Exams: July 17, July 19, September 3,

watch out for
“Systems Programming Lab” in Fall !!!
MODELING DISTRIBUTED SYSTEMS

HERMANN HÄRTIG, DISTRIBUTED OPERATING SYSTEMS, SS2019
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
- you’ll see many models in “Real-Time Systems” class
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 !!!
Reasoning:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics
Reasoning:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics

“Refinement”:
- Abstraction
- Implementation
- Formal Refinement
## Model Examples in General

<table>
<thead>
<tr>
<th>Model</th>
<th>Objective/Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure Trees</td>
<td>are all failure combinations taken into account</td>
</tr>
<tr>
<td>statics models</td>
<td>does a house eventually fall down</td>
</tr>
<tr>
<td></td>
<td>what kind of vehicles on a bridge</td>
</tr>
<tr>
<td>control laws</td>
<td>stability of controllers</td>
</tr>
<tr>
<td>Ohm’s Law</td>
<td>behavior of circuits</td>
</tr>
</tbody>
</table>
## Model Examples Computer Science

<table>
<thead>
<tr>
<th>Model</th>
<th>Objective/Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing Machine</td>
<td>Decidability</td>
</tr>
<tr>
<td>Amdahl’s Law</td>
<td>Scalability</td>
</tr>
<tr>
<td>Logic</td>
<td>Correctness, Precision, ...</td>
</tr>
<tr>
<td>Real-Time “tasks”</td>
<td>can all timing requirements be met</td>
</tr>
<tr>
<td>Byzantine Agreement Two Army</td>
<td>Consensus</td>
</tr>
</tbody>
</table>
Objective of lecture:
understand the power of models and the need for their careful understanding

Intuition, No proofs
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 !!!
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 OF FT

- Fault detection and confinement
- Recovery
- Repair
  - Redundancy
    - Information
    - time
    - structural
    - functional
John v. Neumann
Voter: *single point of failure*

Can we do better
→ distributed solutions?
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems
(Pfitzmann/Härtig 1982)
Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)
Serial-Systems

Each component must work for the whole system to work.

\[ R_{\text{whole}} = \prod_{j=1}^{m} R_j \]
Parallel-Systems

One component must work for the whole system to work.
Each component must fail for the whole system to fail.

\[ R_{\text{whole}} = 1 - \prod_{i=1}^{m} (1 - R_i) \]
Serial-Parallel-Systems

\[
R_{\text{whole}} = 1 - \prod_{j=1}^{m} \left( 1 - \prod_{i=1}^{n} R_{i,j} \right)
\]
Parallel-Serial-Systems

(Pfitzmann/Härtig 1982)
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

Computer 1

Bus 1

CC 1,1

BC 1,1
Q1/MODEL1: CONCRETE MODEL

1 Buses
2 Computers

Computer 1

Computer 2

Bus 1

BC 1,1
BC 2,1
CC 1,1
CC 2,1
Q1/Model 1: Concrete Model

1 Buses
N Computers

Bus 1

Computer 1

Computer 2

Computer n
Q1/MODEL1: CONCRETE MODEL

M Buses
1 Computers

Bus 1
- BC 1,1
- CC 1,1

Computer 1
- CC 1,2
- CC 1,m

Bus m
- BC 1,m
Q1/MODEL1: CONCRETE MODEL

M Buses
N Computers

Computer 1
- CC1,1
- CC1,2
- CC1,m

Computer 2
- CC2,1
- CC2,2
- CC2,m

Computer n
- CCn,1
- CCn,2
- CCn,m

Bus 1
- BC1,1
- BC2,1
- Bn,1

Bus m
- BC1,m
- BC2,m
- Bn,m
Q1/MODEL1: CONCRETE MODEL FOR N,M

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

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

Commander

Lieutenant

Lieutenant

he said: retreat

attack

attack
3 Processes: 1 traitor, 2 loyals

=> 3 processes not sufficient to tolerate 1 traitor
4 Processes

Q2/MODEL 2: “BYZANTINE AGREEMENT”

- Commander
- Lieutenant 1
- Lieutenant
- Lieutenant

He said:
- Attack
- Retreat

He said:
- Attack
4 Processes

**General result:** 3 f + 1 processes needed to tolerate f traitors
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?
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 2019)
Reasoning:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics
- “Refinement”:
  - Abstraction
  - Implementation
  - Formal Refinement
“Reasoning”:
- Common sense
- Formal Verification
- Careful Inspection
- Mathematics
- “Common Criteria Assurance”

“Refinement”:
- ...

SECURITY MODELS

TU Dresden: Hermann Härtig, Marcus Völp
Modeling Computer Security, SS 2019
Definition: Policy

Examples:

Higher-Level Policies (very short):
- Bell La Padula
- Chinese Wall
Operating Sys. Mechanisms:
- Access Control List
- Capabilities

Explain Q3 and formalize (2 models)!

Models:
- based on Access Control Matrix
- “take grant” model
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., $\Sigma_{\text{sec}} := \{ \sigma \in \Sigma \mid P(\sigma) \}$) 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., $\Sigma_{\text{reachable}} \subseteq \Sigma_{\text{sec}}$)

Reference: Matt Bishop: Computer Security Art and Science
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.
Model for Confidentiality

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

Categories

TU Dresden: Hermann Härtig, Marcus Völpin
Modeling Computer Security, SS 2019
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
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

COI

CD

Objects

Sanitized O

Subject

VW → BMW

intel → ARM

x

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

S can read O, if any of the following holds

- first-time read
- There is an object O’ such that S has accessed O’ and CD(O) = CD(O’)
- ∀ O’: O’ ∈ PR(S) => COI(O) ≠ COI(O’)
- O is a sanitized Object
PR(S): set of Objects read by S

S can write O, if both both of the following hold

- S is permitted to read O
- ∀ unsanitized O’, S is permitted to read O’ => CD(O) = CD(O’)

CHINESE WALL, EXAMPLE

COI
CD
Objects
Sanitized O
Subject

VW → BMW

Subject (S)

write request
read request

PR
Operating Sys. Mechanisms:

- Access Control List
- Capabilities

Explain Q3 and formalize per model!

Models:

- *based on Access Control Matrix*
- "take grant" model
Subjects: $S$
Objects: $O$
Entities: $E = S \cup O$
Rights: \{read, write, own,\ldots\}
Matrix: $S \times E \times R$

**Simple ACM Operations:**
- create subject / object
- destroy subject / object
- enter / delete $R$ into cell $(s,o)$

### Mechatisms: Access Control Matrix

<table>
<thead>
<tr>
<th></th>
<th>O1</th>
<th>O2</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S1$</td>
<td>r,w,own</td>
<td>r,w</td>
<td>r,w,own</td>
<td>--</td>
<td>r,w</td>
</tr>
<tr>
<td>$S2$</td>
<td>r,w</td>
<td>r,w,own</td>
<td>--</td>
<td>r,w,own</td>
<td>$r$</td>
</tr>
<tr>
<td>$S3$</td>
<td>r,w</td>
<td>$r$</td>
<td>$w$</td>
<td>--</td>
<td>r,w,own</td>
</tr>
</tbody>
</table>

ref MB: chapter 2.2
### ACM

- **Access Control List (ACL)**
- **Capabilities**

#### ref MB: chapter 2.2
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
“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?
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)

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>r,w,own</td>
<td>r,w</td>
<td>r,w</td>
</tr>
<tr>
<td>O2</td>
<td>r,w</td>
<td>r,w,own</td>
<td>r,w,own</td>
</tr>
<tr>
<td>S1</td>
<td>r,w</td>
<td>r,w</td>
<td>r,w</td>
</tr>
<tr>
<td>S2</td>
<td>r,w,own</td>
<td>r,w,own</td>
<td>r,w,own</td>
</tr>
<tr>
<td>S3</td>
<td>r,w</td>
<td>r</td>
<td>w</td>
</tr>
</tbody>
</table>

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)

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
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
“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

Alice → Bob
Bob → Carol
Carol → Alice

Botschaft
Q3/MODEL 2: “TAKE GRANT”

Directed Graph:

Subjects: ●

Objects: ○

Either S or O: ✗

x has capability with set of rights $\alpha$ on y:

$\exists \alpha \in \mathcal{X}$

x take right $t$

x has cap with set of rights $\tau$ that includes $t$

x grant right $g$

x has cap with set of rights $\gamma$ that includes $g$
Q3/ 2: TAKE GRANT RULES

Rules:

take rule (α ⊆ β)

a takes (α to y) from z

grant rule (α ⊆ β)

ref MB: chapter 3.3
Q3/ 2: TAKE GRANT RULES

Rules:

create rule

\[ \text{x create (} \alpha \text{ to new vertex)} \text{ y} \]

remove rule

\[ \text{x removes (} \alpha \text{ to)} \text{ y} \]
CanShare(\(\alpha, x, y, G_0\)):

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

and there is an edge in \(G_n\):

\[ x \xrightarrow{\alpha} y \]
take rule ($\alpha \subset \beta$)

a takes ($\alpha$ to $y$) from $z$

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

$z$ grants ($\alpha$ to $y$) to

Question:

ref MB: chapter 3.3
create rule

z takes (g to v) from x

z grants (α to y) to v

ref MB: chapter 3.3
CanShare(\(\alpha, x, y, G_0\)):

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

and there is an edge:

\[\begin{array}{c}
\bullet \\
\alpha \\
x \\
\rightarrow \\
\circ \\
y
\end{array}\]

CanShare decidable in linear time!
- three questions, 2 models per question, different answers !!!
- modeling is powerful
- need to look extremely carefully into understanding models !!!
REFERENCES

- **Q1/M1:**

- **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