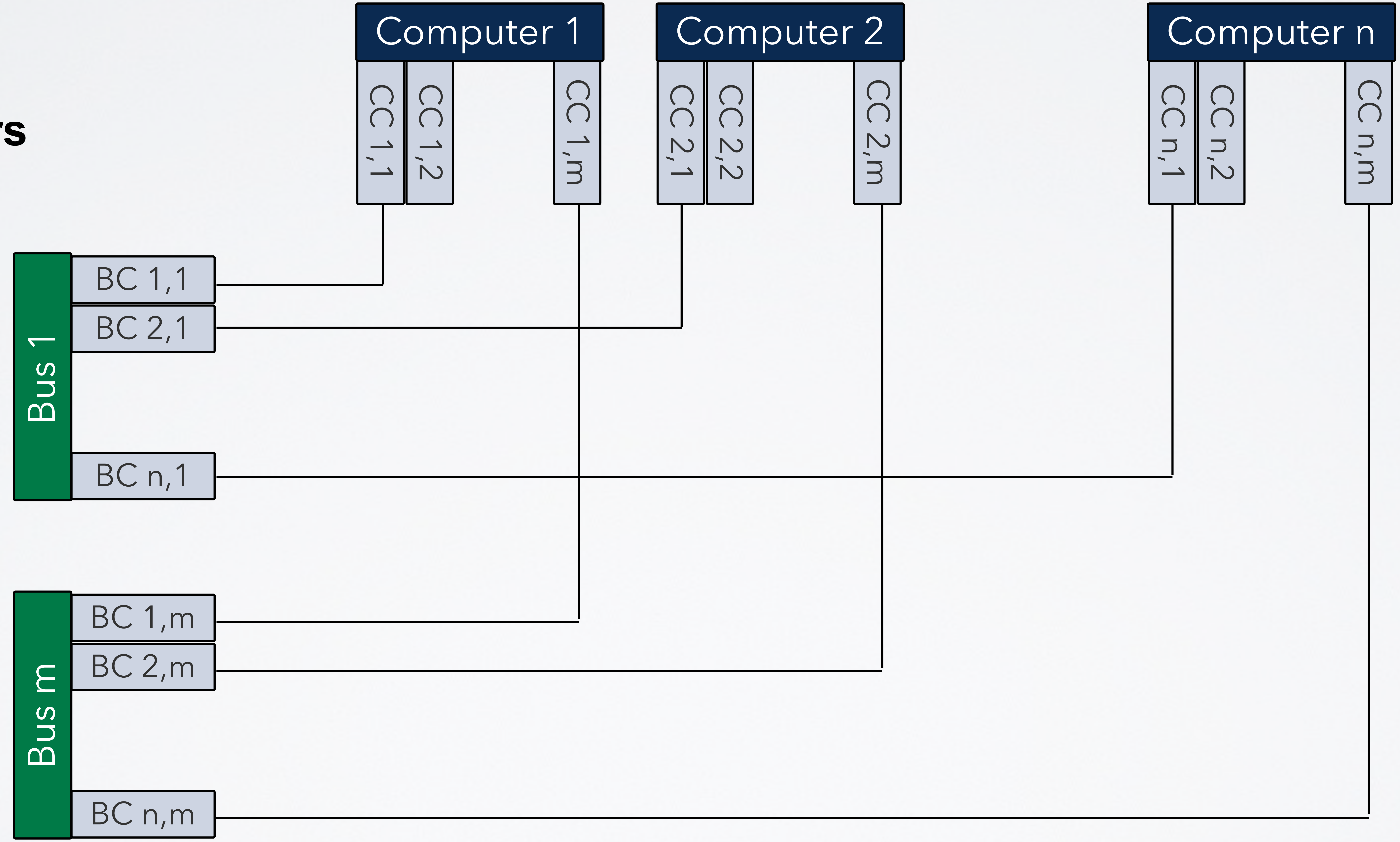
1 Computers



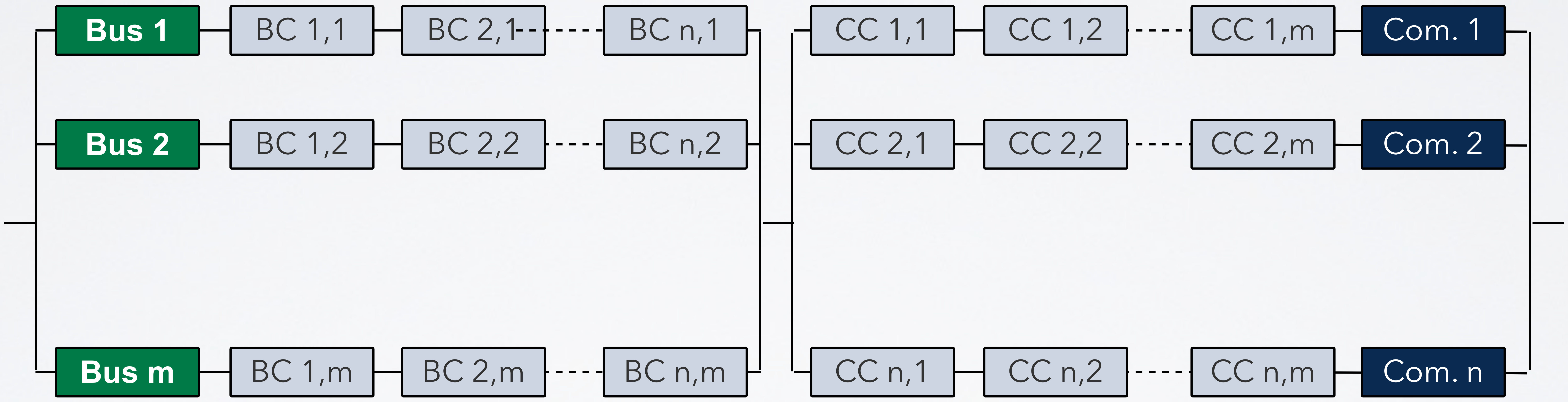
Q1/MODEL1: CONCRETE MODEL

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

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 !

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

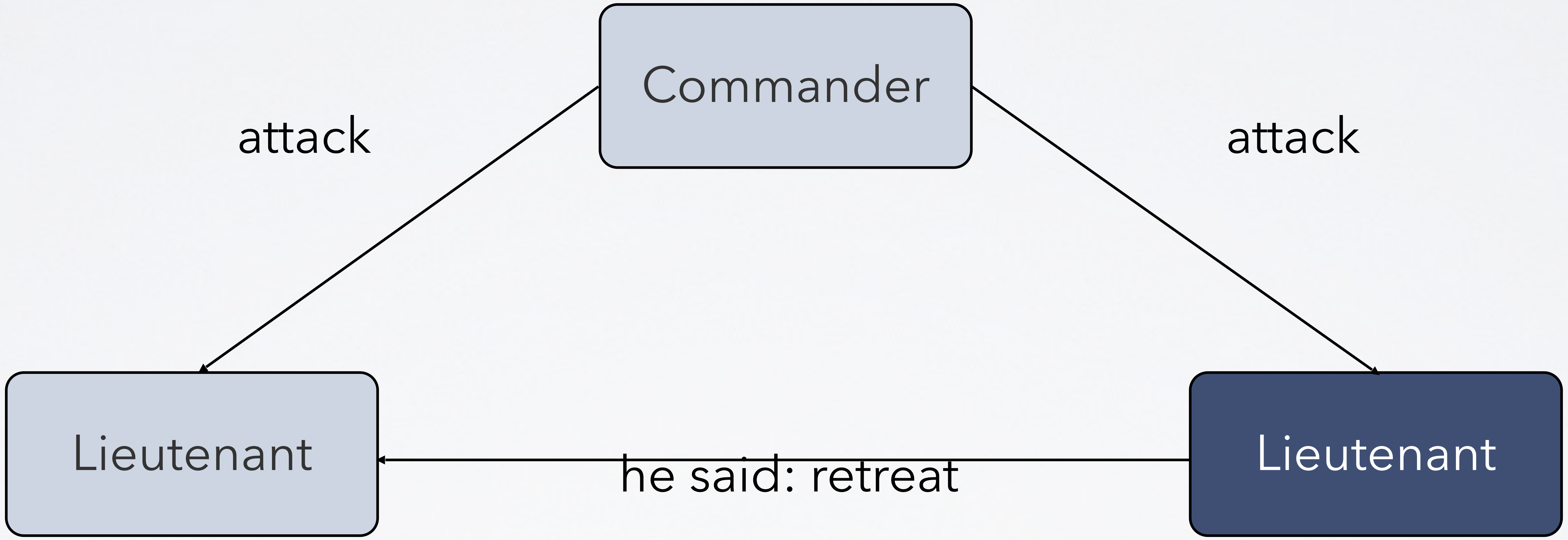
- communicate by reliable and timely messages (synchronous messages)
- traitors lye, also cheat on forwarding messages
- try to confuse loyals

Q2/MODEL 2: "BYZANTINE AGREEMENT"

Goal:

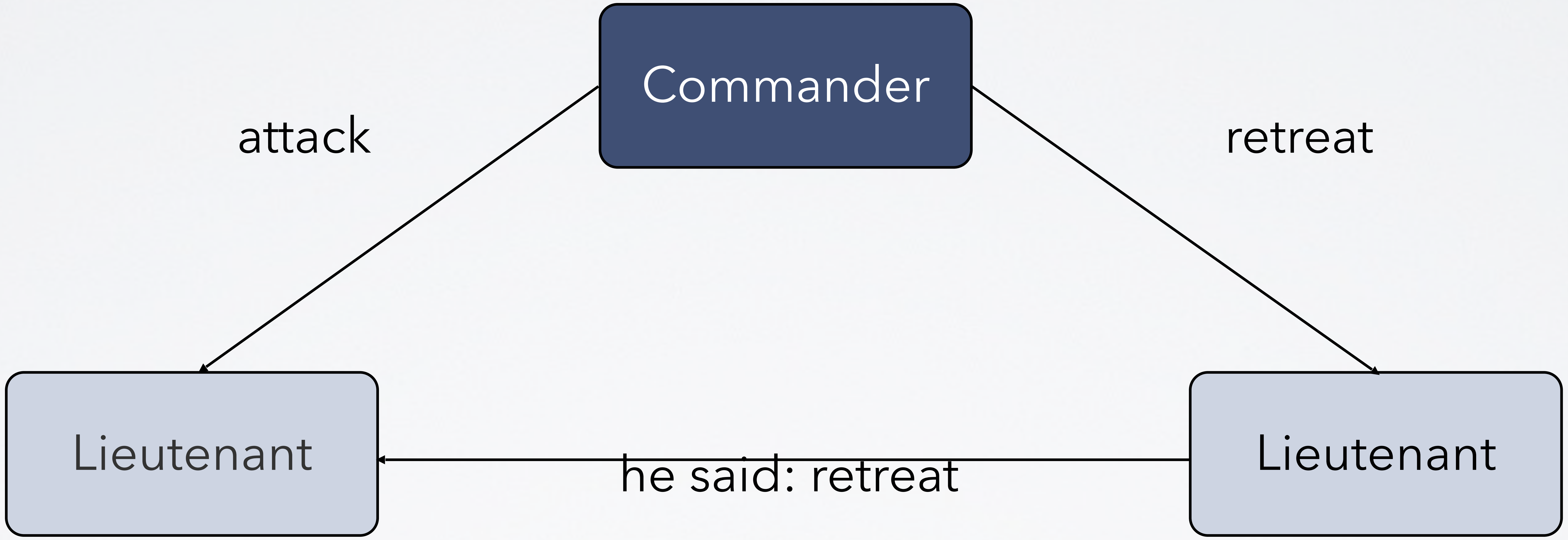
- loyals try to agree on non-trivial action (attack, retreat)
- non-trivial more specific:
  - one process is commander
  - if commander is loyal and gives an order, loyals follow the order otherwise loyals agree on arbitrary action

Q2/MODEL 2: "BYZANTINE AGREEMENT"



3 Processes: 1 traitor, 2 loyals

Q2/MODEL 2: "BYZANTINE AGREEMENT"

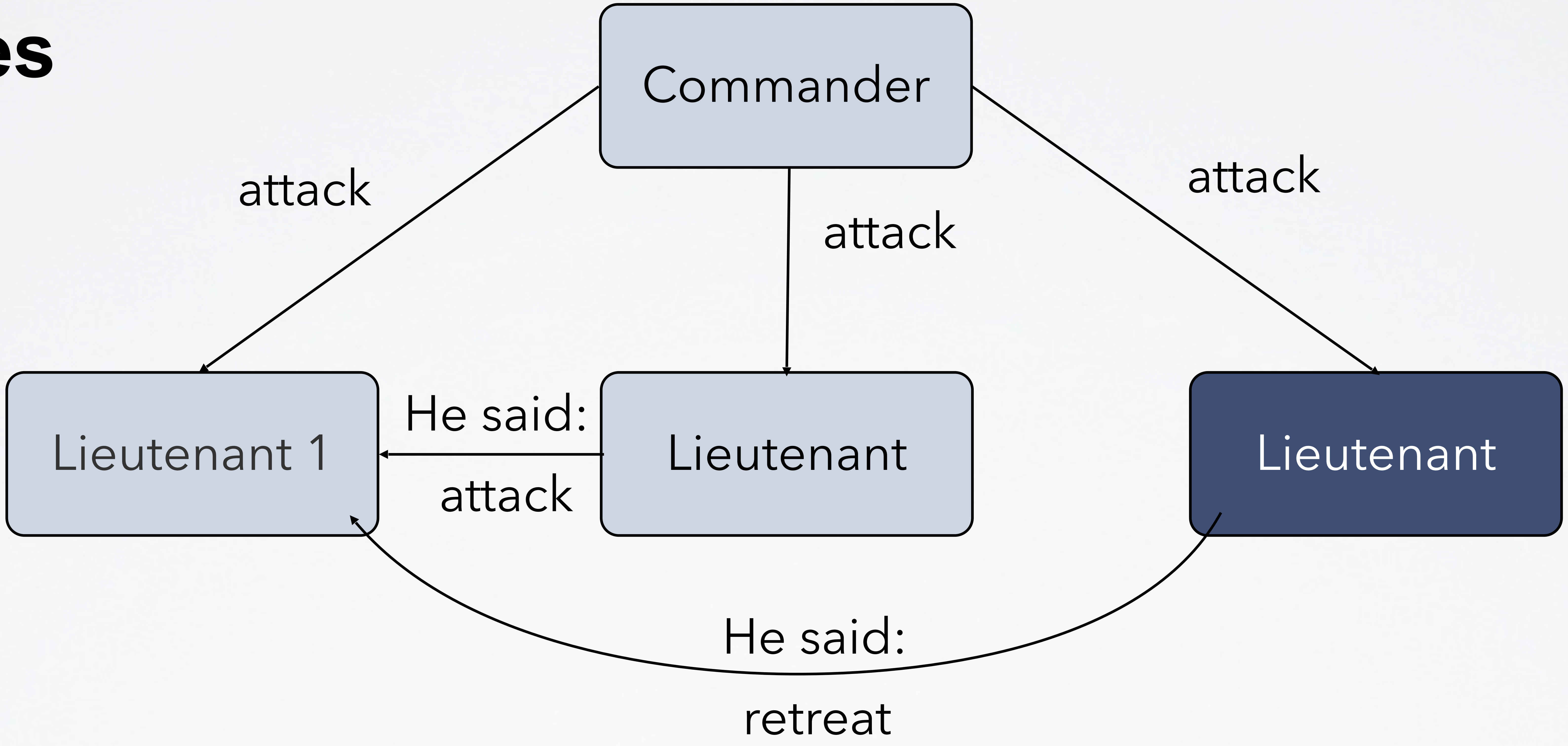


3 Processes: 1 traitor, 2 loyals

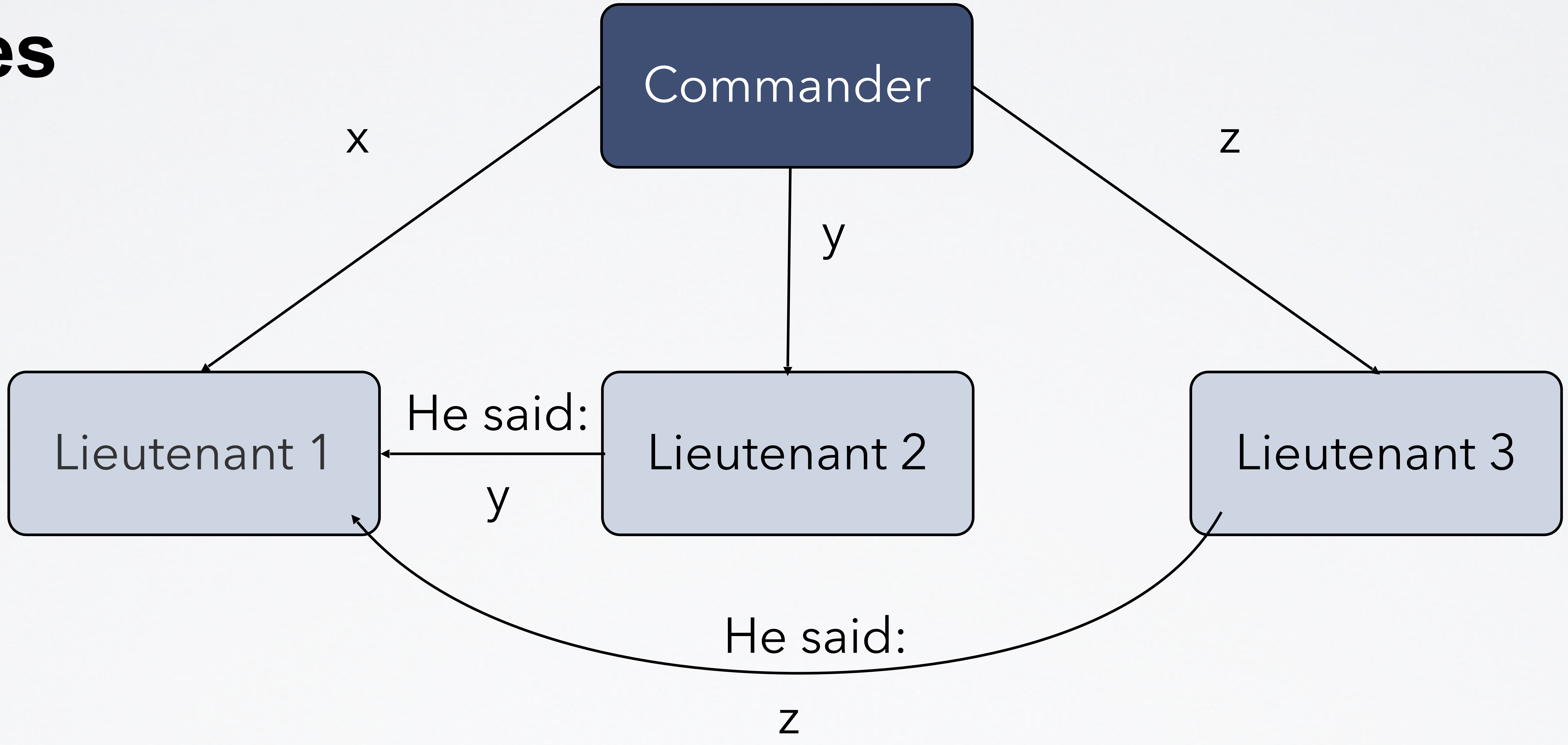
=> 3 processes not sufficient to tolerate 1 traitor

Q2/MODEL 2: "BYZANTINE AGREEMENT"

4 Processes



4 Processes



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

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

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 !

Q3: ACCESS CONTROL

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, acting as subjects and objects subjects performs operations on objects
- dynamic: subjects and objects are created and deleted
- access control permissions between entities can be changed according to some rules

THE GENERAL IDEA

higher level models:

- Bell La Padula,

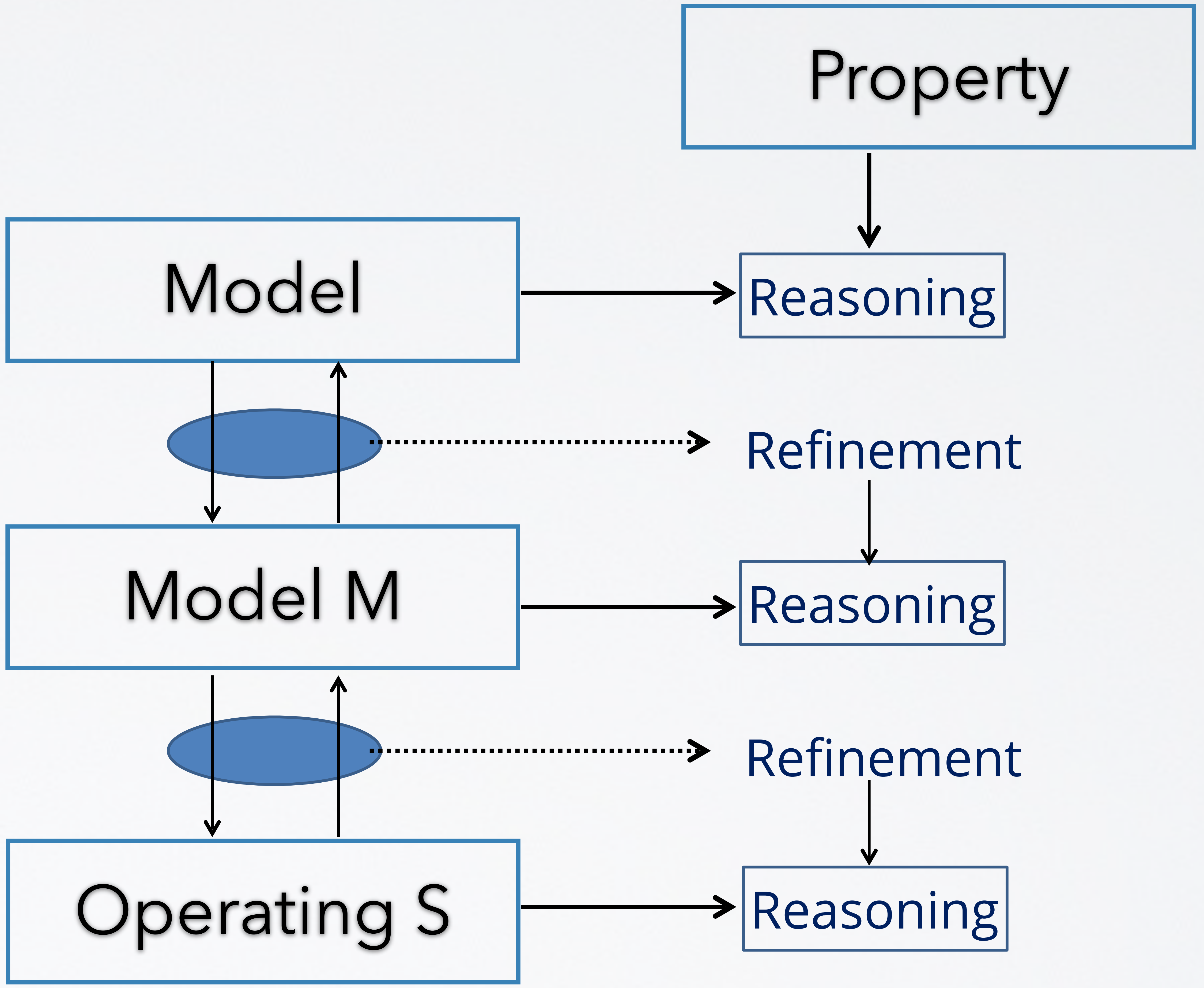
- Chinese wall

…

access control:

1) ACM-based operations

2) take grant



MECHANISMS: ACCESS CONTROL MATRIX

Subjects: S

Objects: O

Entities: E = S ∪ O

Rights: {read, write, own,…}

Matrix: S x E x R

Simple AC**M** Operations:

enter / delete R into cell (s,o)

create subject / object

destroy subject / object

	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

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 sequences of simple ACM operations
- only such defined mechanism provided by the OS can used to manipulate ACM

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"

Define OS-Mechanisms by simple ACM-Operations:

example:

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"

Example:

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

TECHNISCHE UNIVERSITÄT DRESDEN

Q3/MODEL 1: DECIDABILITY OF LEAKAGE

Decidable

- if no subjects/objects can be created

or

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

by exhaustive search !

Q3 in general:

- undecidable (proof: reduction to Turing machine)

ref MB: chapter 3

OS MECHANISMS: ACL & CAPS

ACM

	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

- 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

ref MB: chapter 2.2

Q3/MODEL 2: "TAKE GRANT"

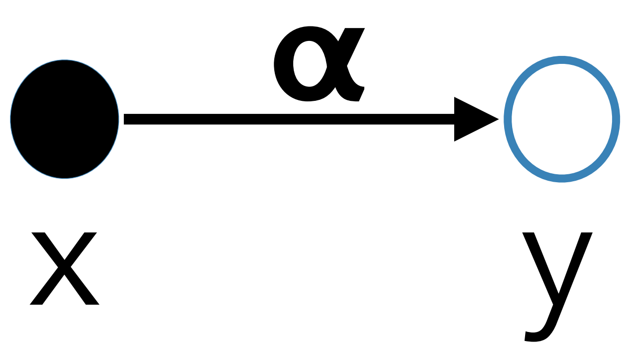
<u>Directed Graph:</u>

Subjects: ●

Objects: ○

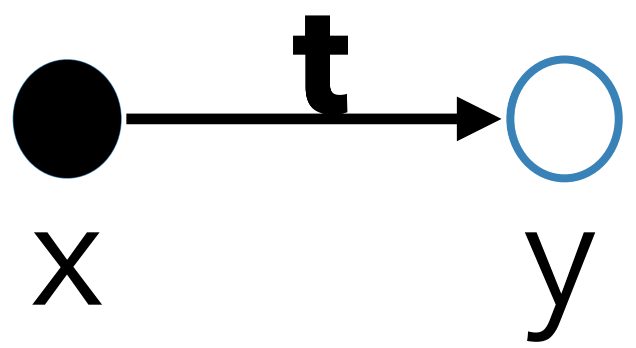
Either S or O: ⊗

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



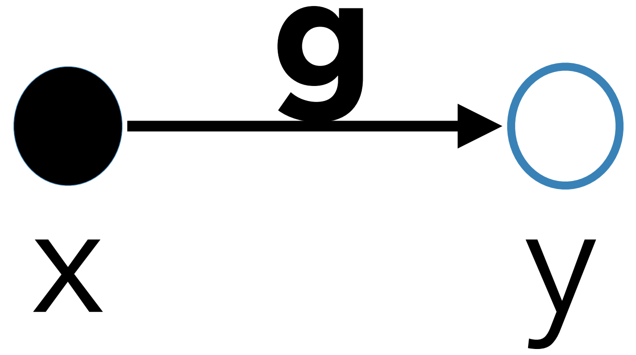
t <u>take right</u>

x has cap with set of rights
τ that includes t



g <u>grant right</u>

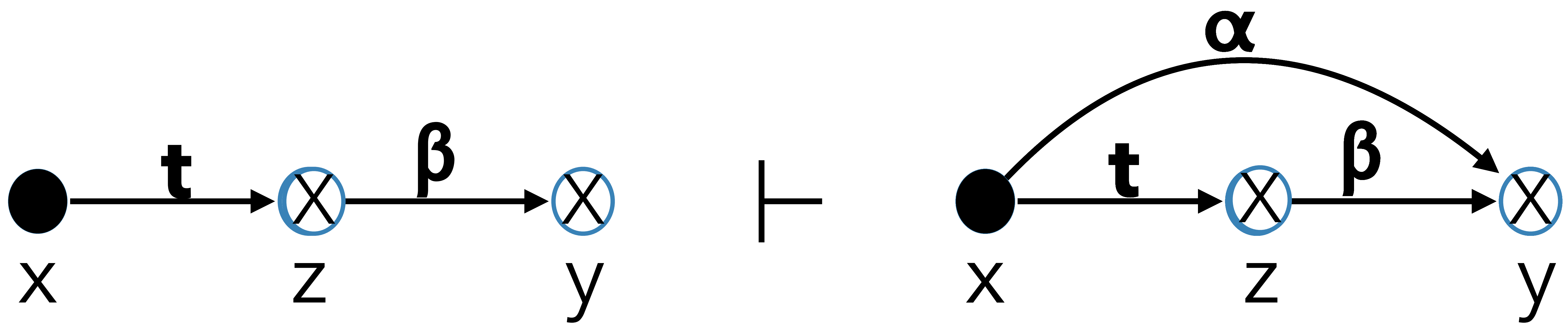
x has cap with set of rights
γ that includes **g**



Q3/ 2: TAKE GRANT RULES

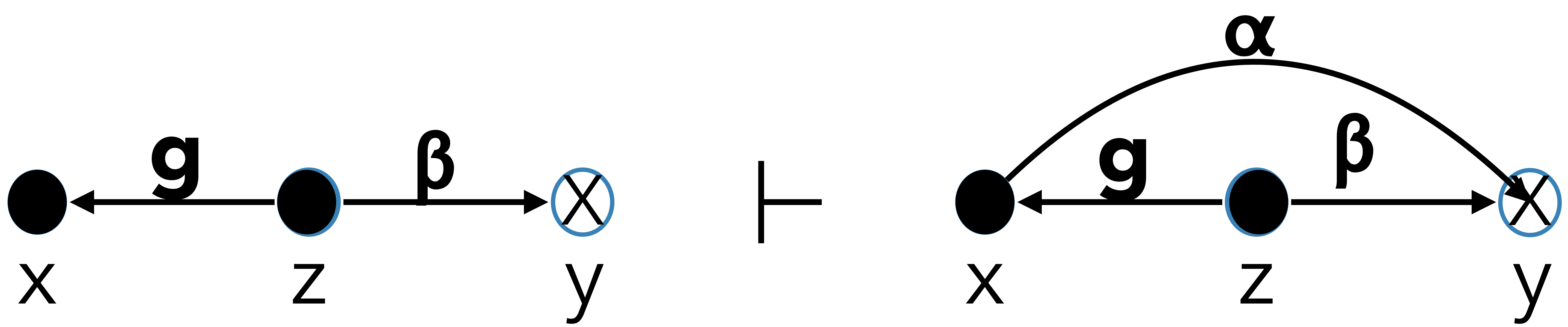
<u>Rules:</u>

take rule ($\boldsymbol{\alpha} \subseteq \boldsymbol{\beta}$)



a takes ($\boldsymbol{\alpha}$ to y) from z

grant rule ($\boldsymbol{\alpha} \subseteq \boldsymbol{\beta}$)

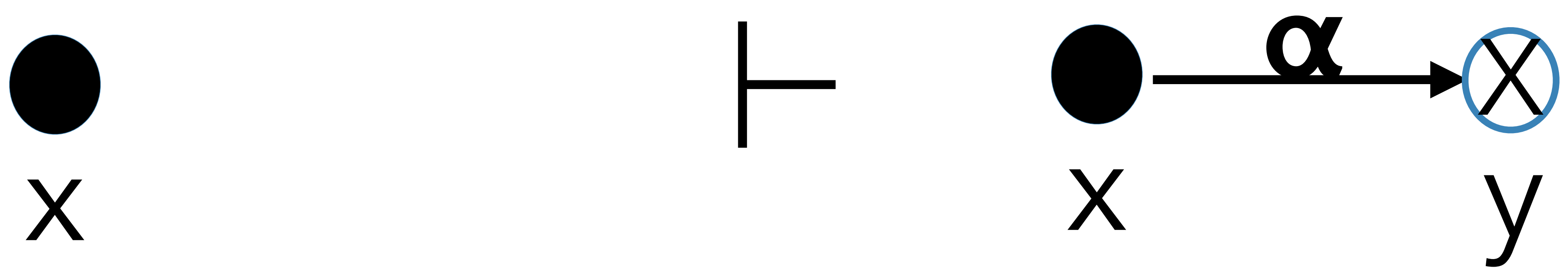


ref MB: chapter 3.3

Q3/ 2: TAKE GRANT RULES

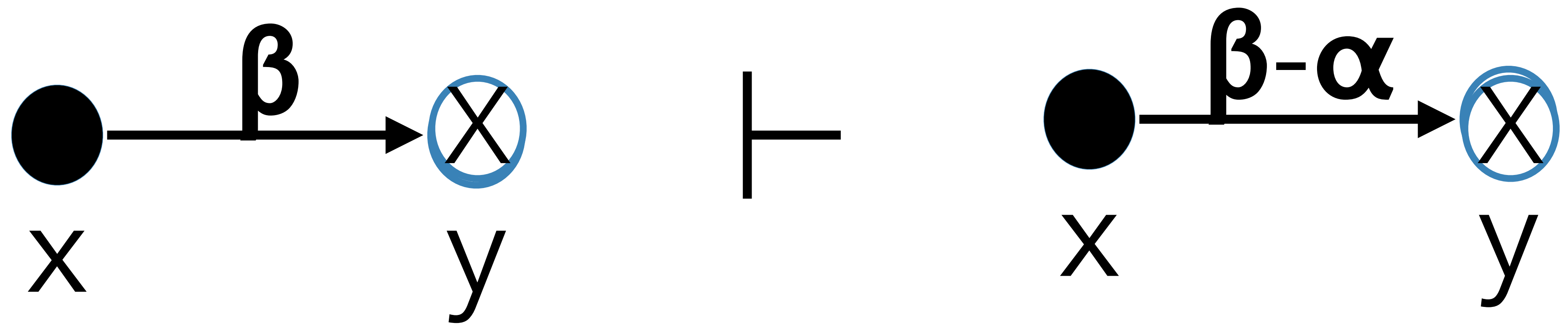
Rules:

create rule



x create (α to new vertex) y

remove rule



x removes (α to) y

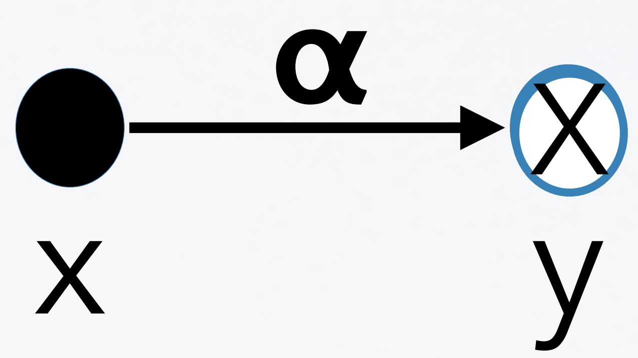
ref MB: chapter 3.3

Q3/M2: FORMALIZED

CanShare(α, x, y, G_0):

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

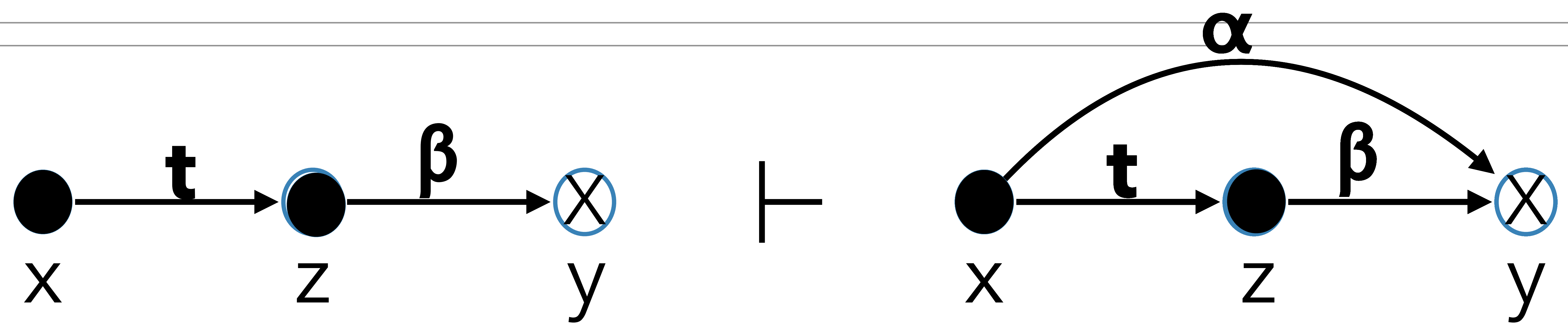
and there is an edge in G_n:



ref MB: chapter 3.3

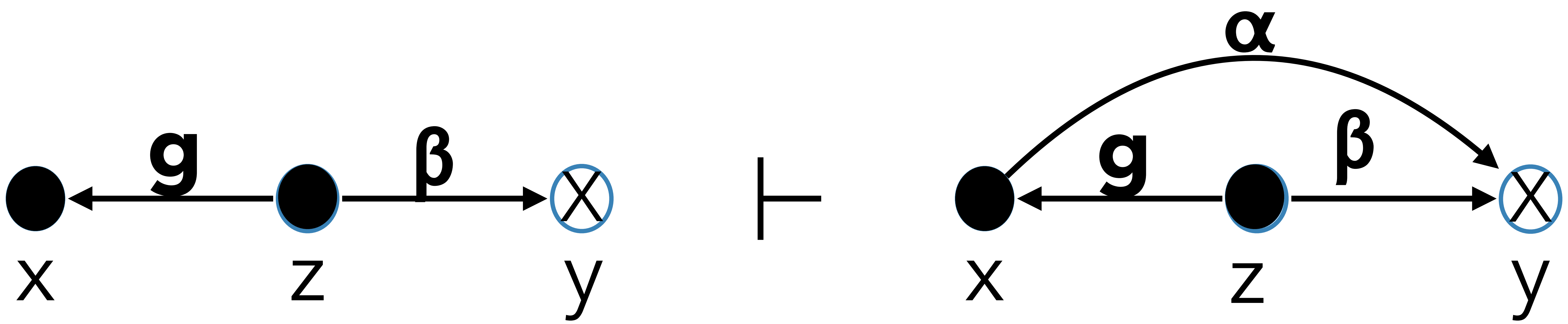
Q3/ 2: CAREFUL: LEMMA

<u>take rule (α⊆β)</u>



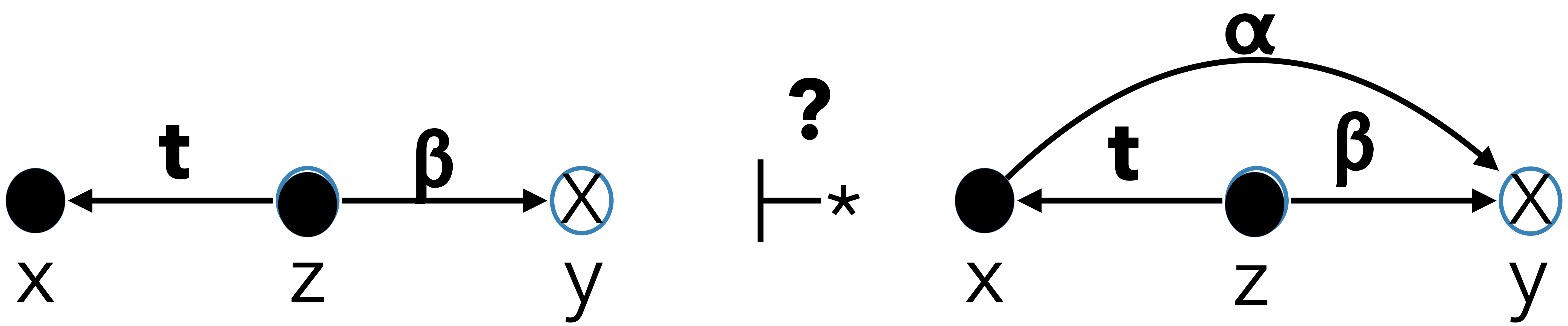
a takes (α to y) from z

<u>grant rule (α⊆β)</u>



z grants (α to y) to

Question:

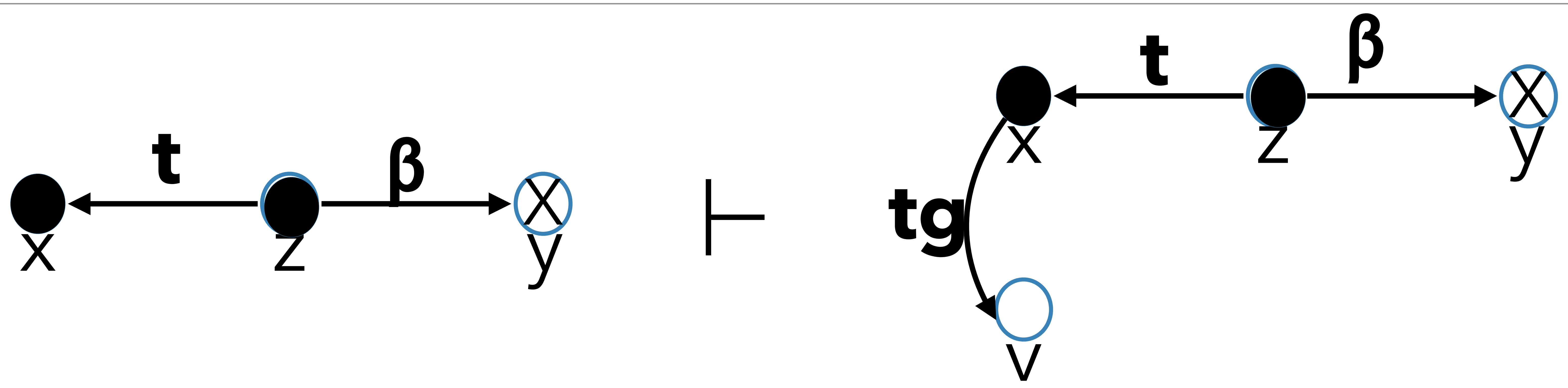


ref MB: chapter 3.3

Q3/ 2: CAREFUL: LEMMA

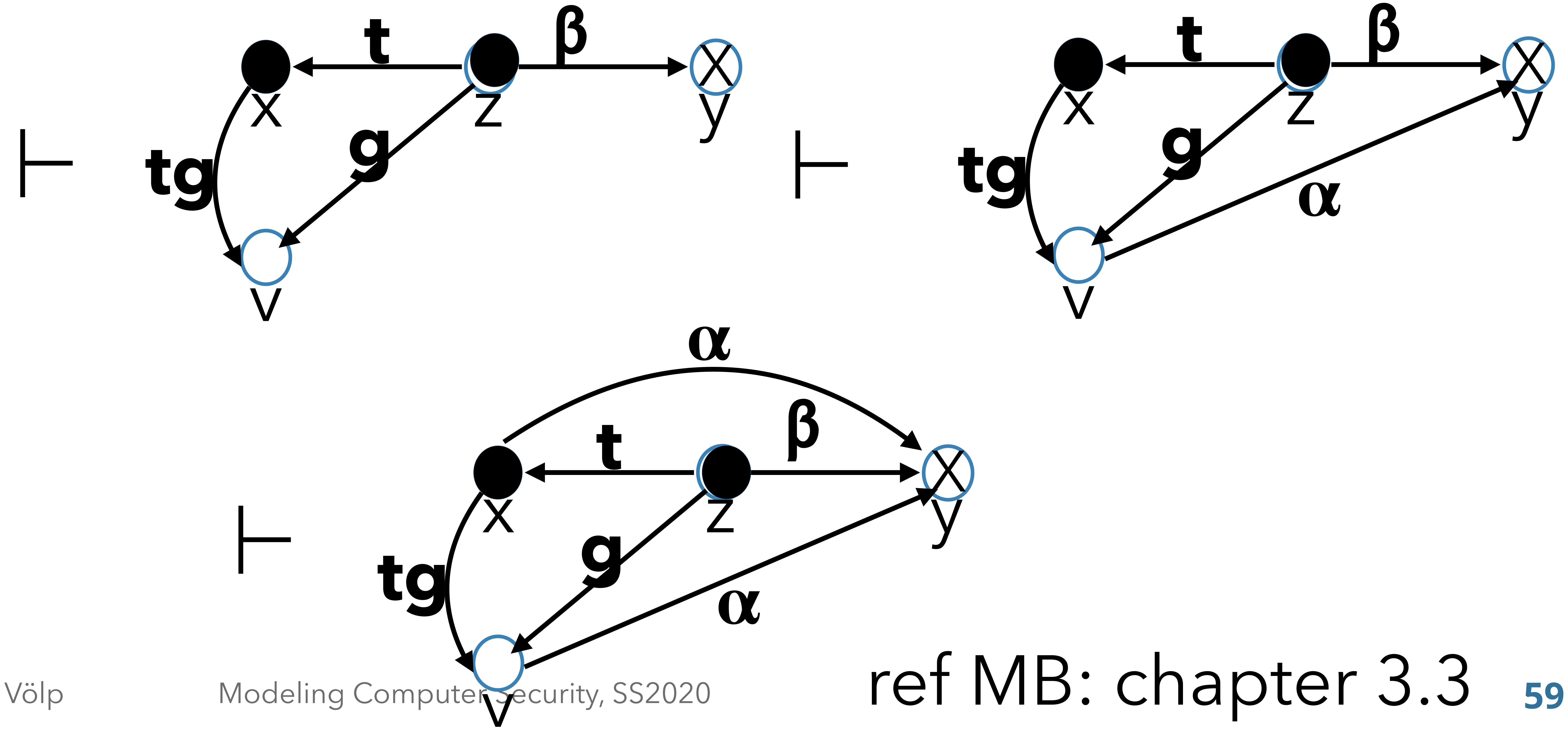
<u>(α⊆β)</u>

create rule



z takes (g to v) from x

z grants (α to y) to v



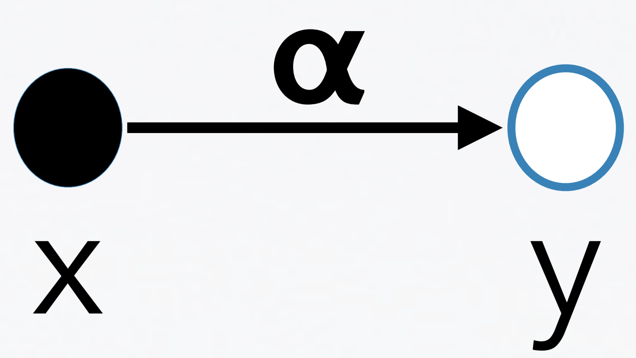
ref MB: chapter 3.3

Q3/M2: FORMALIZED

<u>CanShare(**α**, x, y, G_0):</u>

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

and there is an edge:



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