TECHNISCHE UNIVERSITÄT DRESDEN

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

BASIC INTRO: COMPUTER SECURITY MODELS

HERMANN HÄRTIG, MARCUS VÖLP, 2017

CLOSELY FOLLOWING PRESENTATION IN

MB: MATT BISHOP "COMPUTER SECURITY ART AND SCIENCE" (2003)

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 2017)

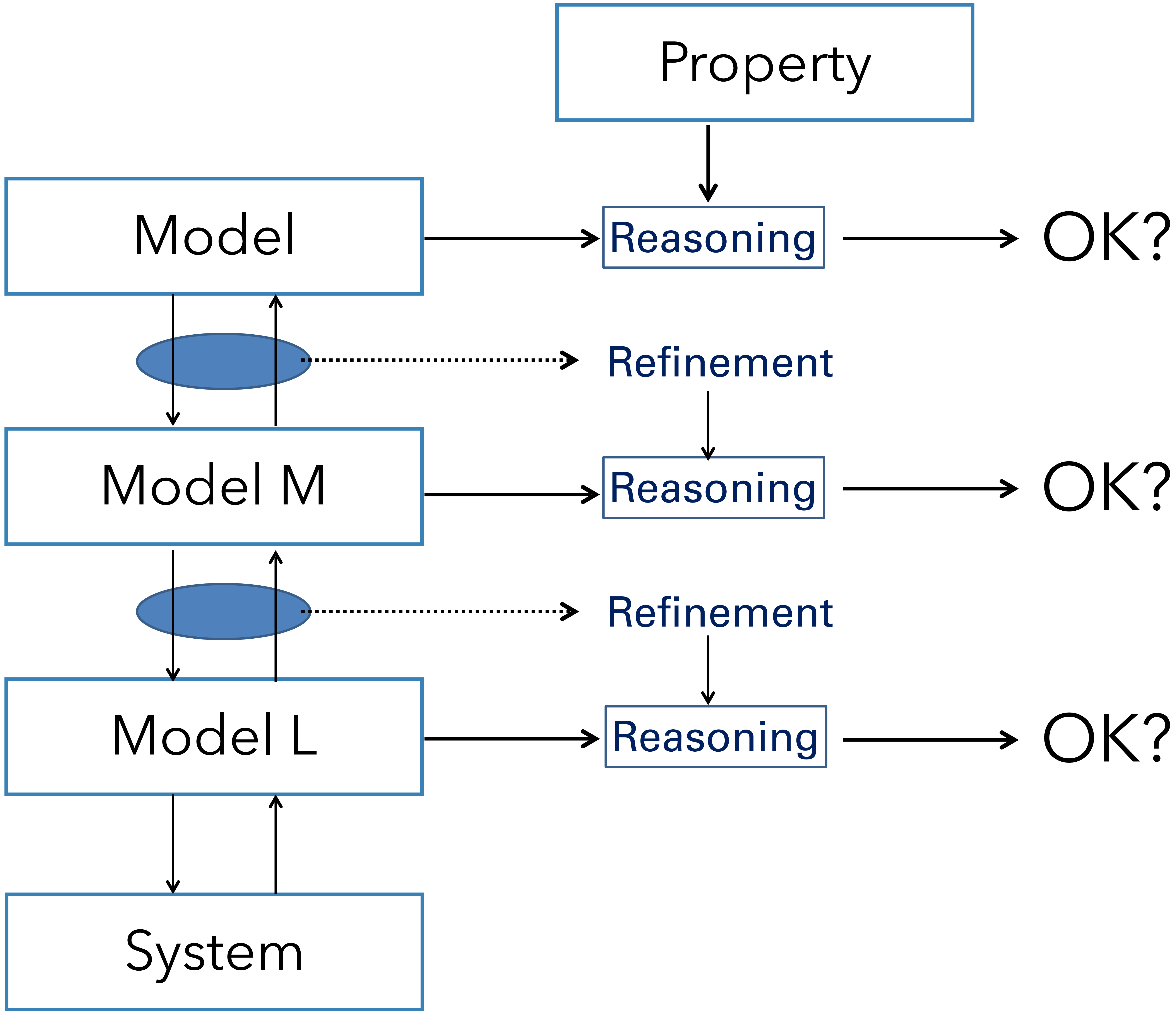
THE GENERAL APPROACH

"Reasoning":

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

"Refinement":

- Abstraction
- Implementation
- Formal Refinement



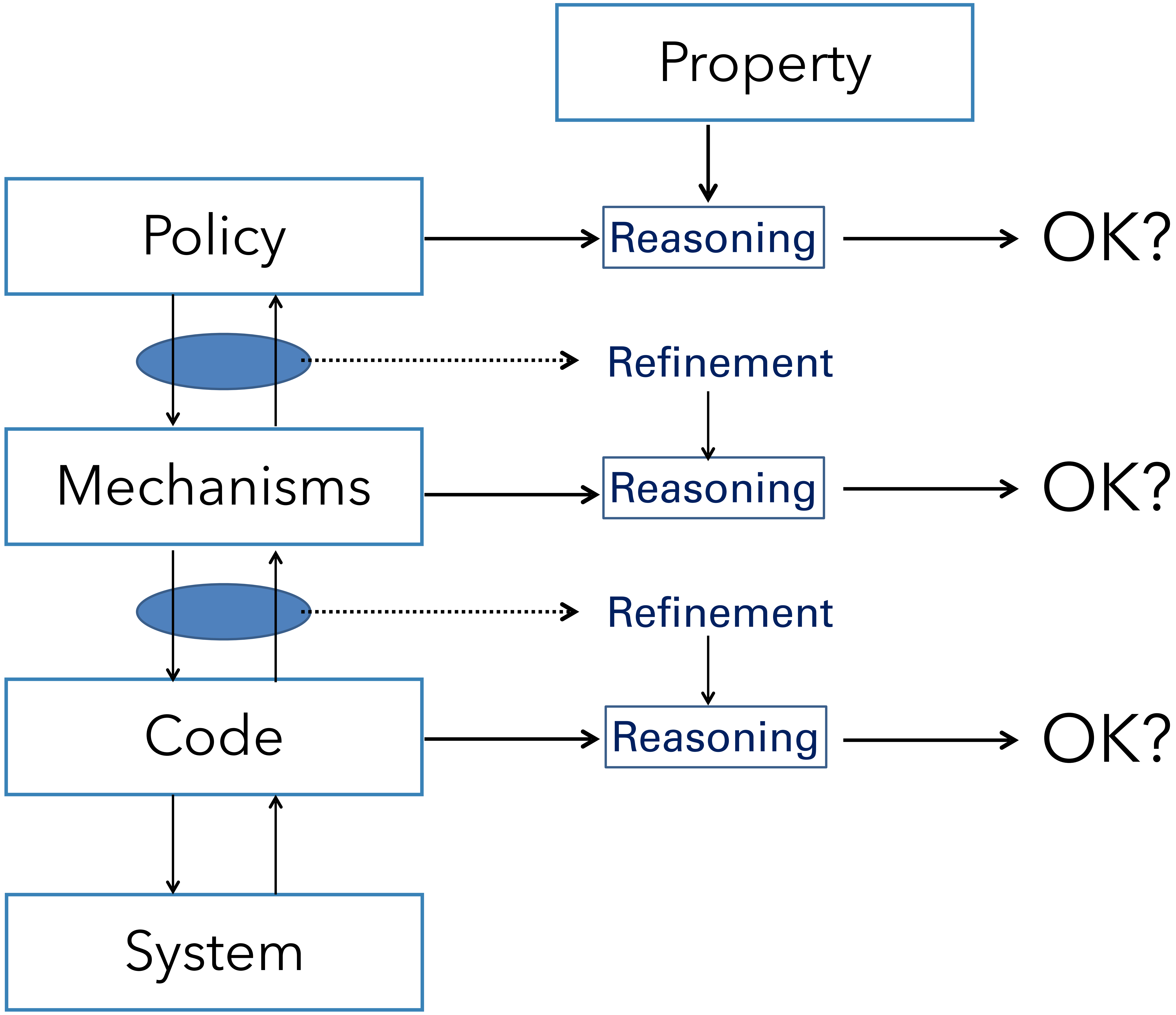
SECURITY MODELS

"Reasoning":

- Common sense
- Formal Verification
- Careful Inspection
- Mathematics …
- "Common Criteria Assurance"

"Refinement":

- Abstraction
- Implementation
- Formal Refinement



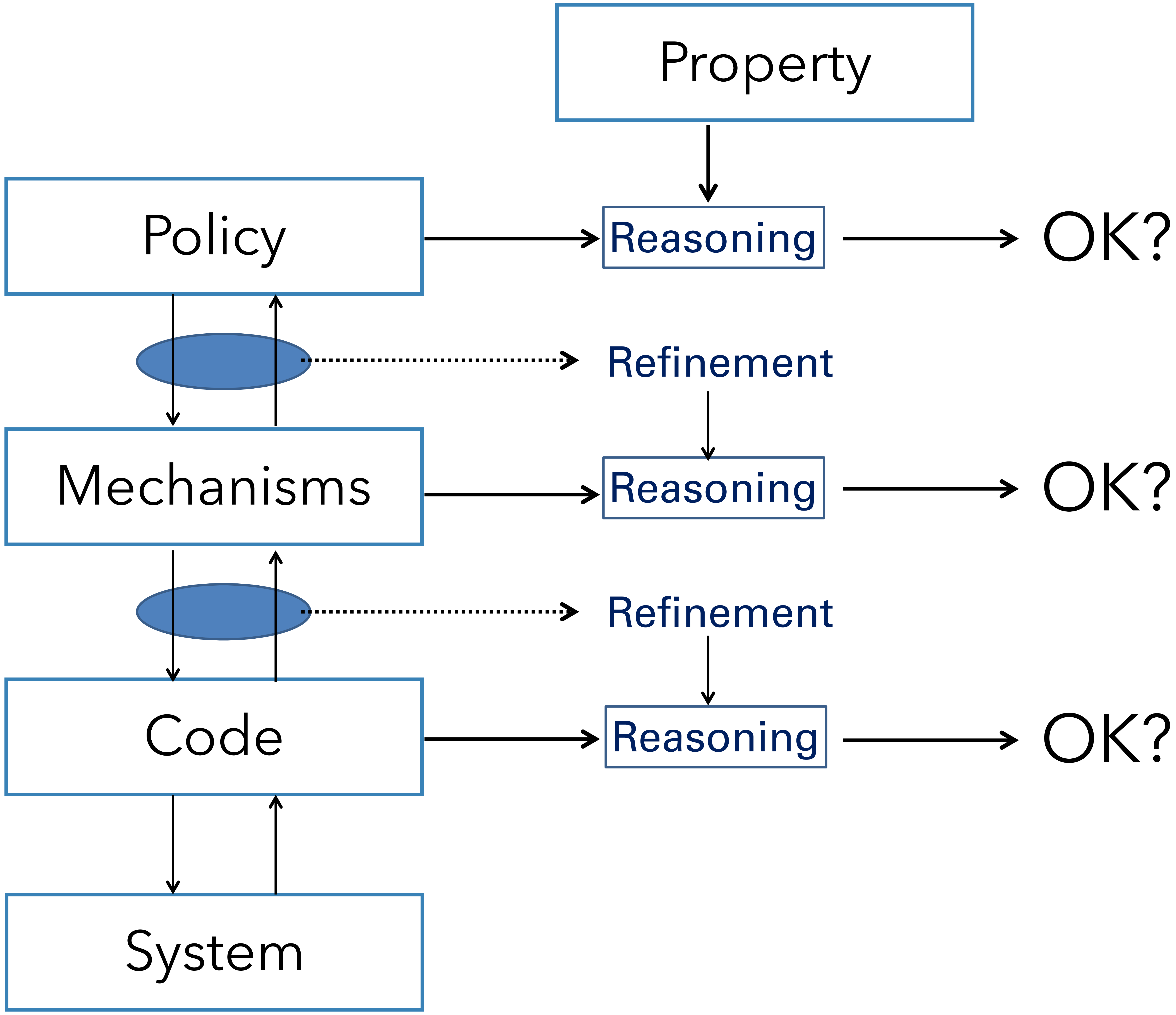
IN THIS LECTURE

Definiton: Policy

Examples:

Higher-Level Policies (very short):

- Bell La Padula
- Chinese Wall



IN THIS LECTURE

Operating Sys. Mechanisms:

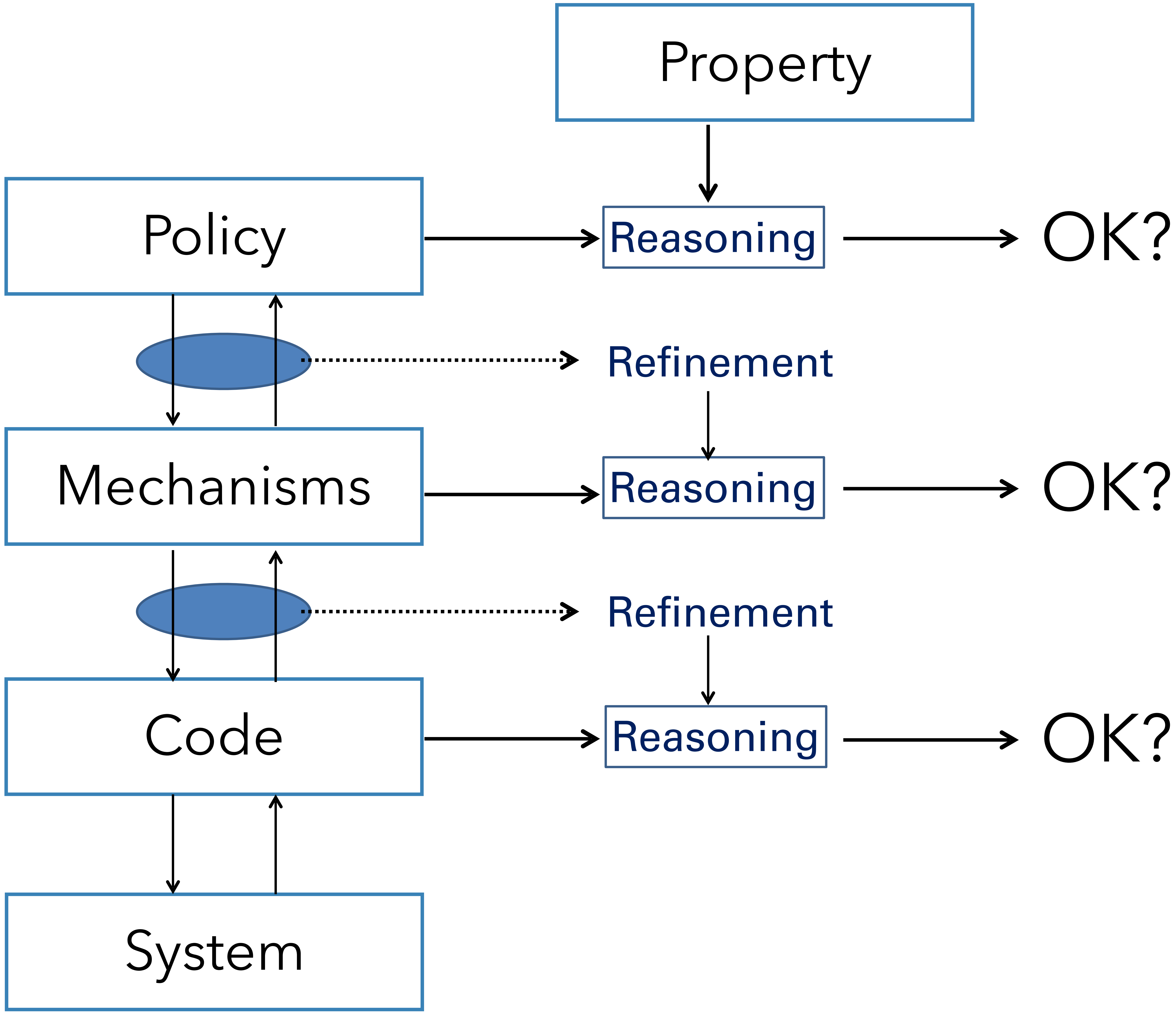
- Access Control List
- Capabilities

Explain Q3 and ...

formalize per model!

Models:

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



SECURITY POLICY

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

Definitions:

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 possible to detect that these properties do not hold.

INFORMAL BELL LAPADULA

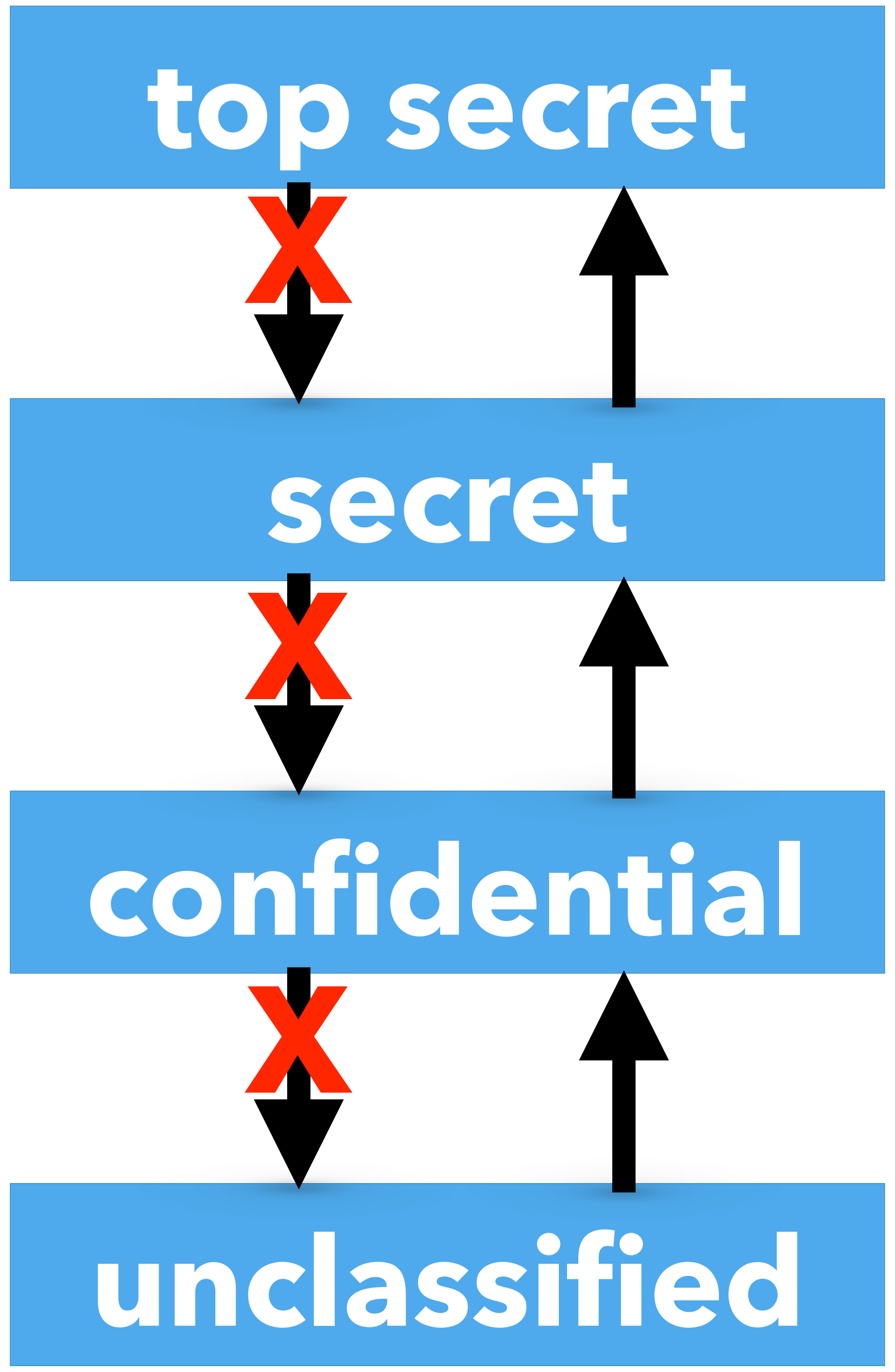
Model for Confidentiality

Secrecy Levels:

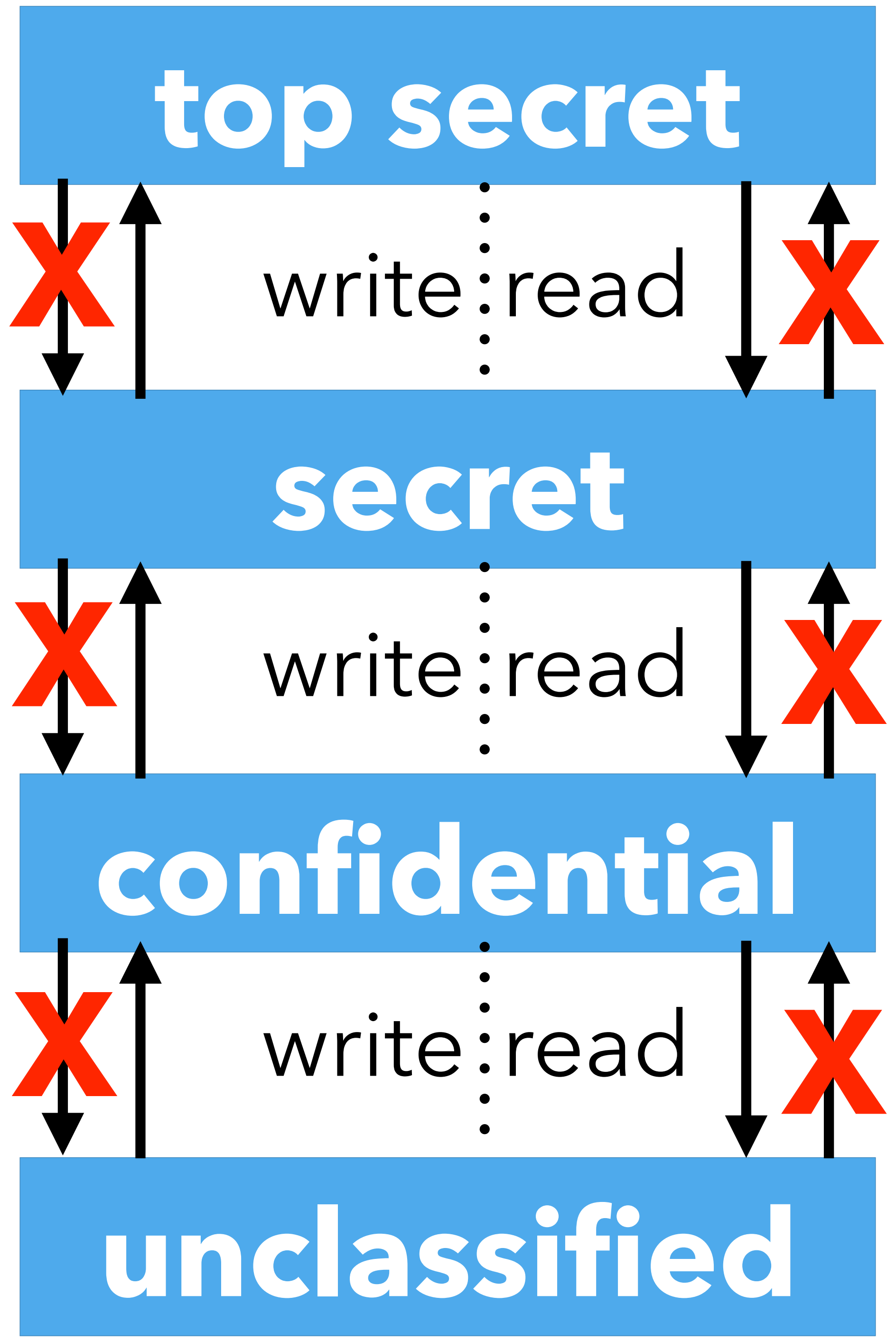
- Classification (documents)
- Clearance (persons)
- The higher the level the more sensitive the data
- totally ordered

Categories

information



operations



EXAMPLES BLP(TANENBAUM)

- categories: NATO, Nuclear
  levels/clearance: top secret, secret, confidential, unclassified
- document: Nato, secret
- <u>person clearance: read</u>

person clearance	read
secret, Nato	-> allowed
secret, Nuclear	-> not allowed
confidential, Nato	-> not allowed

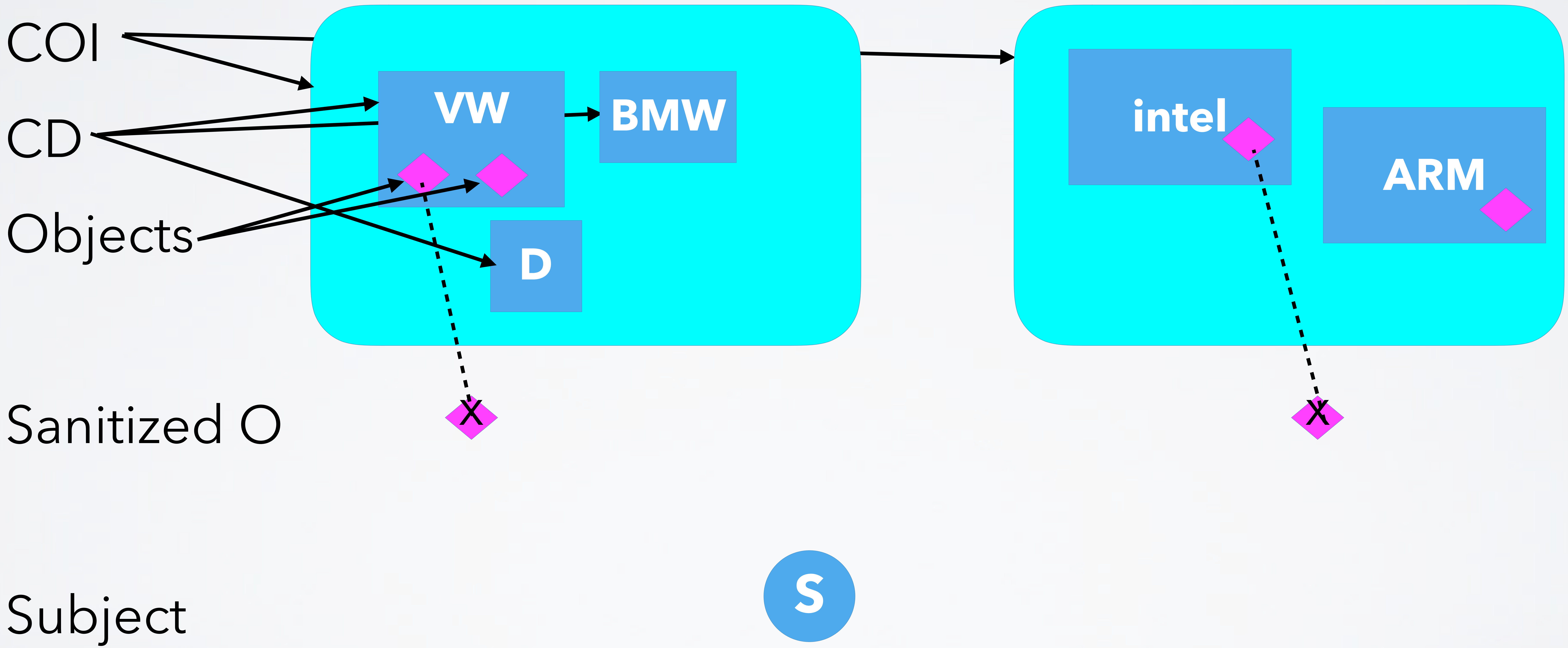
CHINESE WALL POLICY

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 contain critical information

Ref MB: Chapter 7.1

CHINESE WALL, EXAMPLE



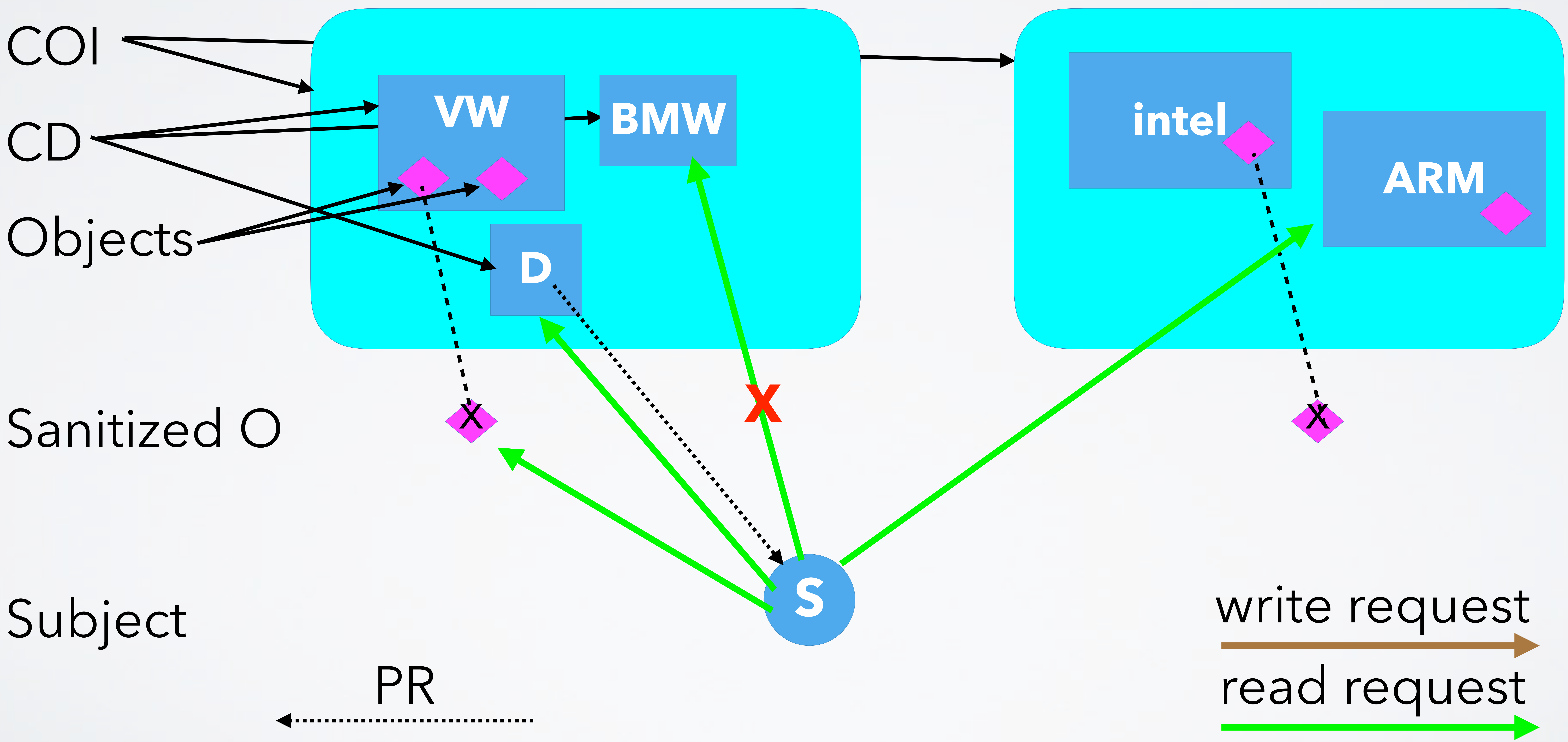
CHINESE WALL, RULES

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

S can read O, if any of the following holds

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

CHINESE WALL, EXAMPLE



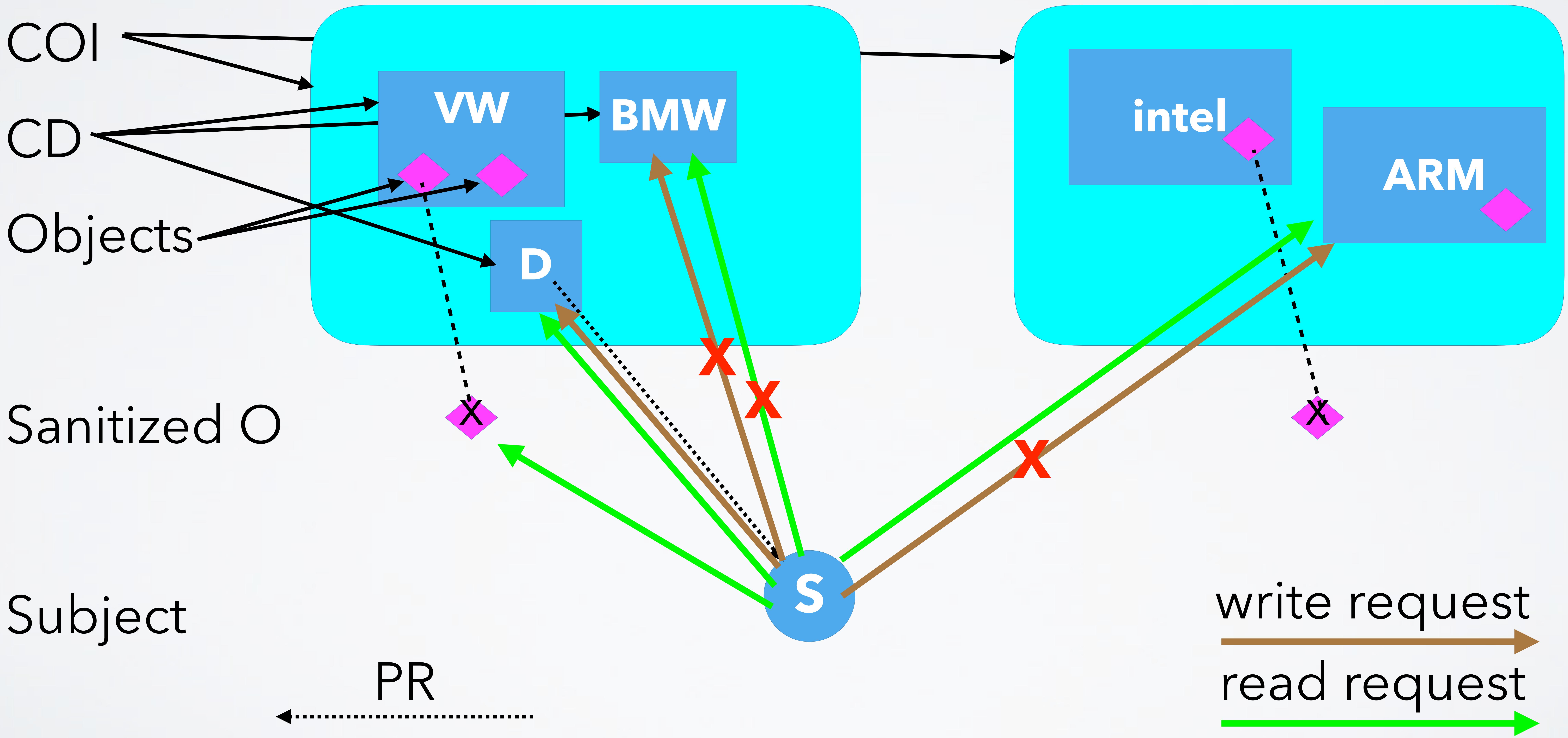
CHINESE WALL, RULES

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:

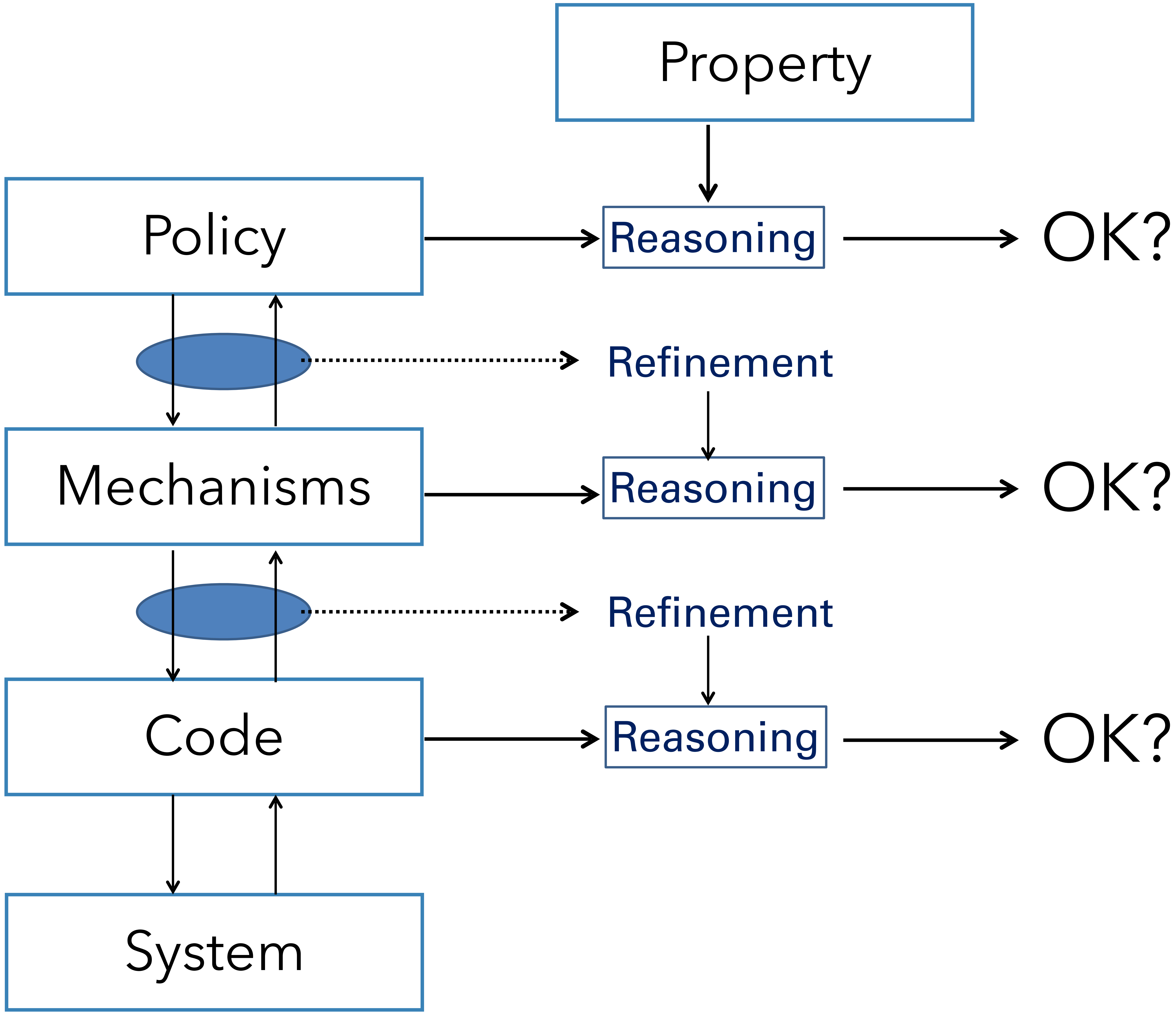
- Access Control List
- Capabilities

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

Simple ACM Operations:

create subject / object

destroy subject / object

enter / delete R into cell (s,o)

	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: ACM & "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: ACM & "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: ACM & "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: ACM & "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"

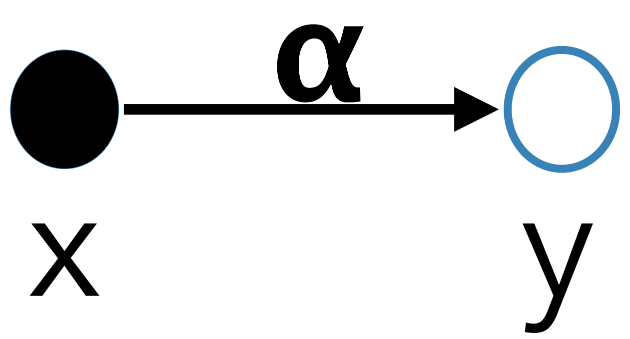
<u>Directed Graph:</u>

Subjects: ●

Objects: ○

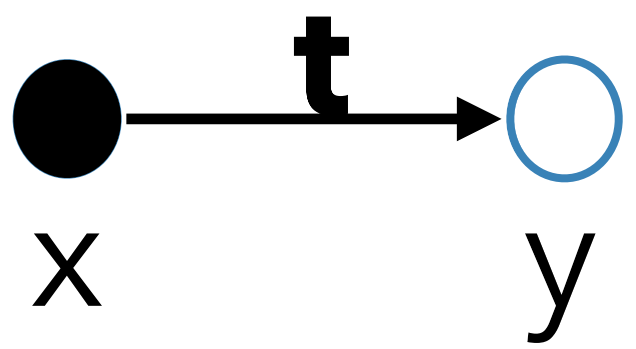
Either S or O: ⊗

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



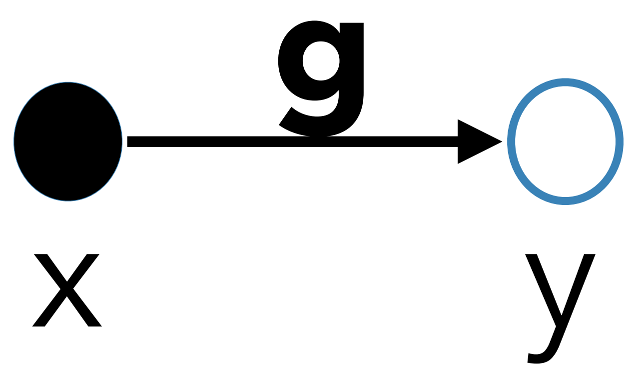
t <u>take right</u>

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**

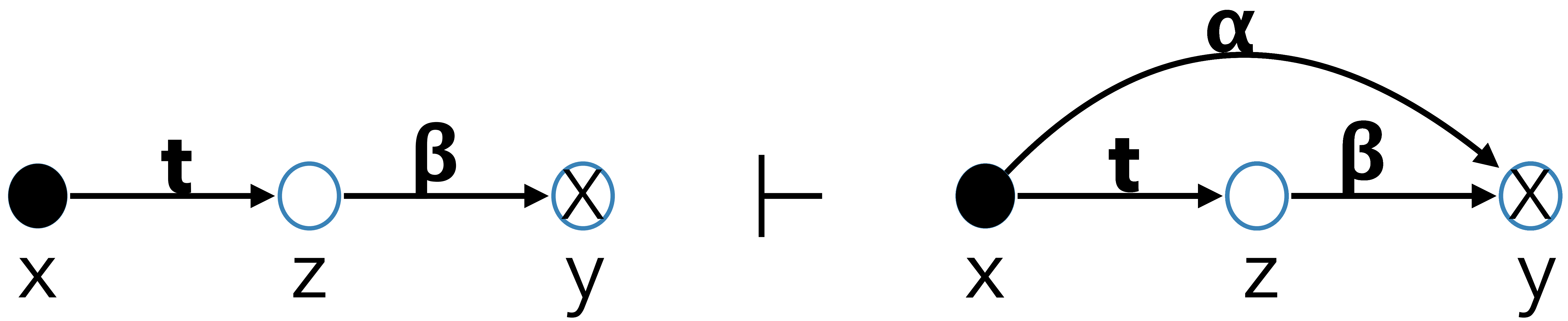


Q3/ 2: TAKE GRANT RULES

Rules:

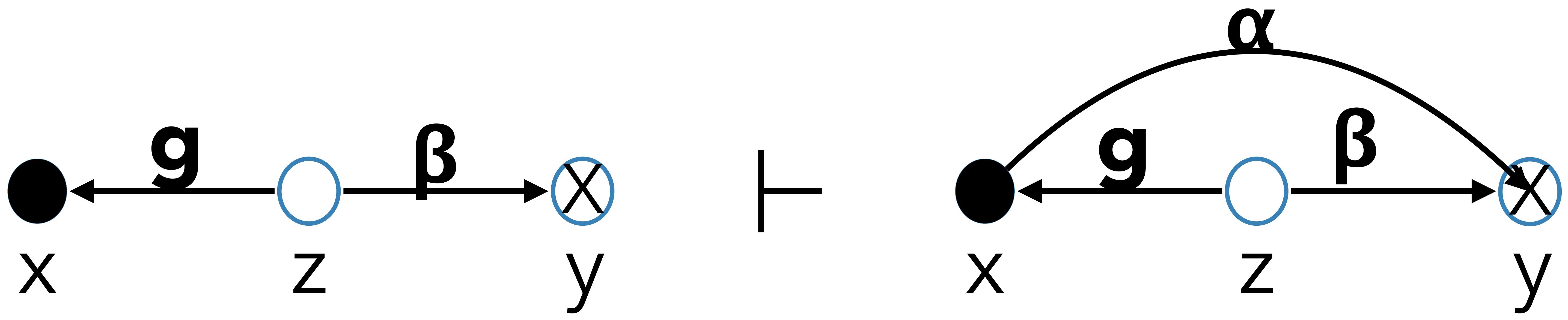
take rule (**α**⊆**β**)

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

<u>Rules:</u>

create rule

$$x \quad \vdash \quad x \xrightarrow{\alpha} y$$

x create (**α** to new vertex) y

remove rule

$$x \xrightarrow{\beta} y \quad \vdash \quad x \xrightarrow{\beta - \alpha} y$$

x removes (**α** to) y

Application of rules $\vdash^*$ creates sequences of Graphs G_i

ref MB: chapter 3.3

Q3/M2: FORMALIZED

<u>CanShare($\boldsymbol{\alpha}$, x, y, G_0):</u>

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

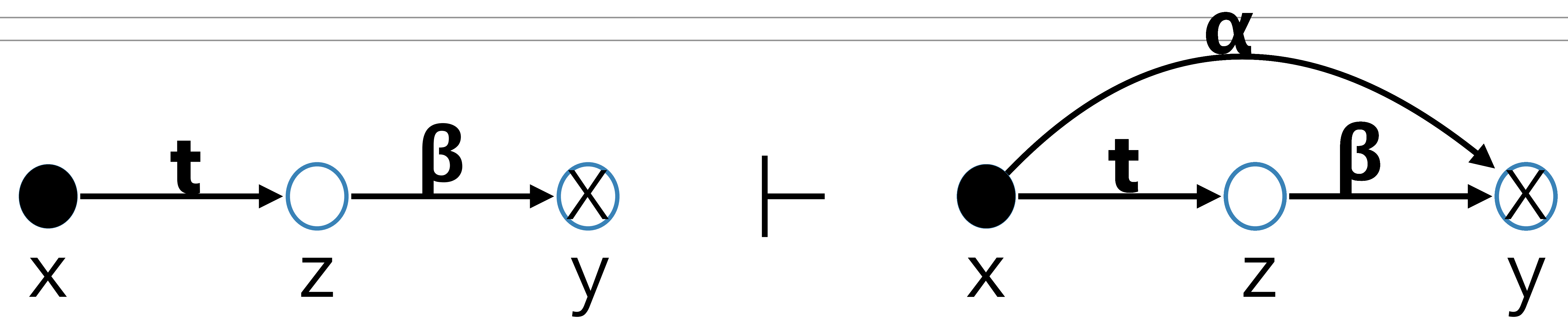
and there is an edge in G_n: $x \xrightarrow{\alpha} y$

ref MB: chapter 3.3

Q3/ 2: CAREFUL: LEMMA

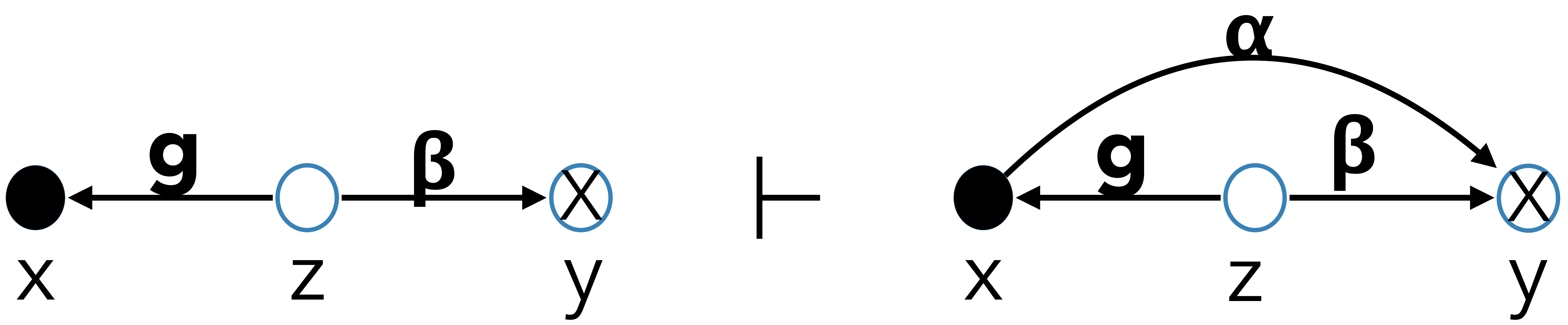
take rule ($\alpha \subseteq \beta$)

a takes (α to y) from z

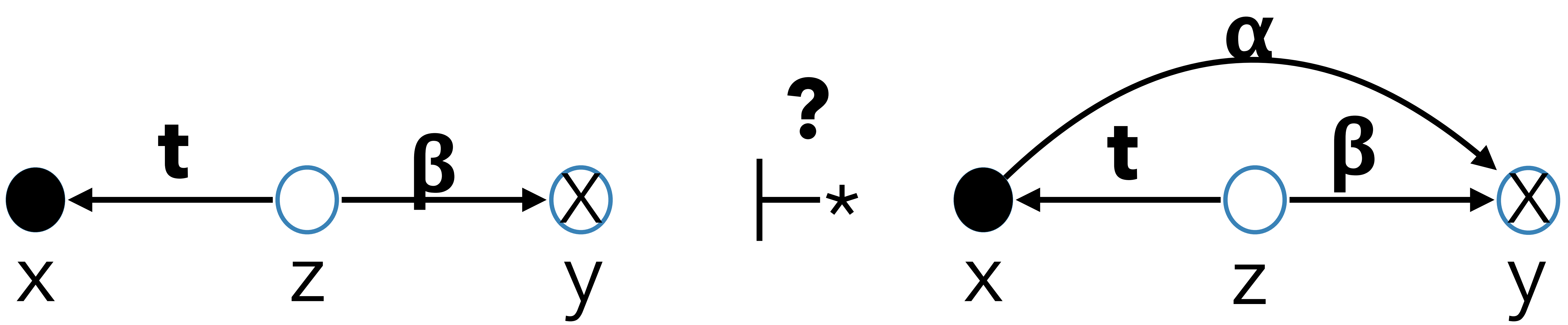


grant rule ($\alpha \subseteq \beta$)

z grants (α to y) to x



Question:

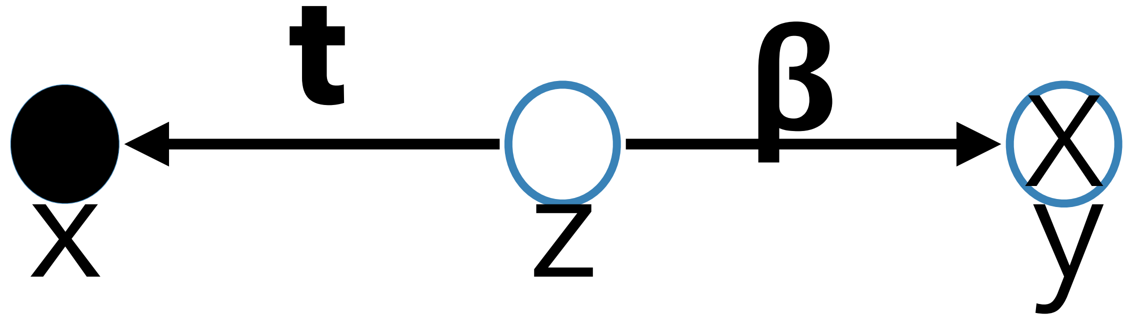


ref MB: chapter 3.3

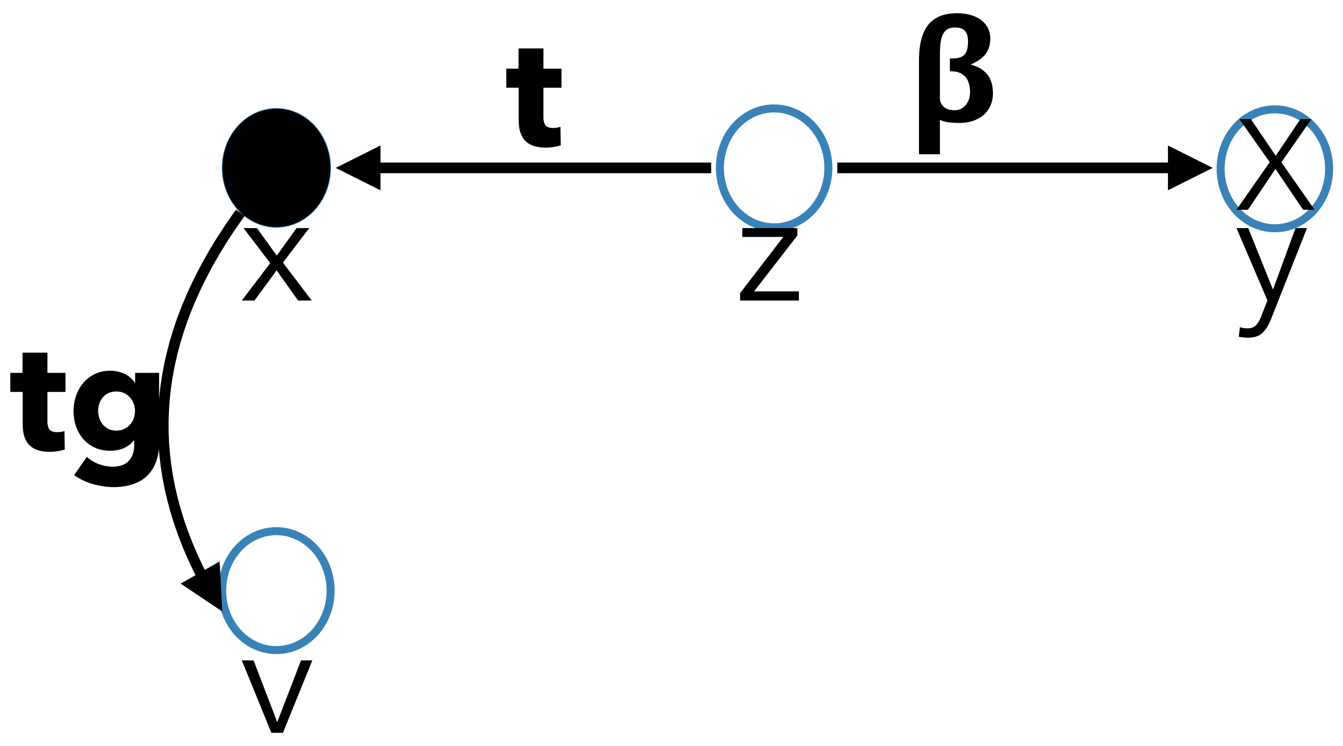
Q3/ 2: CAREFUL: LEMMA

<u>($\alpha \subseteq \beta$)</u>

create rule



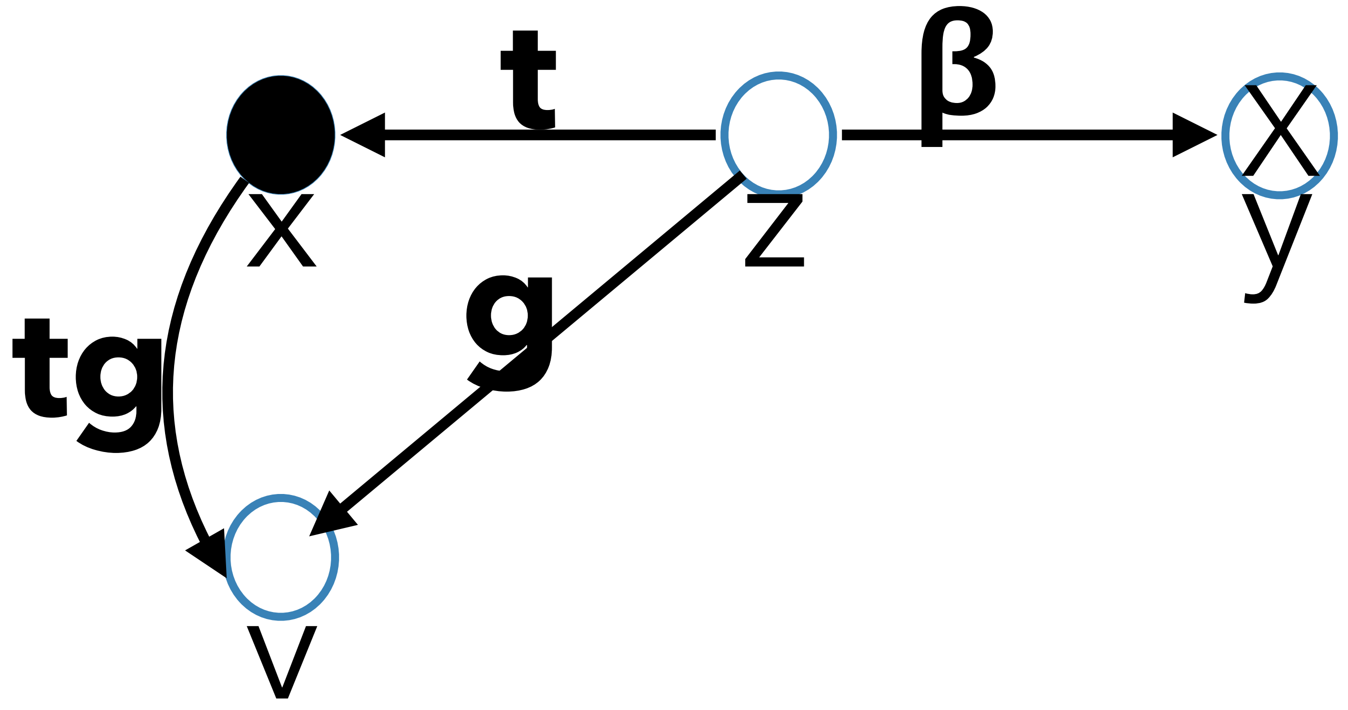
$\vdash$



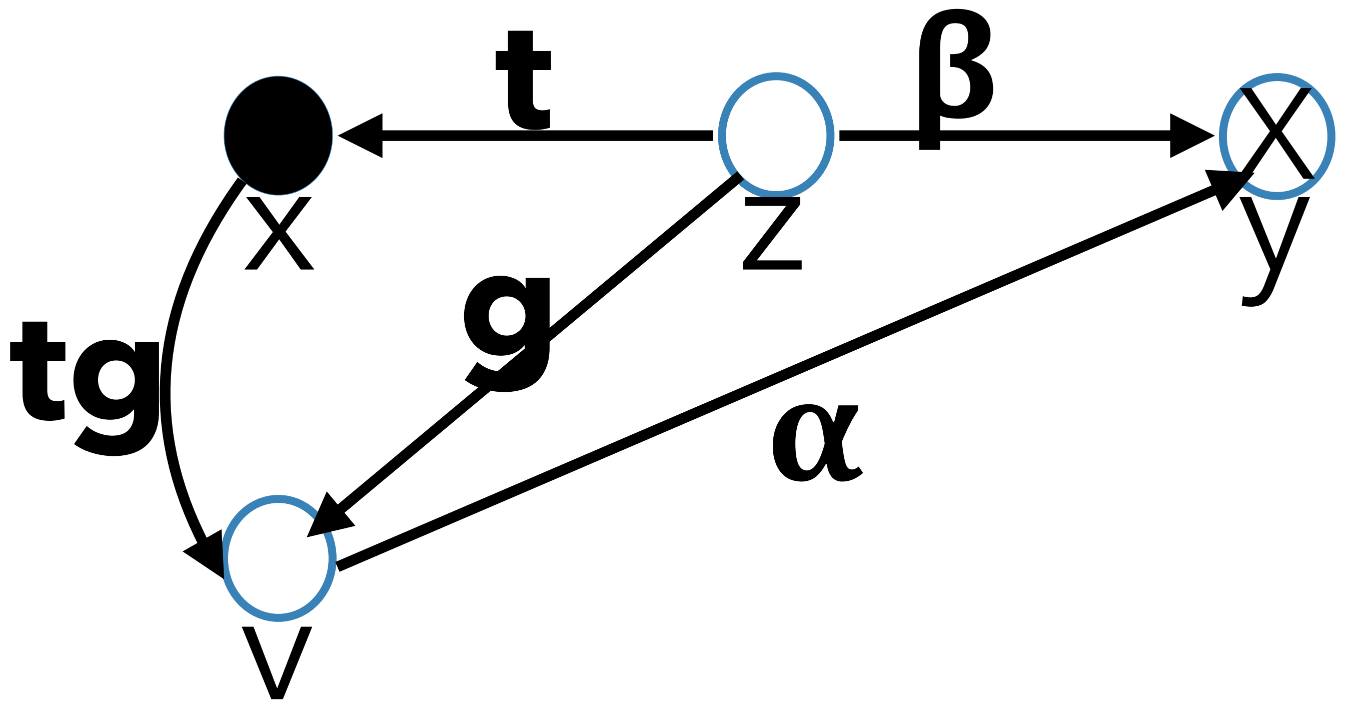
z takes (g to v) from x

z grants (**α** to y) to v

$\vdash$

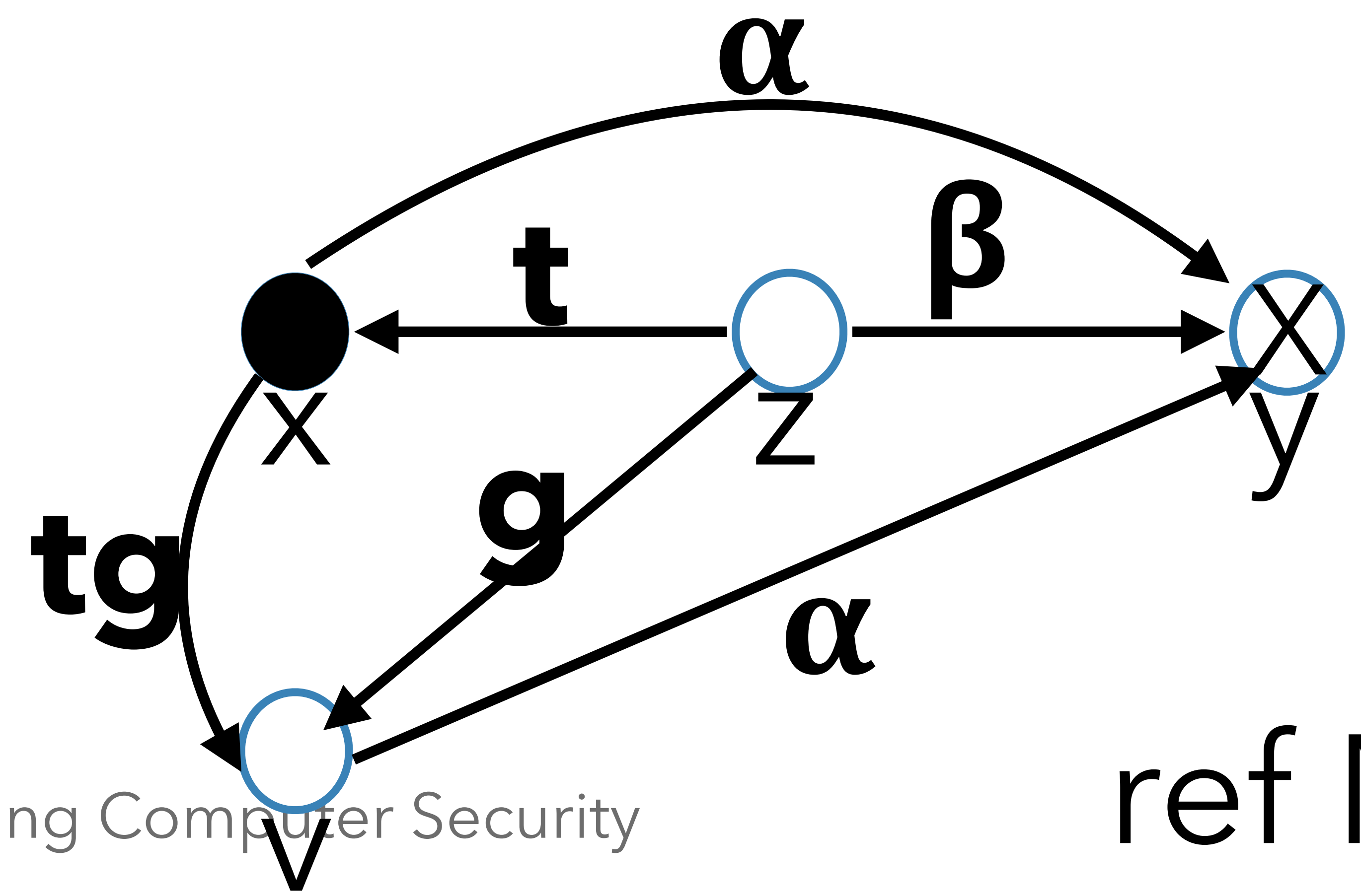


$\vdash$



x takes (**α** to y) from v

$\vdash$



ref MB: chapter 3.3

Q3/M2: FORMALIZED

CanShare($\boldsymbol{\alpha}$, x, y, G_0):

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