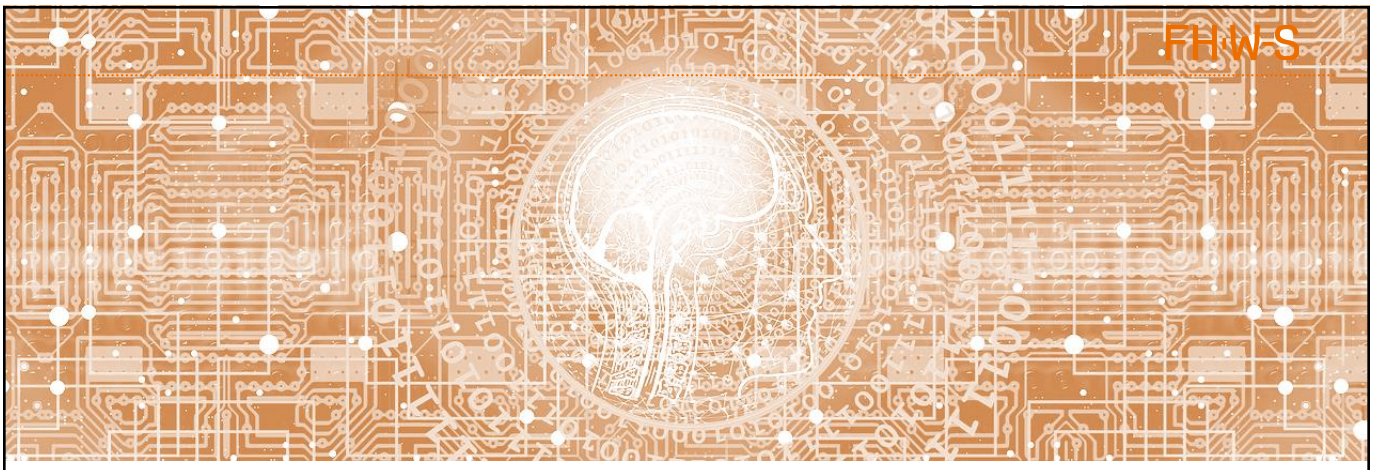


Embedded Systems and Field Buses

Prof. Dr. Marian Daun

FH·W·S University of Applied Sciences Würzburg-Schweinfurt

Group Discussion

Expectations

What this course is about and what it is not

- Fundamentals
- Structure of Embedded Systems
- Behavior of Embedded Systems
- Design of Embedded Systems
- Communication
- Real-time
- Collaborative Embedded Systems

FH·W-S

Embedded Systems Fundamentals

Terminology

Group Discussion

What is an Embedded System

Exercise

Define and differentiate from Embedded Systems:

- Information Systems
- Cyber-physical Systems
- IOT-Systems
- Software-intensive Systems
- Safety-critical Systems
- Distributed Systems
- Collaborative Systems

Characteristics

Characteristics of Embedded Systems include:

- **Real-time ability** eg: car brake
- **Safety** – physical devices that can harm people

- **Dependability**
- **Reactivity**
- **Efficiency**

- **New: Security**

Safety vs Security

In English often used synonymous but when used in German strictly distinguished!
Also officially distinguished according to ISO/IEC/IEEE 24765:

Safety
- expectation that a system does not, under defined conditions, lead to a state in which human life, health, property, or the environment is endangered

Safety vs Security

Security (according to ISO/IEC/IEEE 24765)

- protection against intentional subversion or forced failure
- defining, achieving, and maintaining confidentiality, integrity, availability, non-repudiation, accountability, authenticity, and reliability of a system
- degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization
- protection of computer hardware or software from accidental or malicious access, use, modification, destruction, or disclosure
- protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them
- protection against intentional subversion or forced failure, containing a composite of four attributes: confidentiality, integrity, availability and accountability, plus aspects of a fifth, usability, all of which have the related issue of their assurance

Group Discussion

How do Safety and Security influence each other?

Dependability

Typically consists (at least) of:

- **Availability**
  - degree to which a system or component is operational and accessible when required for use
- **Reliability**
  - ability of a system or component to perform its required functions under stated conditions for a specified period of time
- **Maintainability**
  - ease with which a software system or component can be modified to change or add capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment
  - ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions
  - …

[ISO/IEC/IEEE 24765]

Real-Time

1. problem, system, or application that is concurrent and has timing constraints whereby incoming events must be processed within a given timeframe
2. pertaining to a system or mode of operation in which computation is performed during the actual time that an external process occurs, in order that the computation results can be used to control, monitor, or respond in a timely manner to the external process

[ISO/IEC/IEEE 24765]

Reactivity and Efficiency

Reactivity

- Stimulus and Response
- Input-driven systems

Efficiency

- degree to which a system or component performs its designated functions with minimum consumption of resources [ISO/IEC/IEEE 24765]
- Hardware is expensive
- Control units are expensive
- Communication is expensive

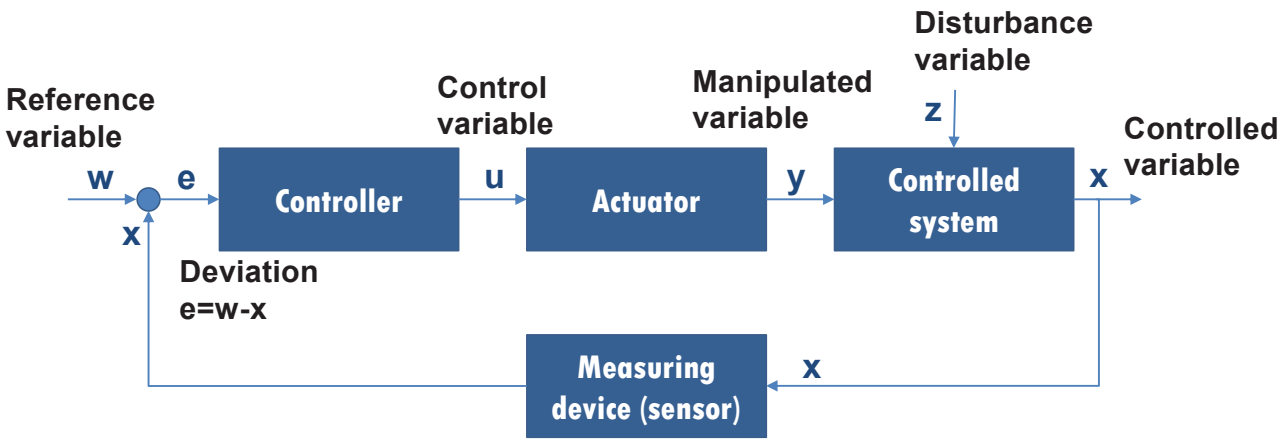
Exercise

Where can these characteristics be found in:

- Aircrafts?
- Cars?
- Smart Factories?
- Autonomous Driving?

Function of Embedded Systems

Control Loop (Steuer- und Regelkreis)

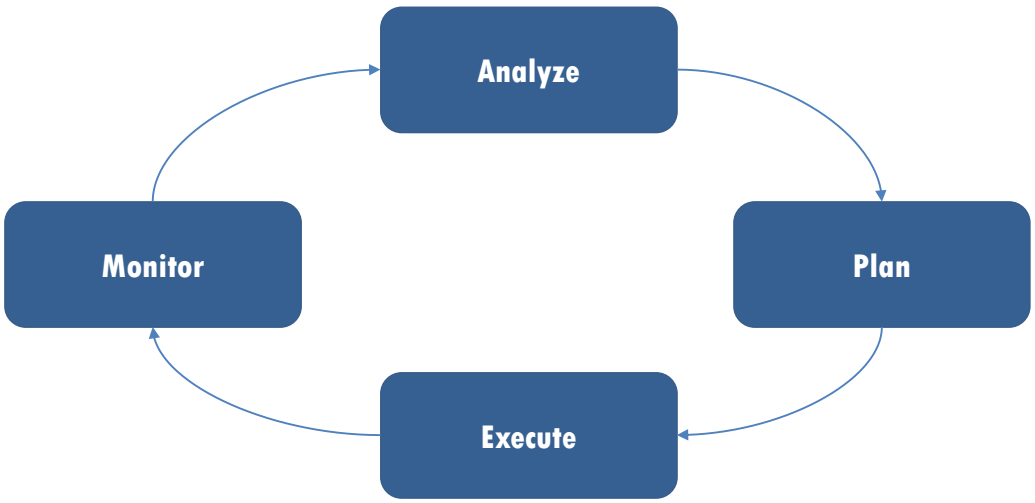


Exercise

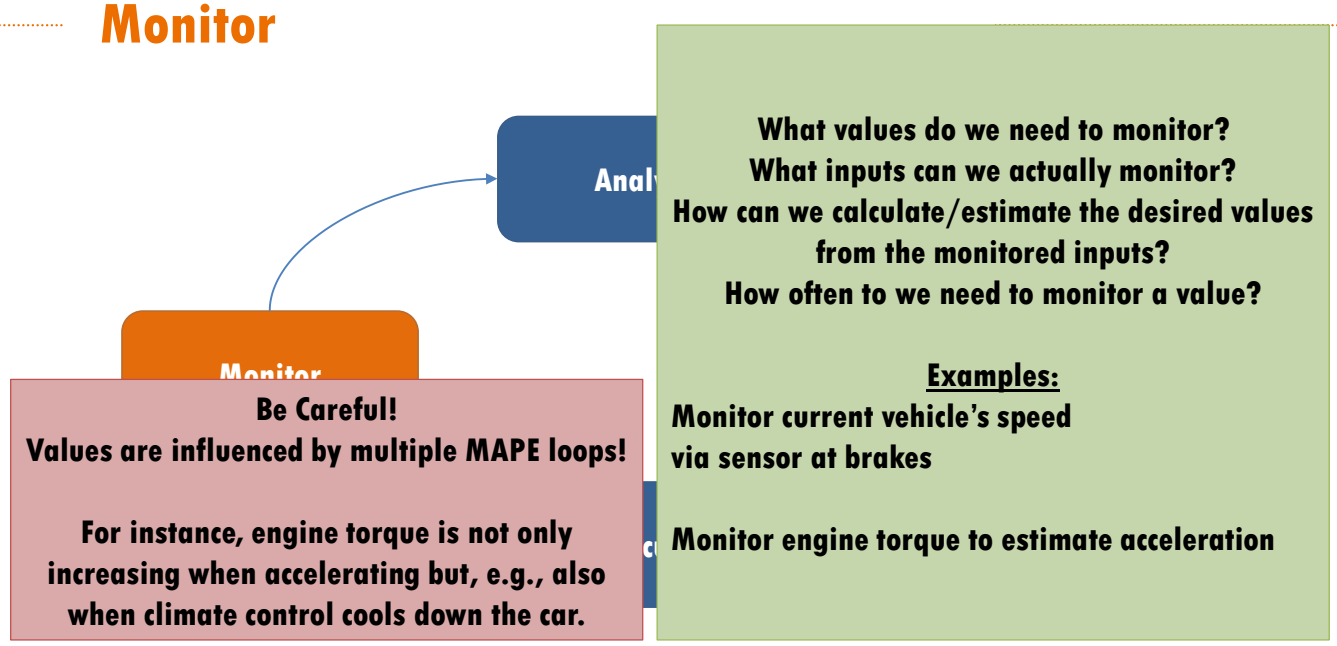
For each system, identify variables to be controlled:

- Aircraft
- Car
- Smart Factory
- Autonomous Driving

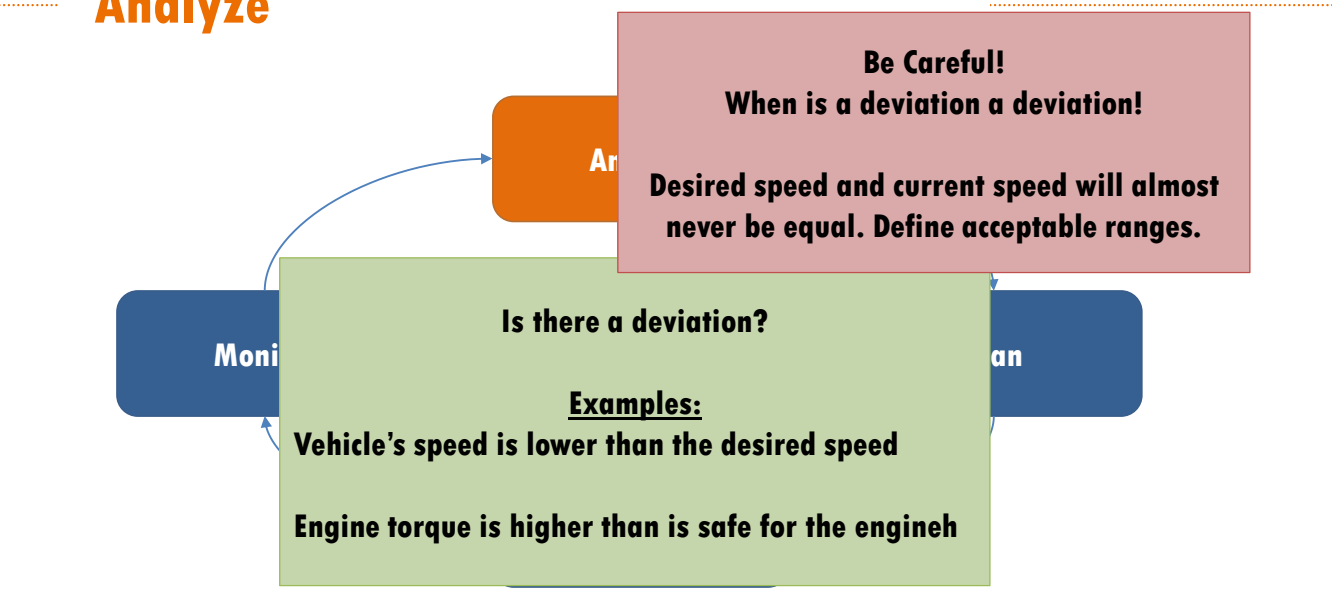
MAPE-Loop



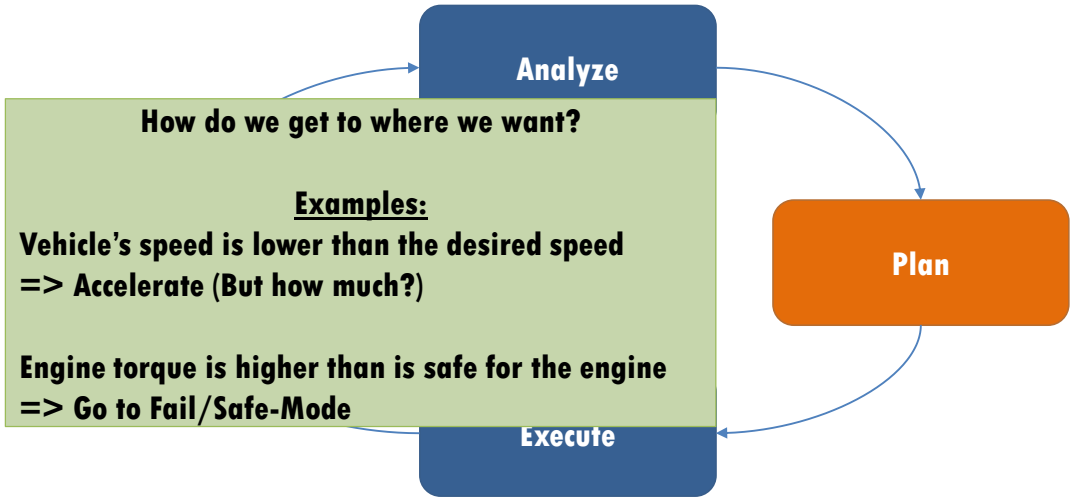
Monitor



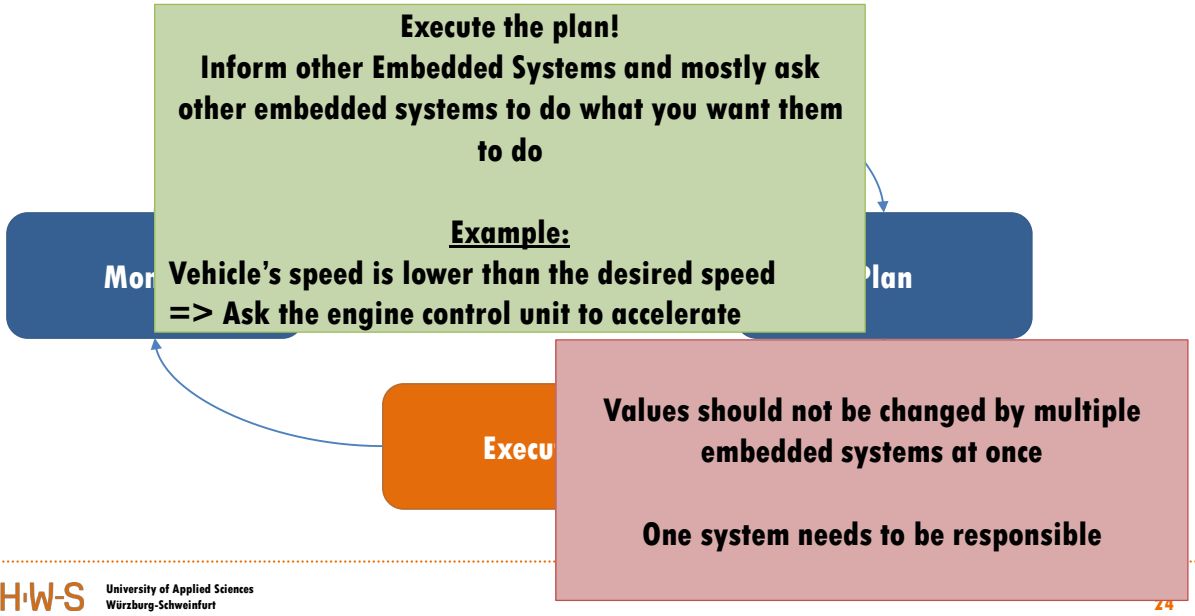
Analyze



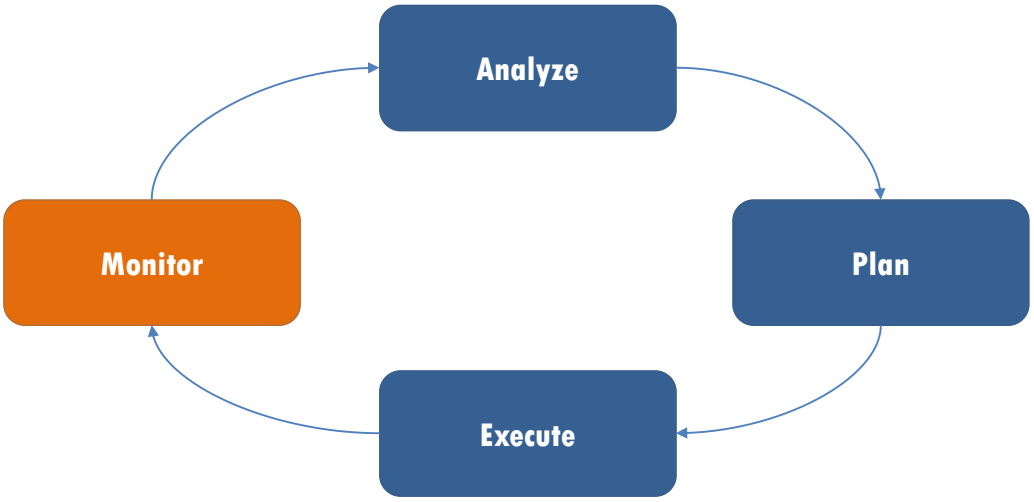
Plan



Execute



And Again: Monitor



Exercise

For each system, define one relevant MAPE-Loop:

- Aircraft
- Car
- Smart Factory
- Autonomous Driving

Monitor
Analyse
Plan
Execute

HW/SW Co-design

Engineers like top-down approaches.

However, this does not work with software.

The development of Software and Hardware components of Embedded Systems needs to be intertwined.

There is a multitude of solutions. We can have hardware solutions, software solutions, or mixtures. Thus, the hardware defines what software we need and the software defines what hardware we need.

Continuous vs Discrete Values

Hardware is continuous

Software is discrete

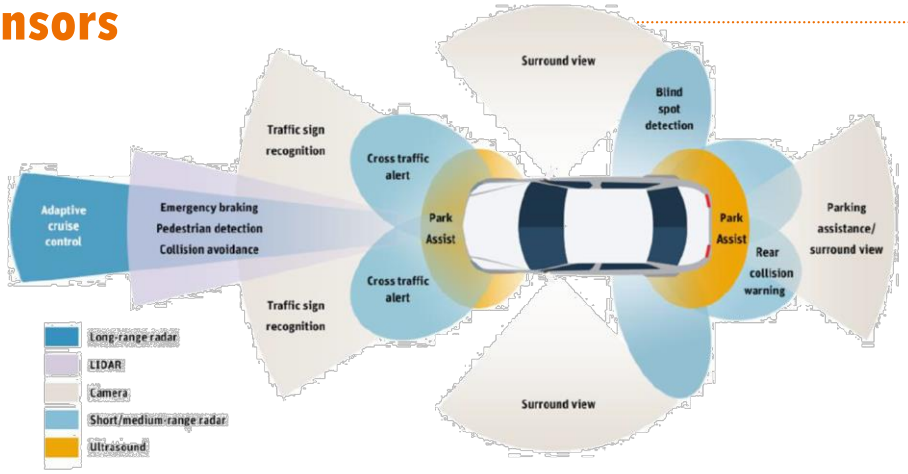
What does this mean?

Sensors

Group Discussion

What sensors do we know?

Exemplary Sensors



https://www.ansys.com/about-ansys/advantage-magazine/volume-xii-issue-1-2018/autonomous-vehicle-radar

Slide taken from Lecture Materials provided by Prof. Dr. Marco Schmidt

Actuators

Group Discussion

What actuators do we know?

Exercise

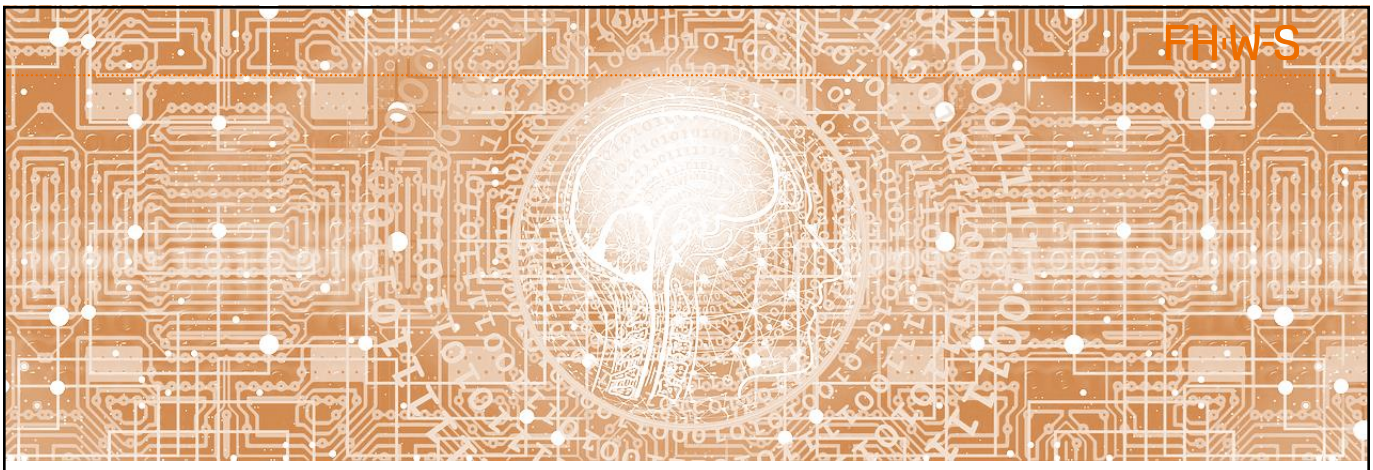
For the MAPE-Loops identified before, define the necessary Sensors and Actuators

Literature

[ISO/IEC/IEEE 24765] Systems and software engineering – Vocabulary. International Standard, ISO/IEC/IEEE, 2017.

Questions for Self-Assessment

- What is an embedded system?
- How do embedded systems differ from other systems?
- What are the main characteristics of embedded systems?
- Where is the difference between safety and security?
- What is the difference between a control loop and a MAPE loop?
- What problems can arise when defining a MAPE loop?
- Why is the development of hardware and software intertwined?



Embedded Systems and Field Buses

Prof. Dr. Marian Daun

FH·W·S University of Applied Sciences Würzburg-Schweinfurt

Agenda

- Fundamentals
- Structure of Embedded Systems
- Behavior of Embedded Systems
- Design of Embedded Systems
- Communication
- Real-time
- Collaborative Embedded Systems

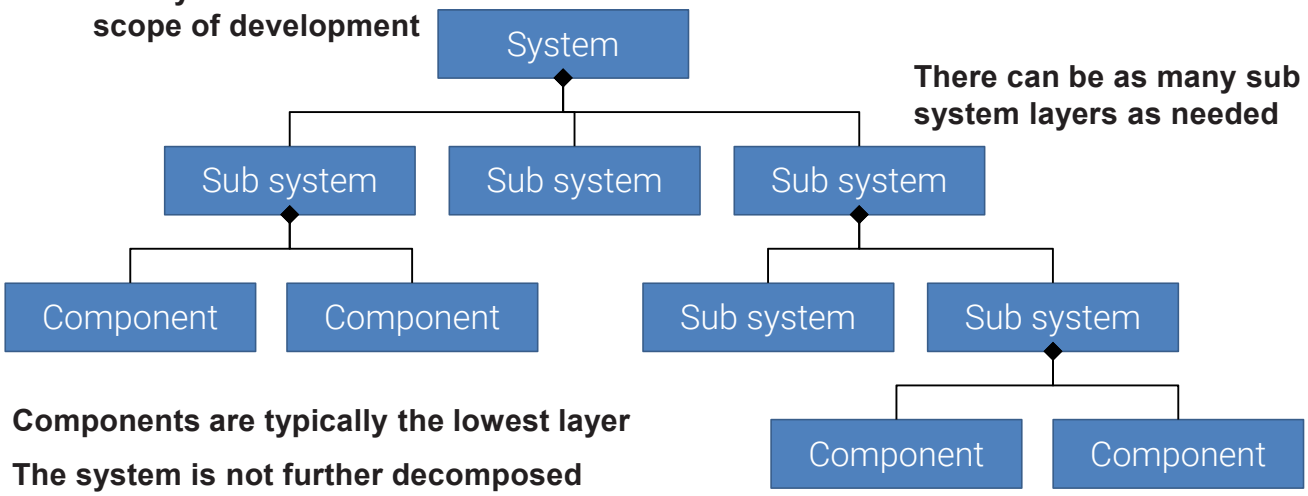
FH·W-S

Structure of Embedded Systems

Hierarchies

System Layers

The system defines the scope of development



There can be as many sub system layers as needed

Components are typically the lowest layer

The system is not further decomposed

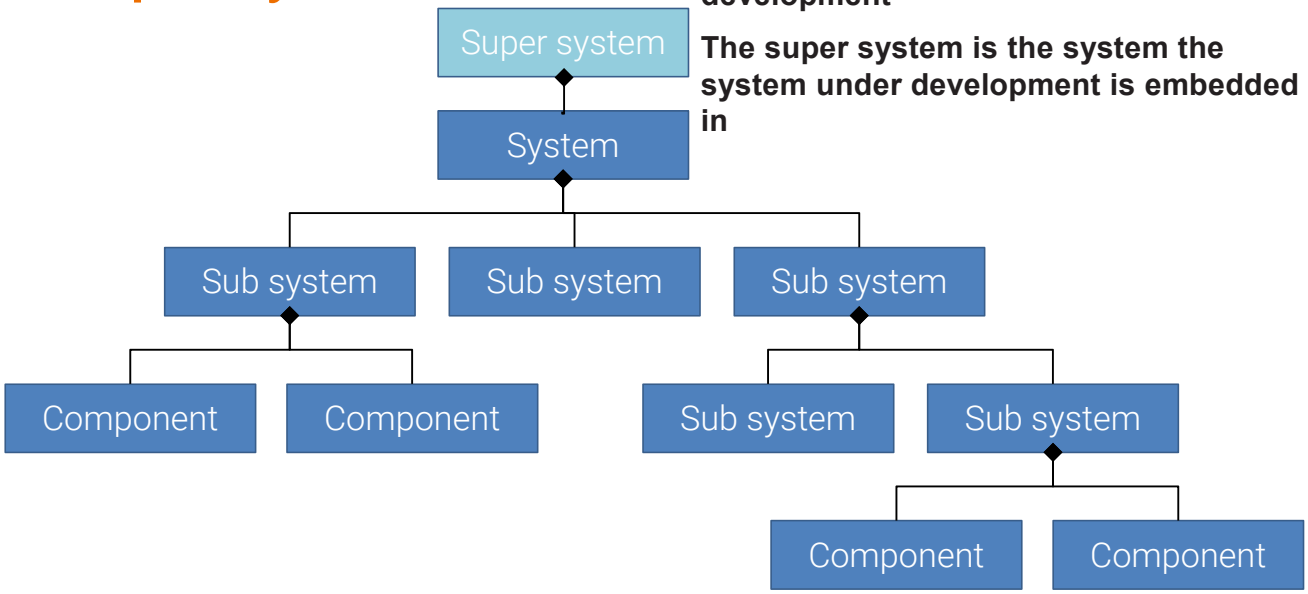
Group Discussion

When has Development reached the Component Layer?

Super System

The super system is not the scope of development

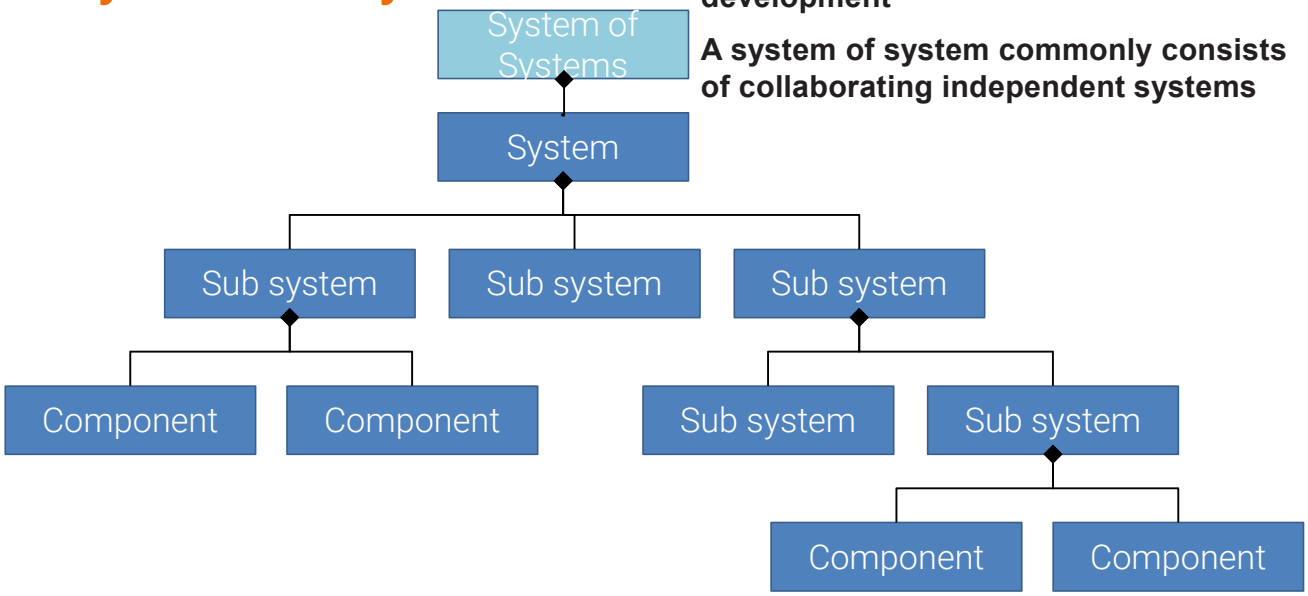
The super system is the system the system under development is embedded in



System of Systems

A system of systems can be the scope of development

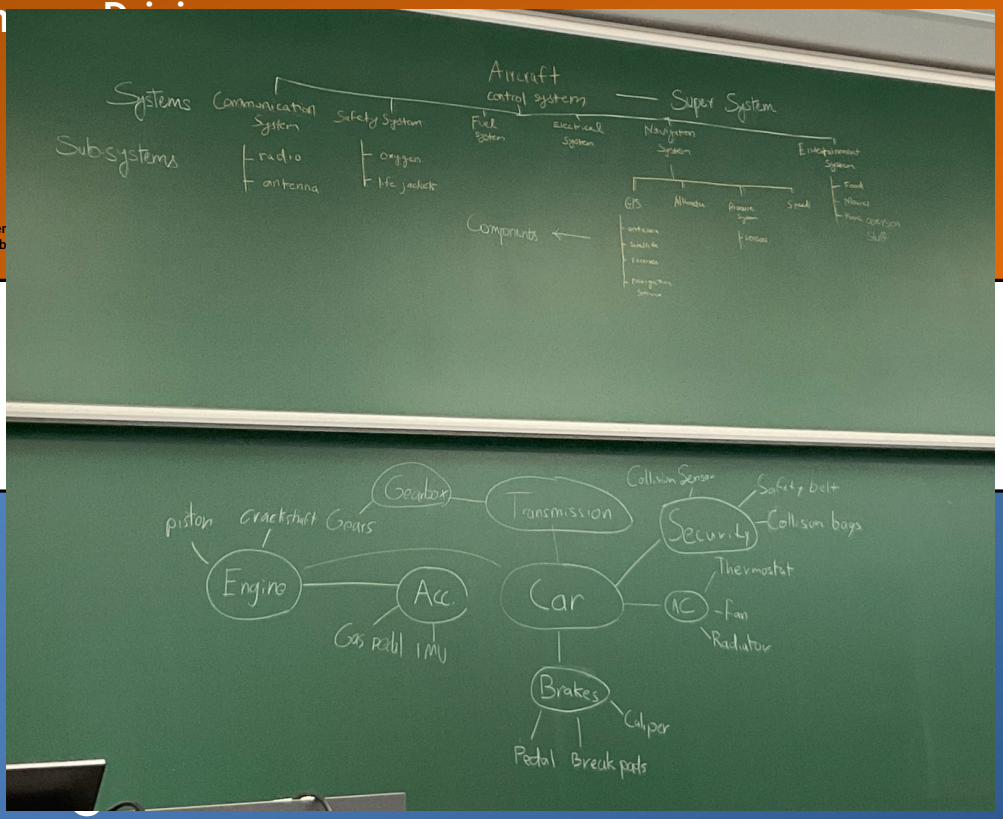
A system of system commonly consists of collaborating independent systems



Exercise

Define the System Layers for:

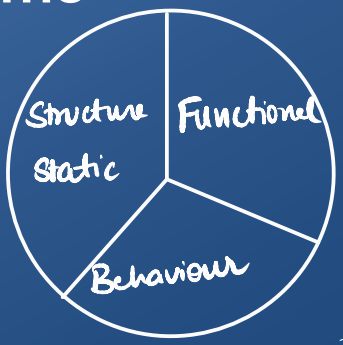
- An Aircrafts
- A Car
- A Smart Factory
- Autonon



FH W-S University of Applied Sciences Würzb

9

Model
SysML Block Definition Diagrams

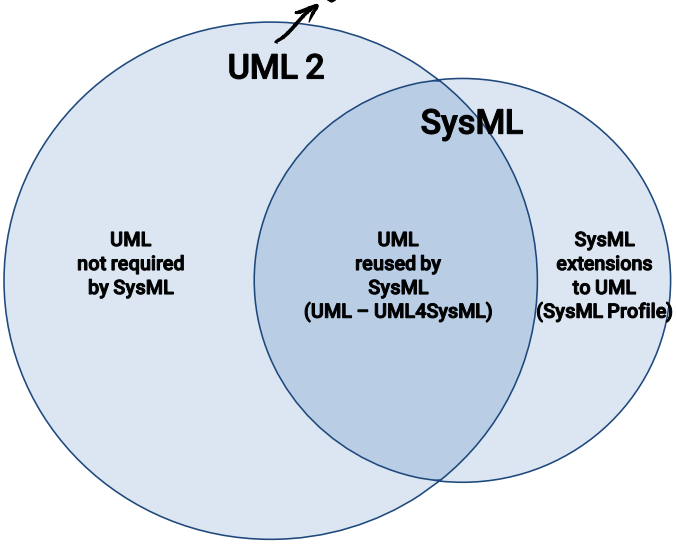


FH W-S University of Applied Sciences Würzburg-Schweinfurt

10

SysML

Unified modelling language

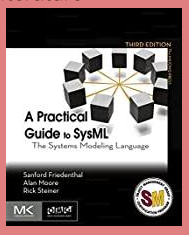


Standards

https://www.omg.org/spec/SysML/

https://www.omg.org/spec/UML/

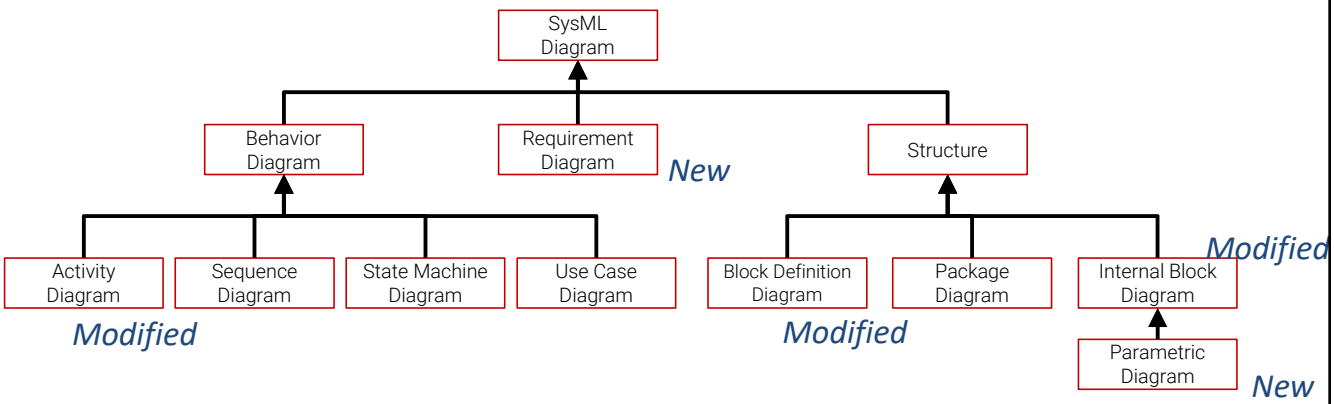
Literature



Sanford Friedenthal, Alan Moore & Rick Steiner: A Practical Guide to SysML – The Systems Modeling Language. *Morgan Kaufmann OMG Press, 2014.*

[OMG Systems Modeling Language v1.6, 2019]

SysML Diagram Taxonomy

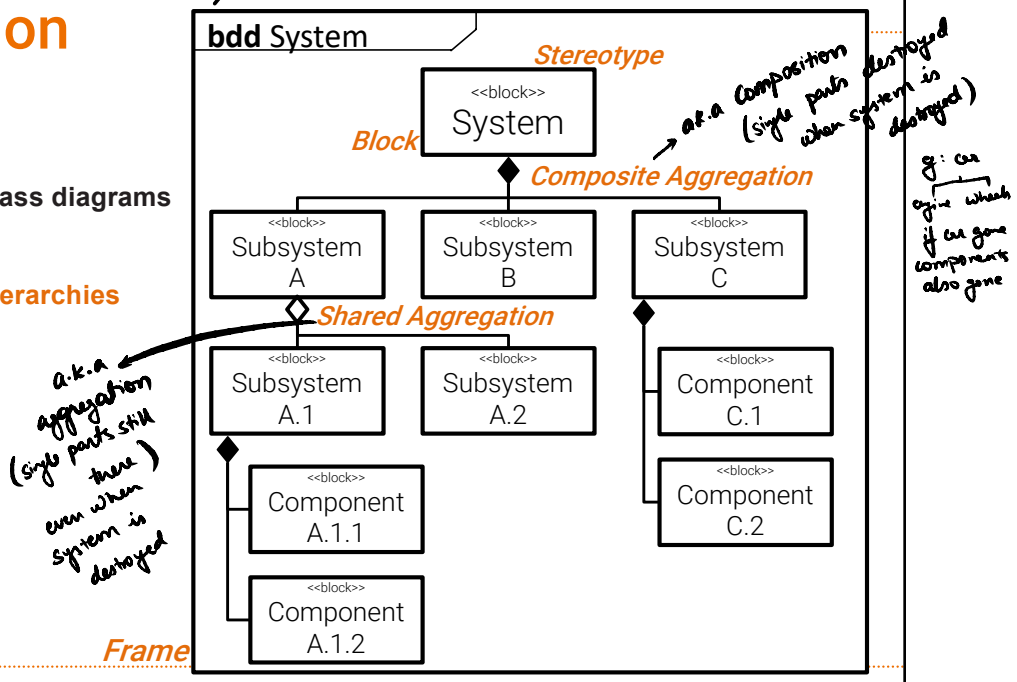


[OMG Systems Modeling Language v1.6, 2019]

Block Definition Diagram

BDDs are based on UML class diagrams

BDDs are used to model hierarchies

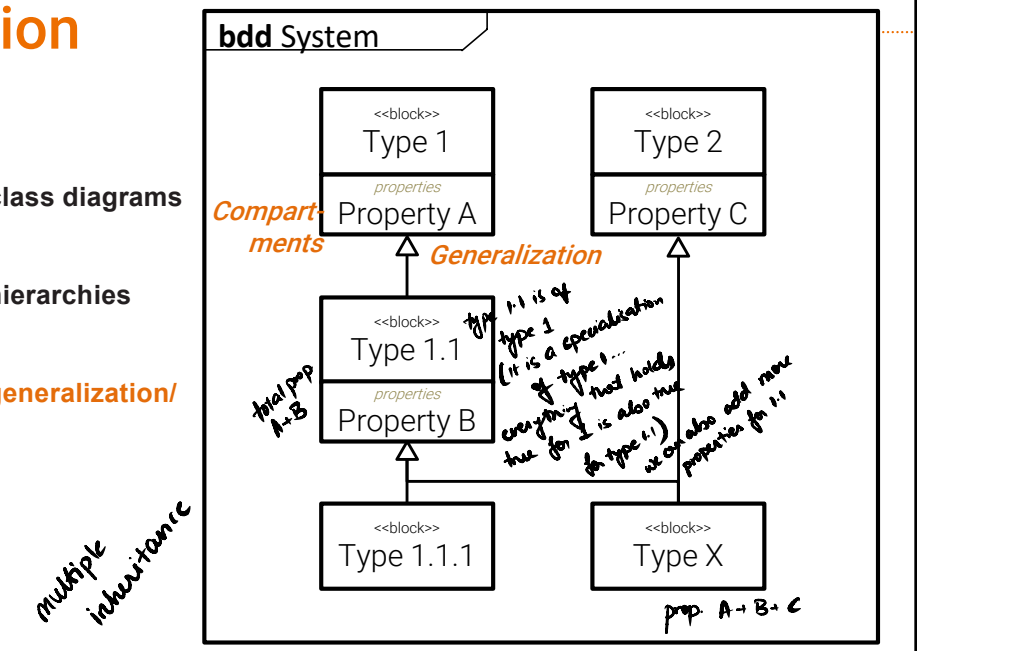


Block Definition Diagram

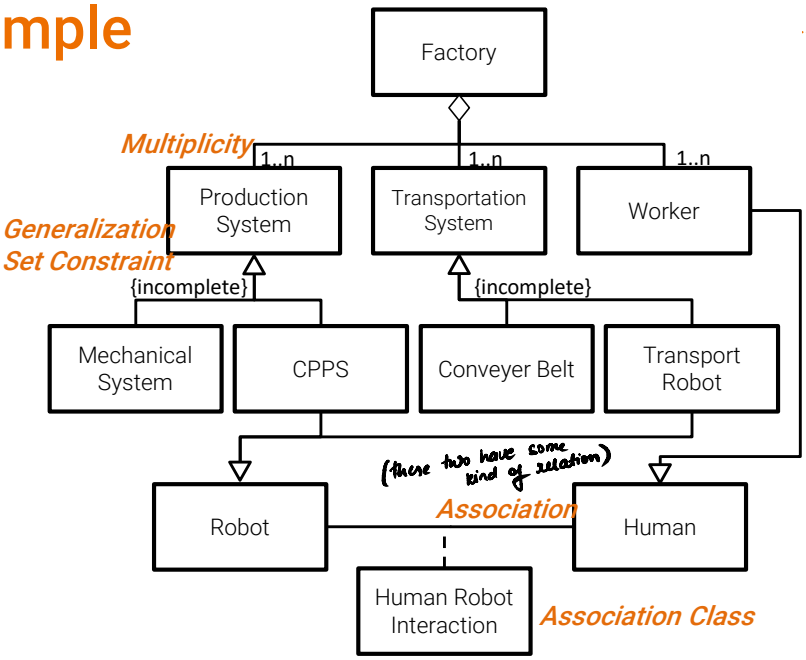
BDDs are based on UML class diagrams

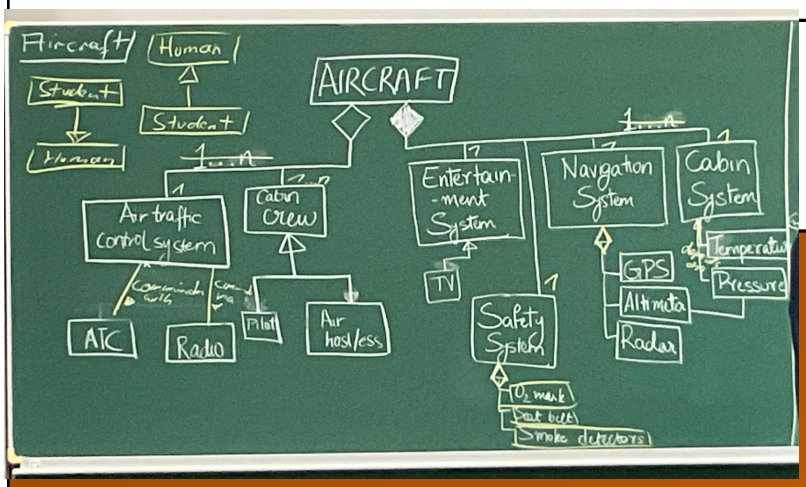
BDDs are used to model hierarchies

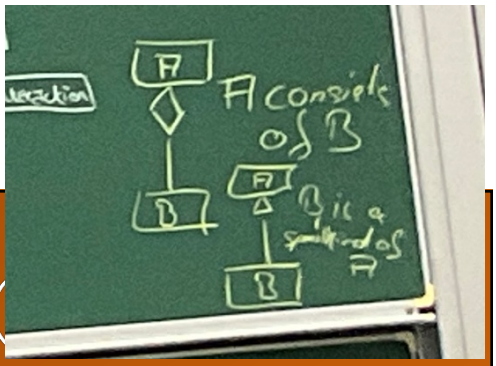
BDDs are used to model generalization/ specialization relations



Example

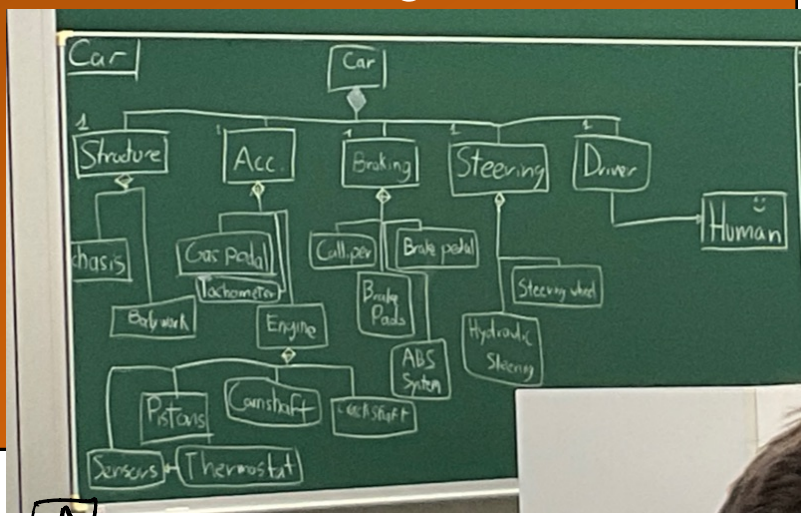




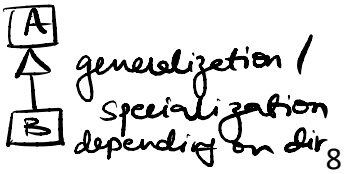


Create SysML Block Definition Diagrams for:

- An Aircrafts
- A Car
- A Smart Factory
- Autonomous Driving



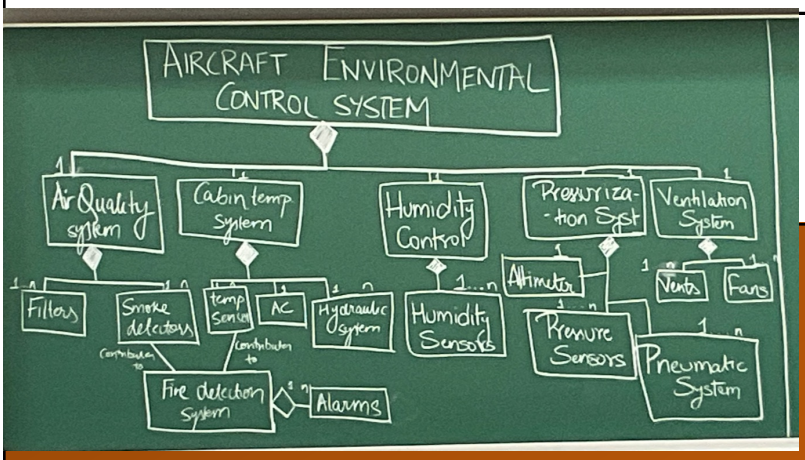
all compositions are aggregations but all aggr. is not composition



Scope of Development

From the point of an original equipment manufacturer a car might be the system under development. However, it is uncommon to refer to a car as the system. Typically, the system under development is a system embedded in the car.

An aircraft, a car, a smart factory are thus super systems.
For autonomous driving a platoon is a system of systems.

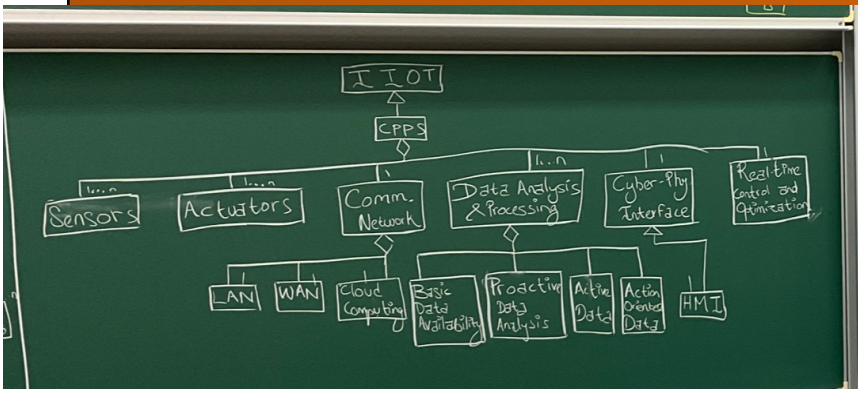


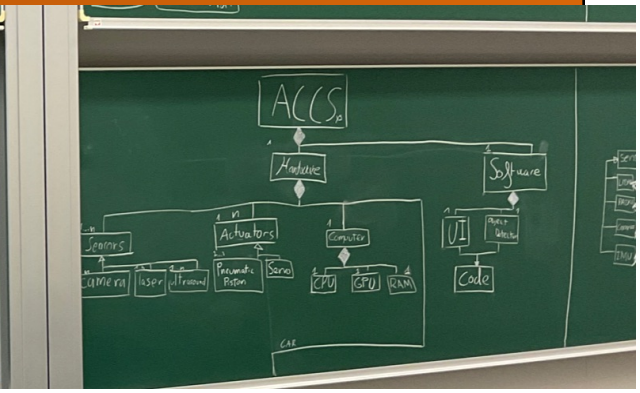
Go through

Exercise

Create SysML Block Definition Diagrams for:

- An embedded system of an Aircraft
- An embedded system of a Car
- A cyber-physical production system within a Smart Factory
- The transport system of a smart factory as a System of Systems





Group Discussion

What do we notice when comparing the model of the super system with the model of the embedded system?

Context

Context is Important

Separation of **Context** and **System** allows to differentiate between the scope of development (i.e. **what can be changed**) and the **constraining** environment

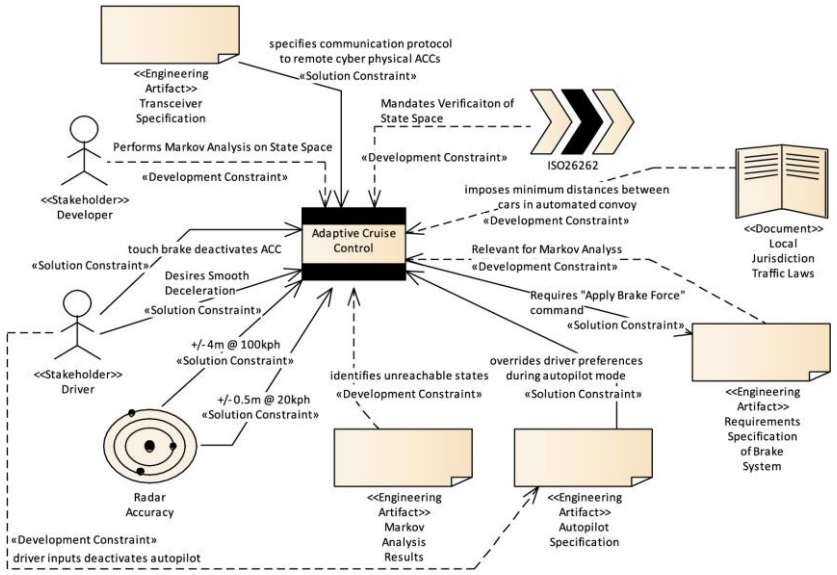
- Context is main source of requirements and rationales
- Context is source of safety and security risks/threats

Context changes can severely affect the system

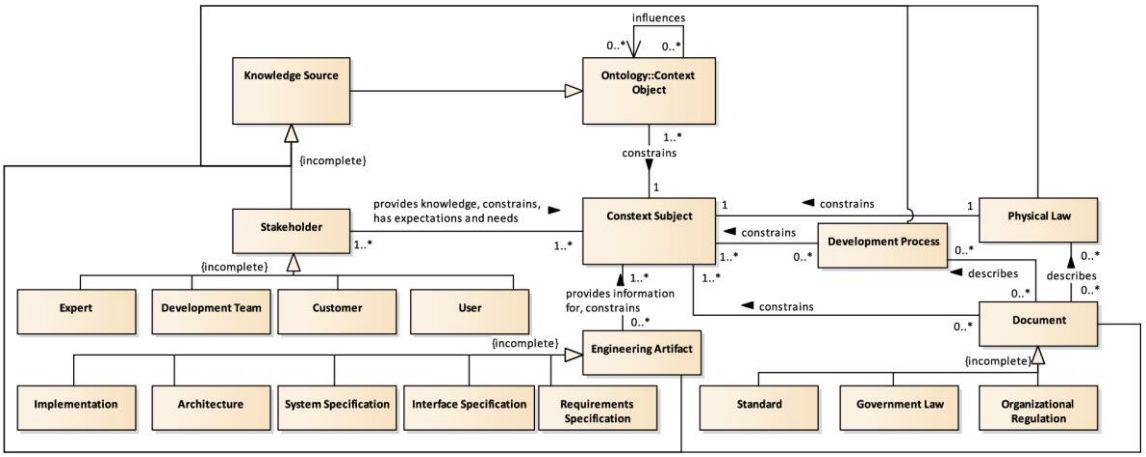
- Functionality
- Correctness
- Safety
- Security

Context of Knowledge

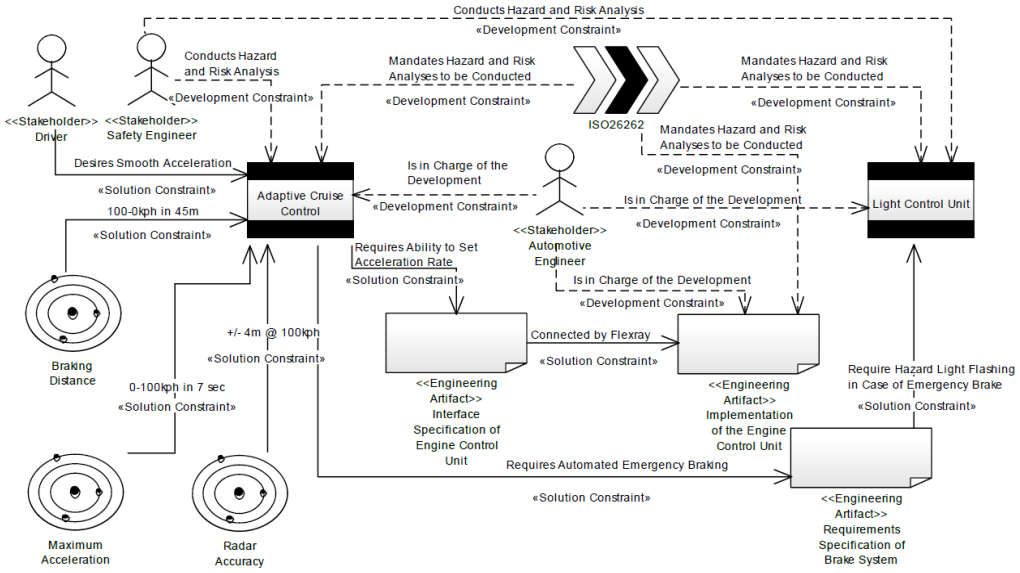
The context of knowledge defines the major requirements sources and important knowledge bearers for the development of a system.



Context of Knowledge Meta Model



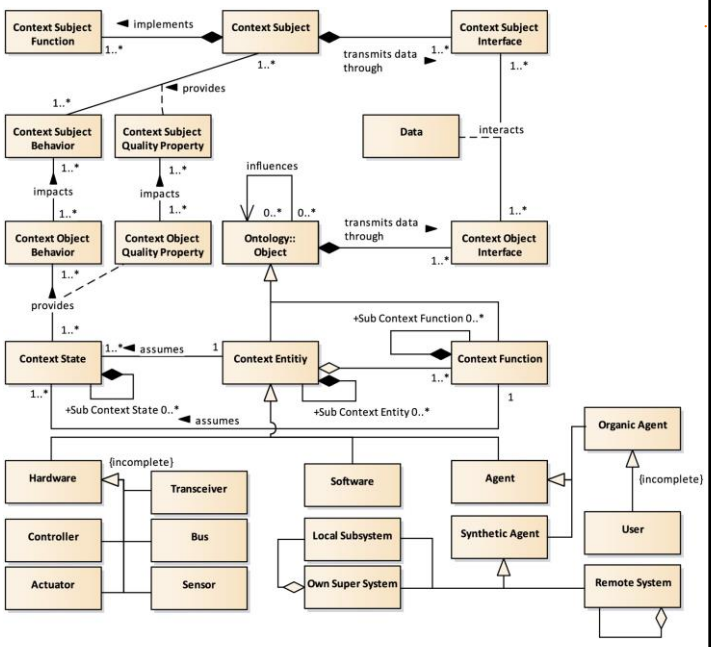
Combining different Contexts



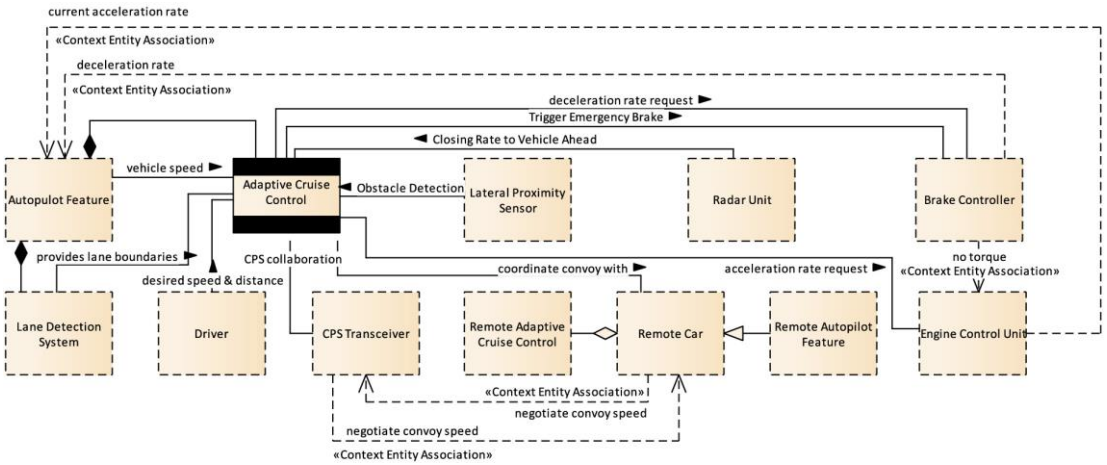
Operational Context

The operational context focusses on context objects that intetract with the system during runtime.

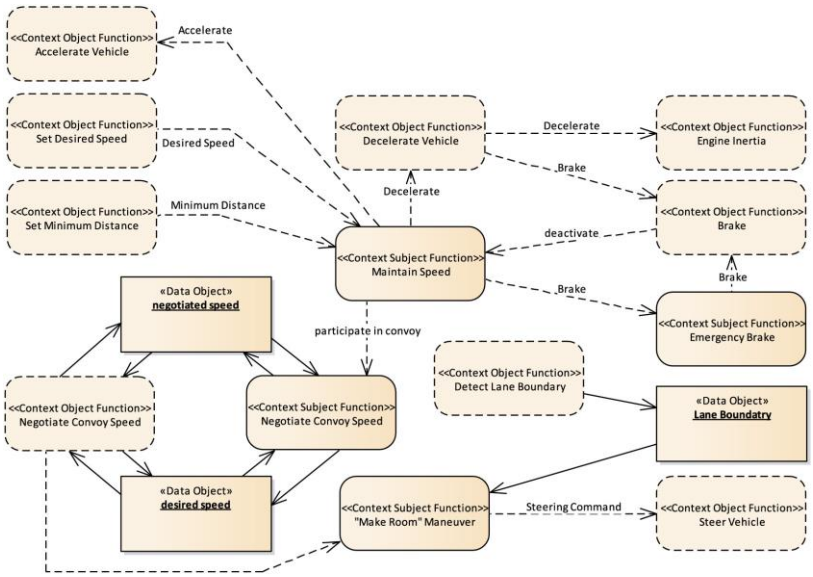
Thus, consideration of the operational context is important for defining interfaces and system behavior in accordance with its environment



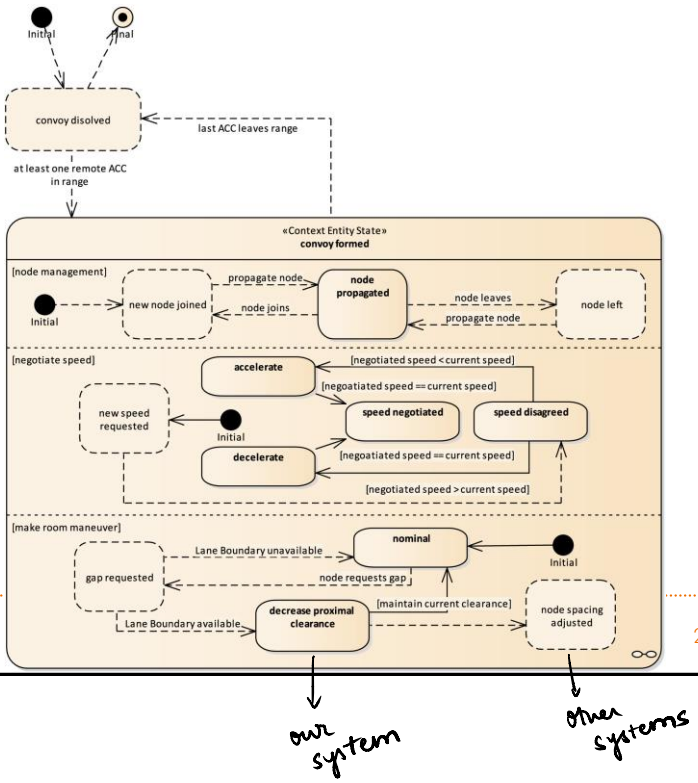
Structural Operational Context



Functional Operational Context



Behavioral Operational Context



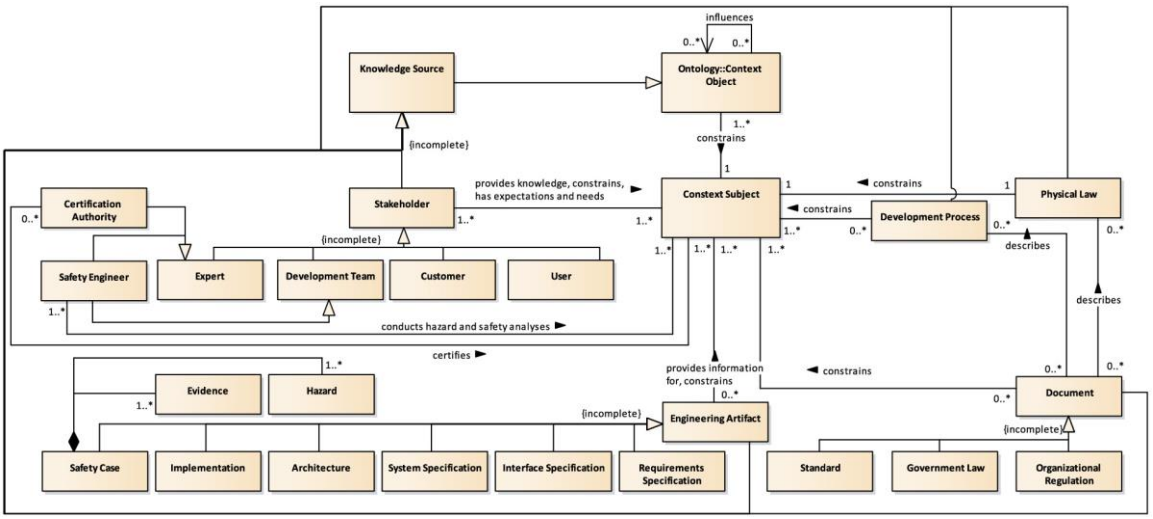
our system

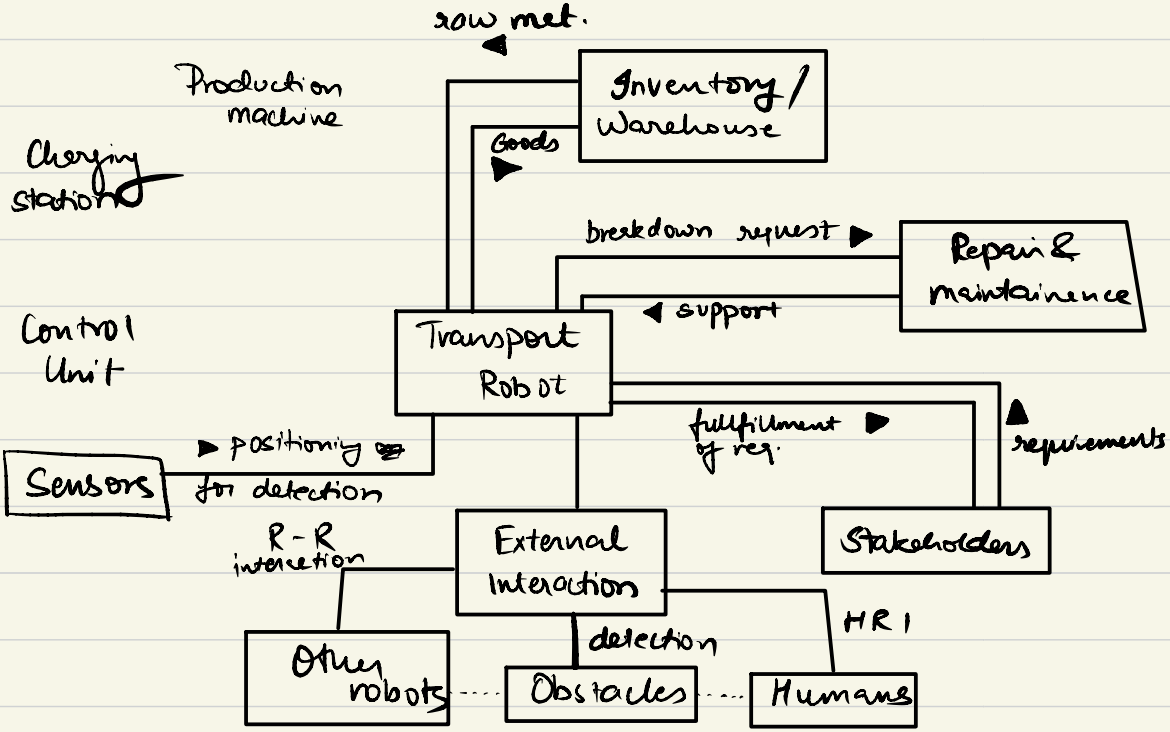
other systems

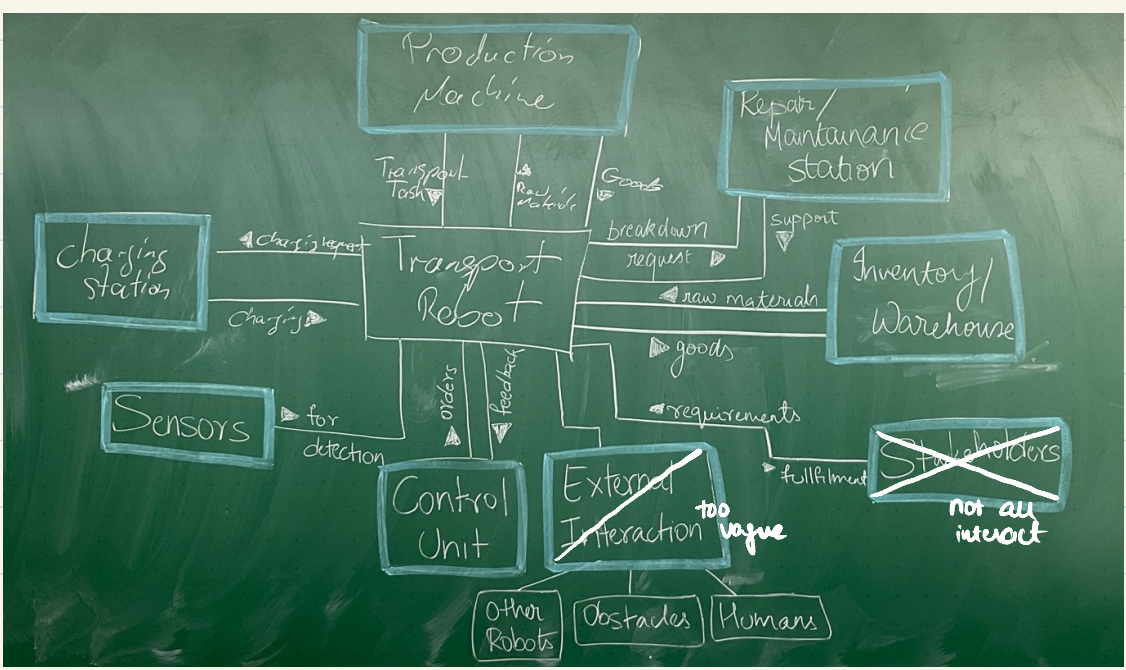
Exercise

Define the Context of
- A Cobot
- A Transport Robot

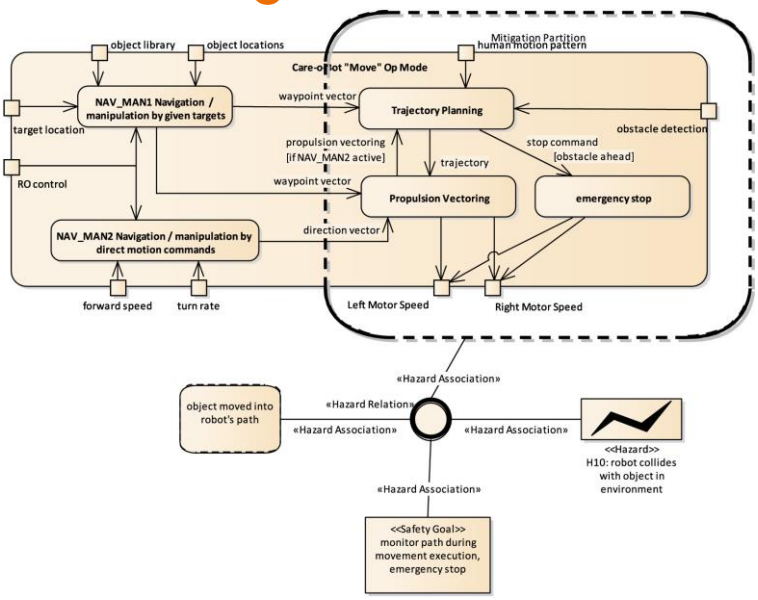
Safety Analyses




raw mat.
Production machine
Inventory / Warehouse
Goods
Charging station
breakdown request
Repair & maintainence
support
Control Unit
Transport Robot
positioning for detection
Sensors
fullfillment of req.
requirements
R-R interaction
External Interaction
Stakeholders
HRI
detection
Other robots
Obstacles
Humans


Production Machine
Repair/ Maintanance station
Transport Tasks
Raw Materials
Goods
Charging station
charging request
charging
Transport Robot
breakdown request
support
raw materials
Inventory/ Warehouse
goods
orders
feedback
Sensors
for detection
requirements
fullfilment
Stakeholders
not all relevant
Control Unit
External Interaction
too vague
Other Robots
Obstacles
Humans

Hazard Modelling



Exercise

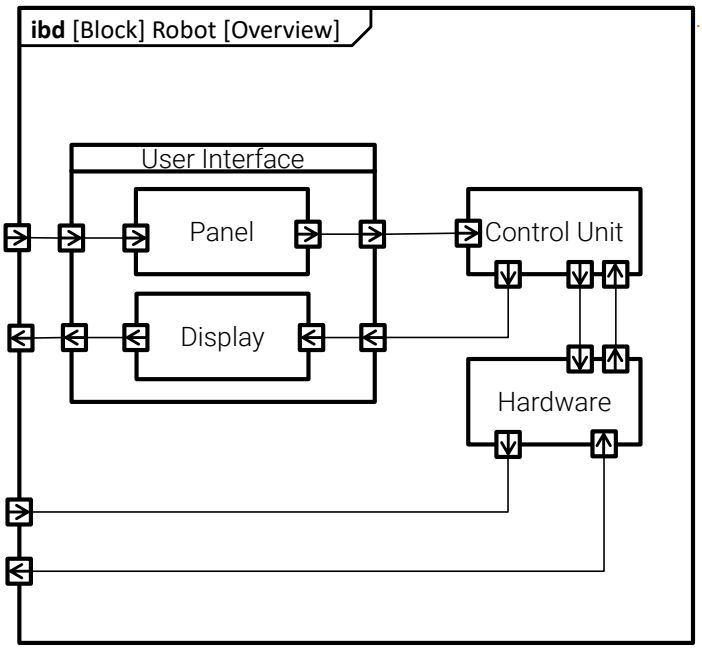
Identify Safety Hazards for
- A Cobot
- A Transport Robot

Modelling Structure with SysML Internal Block Diagrams

Internal Block Diagram

IBDs define the internal structure

IBDs define interfaces



What is an IBD?

Internal Block Diagram (ibd): An *Internal Block Diagram* is a static structural diagram owned by a particular Block that shows its encapsulated structural contents: Parts, Properties, Connectors, Ports, and Interfaces. Stated otherwise, an IBD is a "white-box" perspective of an encapsuated ("black-box") Block.

- Blocks can be recursively decomposed ("nested") into Parts by alternating between Block Definition Diagram (BDD) *definitions* and Internal Block Diagram (IBD) usages (See *Usage Notes* below.)
- Behaviors can either be encapsulated by Blocks (e.g., Operations, Signals, and State Machines) or Allocated (via «allocate» Dependency) to Blocks (e.g., Activities/Actions) directly or indirectly (via Interfaces).
- […]

From: https://sysml.org/sysml-faq/what-is-internal-block-diagram.html

Purpose of IBDs

The purpose of Internal Block Diagrams (IBDs) is to show the encapsulated structural contents (Parts, Properties, Connectors, Ports, Interfaces) of Blocks so that they can be recursively decomposed and "wired" using Interface Based Design techniques. […]

From: https://sysml.org/sysml-faq/what-is-internal-block-diagram.html

BDD vs IBD

BDD Block *Definition* vs. IBD Block *Usage* Dichotomy

BDDs and IBDs complement each other (cf. black-box vs. white-box) and support recursive structural decomposition techniques during System Analysis & Design.

- A BDD *defines* a Block's Properties, including its Part Properties (strongly owned Parts) and Reference Properties (shared Parts)
- IBD specifies Part Properties and Reference Properties *usages* or roles in the structural context of the Block that encapsulates them. Stated otherwise, Part Properties and Reference Properties in an IBD can have a different usages or roles depending upon how they are *realized* ("wired") in the IBD.
- […]

From: https://sysml.org/sysml-faq/what-is-internal-block-diagram.html

Exercise

Model BDDs and IBDs for describing
- A Cobot
- A Transport Robot

Questions for Self-Assessment

What is a Block Definition Diagram?
What is an Internal Block Diagram?
How do BDDs and IBDs complement each other?
What are important system layers in the development of embedded systems?
What influences the definition of components?
Why is context important?
How can context be used to identify safety hazards?

Literature

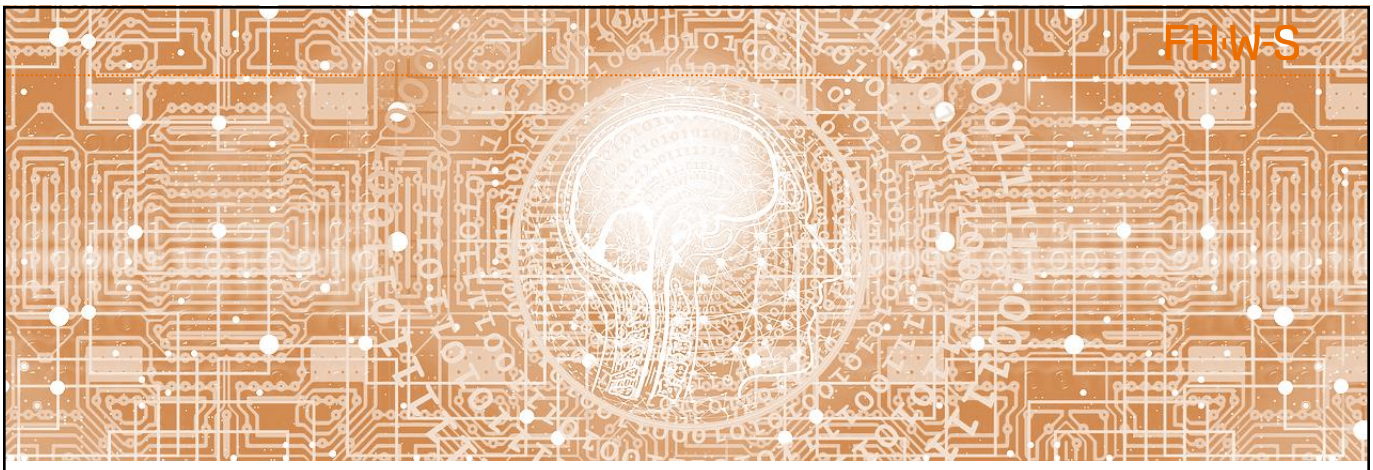
[Daun et al. 2016] M. Daun, B. Tenbergen, J. Brings, T. Weyer: SPES XT Context Modeling Framework. In: K. Pohl, M. Broy, H. Daembkes, H. Hönninger (eds.) Advanced Model-Based Engineering of Embedded Systems, Springer, 2016, pp. 43-57.

[Daun & Tenbergen 2022] M. Daun, B. Tenbergen: Context modelling for cyber-physical systems. In: J. Softw. Evol. Proc., Wiley, 2022.

[Friedenthal et al. 2014] Sanford Friedenthal, Alan Moore & Rick Steiner: A Practical Guide to SysML – The Systems Modeling Language. Morgan Kaufmann OMG Press, 2014.

[OMG SysML] OMG System Modeling Language. Version 1.6, Object Management Group, 2019.

[OMG UML] OMG Unified Modeling Language. Version 2.5.1, Object Management Group, 2017.



Embedded Systems and Field Buses

Prof. Dr. Marian Daun

FH·W·S University of Applied Sciences Würzburg-Schweinfurt

Agenda

- Fundamentals
- Structure of Embedded Systems
- Behavior of Embedded Systems
- Design of Embedded Systems
- Communication
- Real-time
- Collaborative Embedded Systems

FH·W-S

Behavior of Embedded Systems

Part A: Petri Nets

Group Discussion

Why is Structure not enough?

Formal Languages

Behavior is typically described using formal languages.

The use of formal languages allows (among others)

- Formal **Proof of Concept**
- **Simulation**
- Analyses (e.g. **Deadlock** and **Lifelock** detection, finding an **Equilibrium**)
- **Verification**
- Test Case Generation
- …

Automata Theory

Finite State Machines

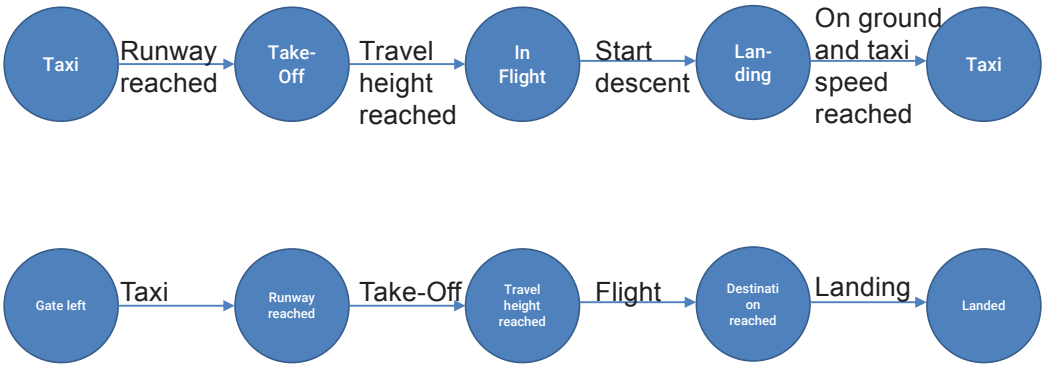
Automata typically have
- A **formal** representation
- A **graphical** representation

Automata foremost consist of
- **States**
- **Transitions**

Further concepts include:
- Start/initial states
- Final/accepting states
- Events and Conditions

Depending on the language the focus is either on
- the **States**
- the **Transitions**
- or **Both**

State vs Transition



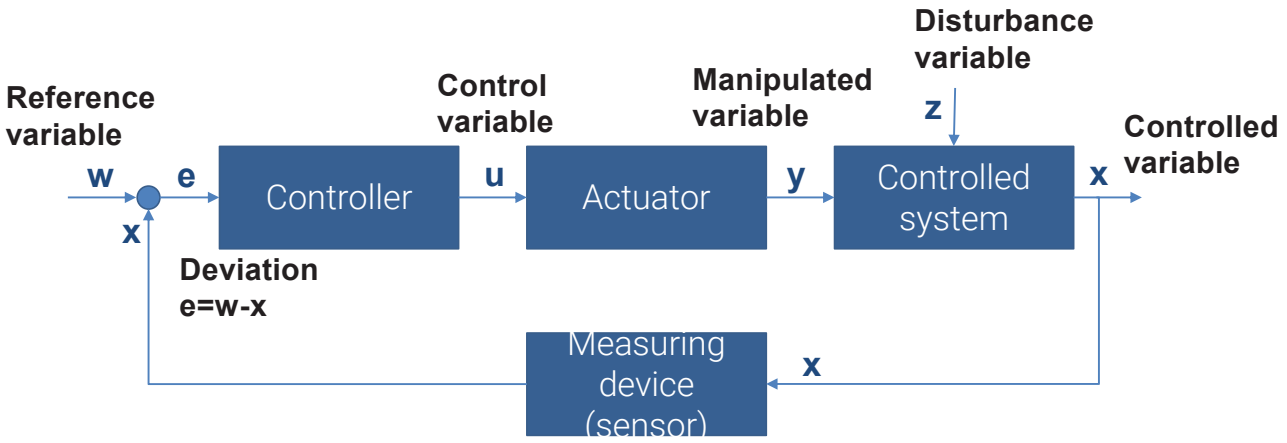
Group Discussion

What is an Alphabet?

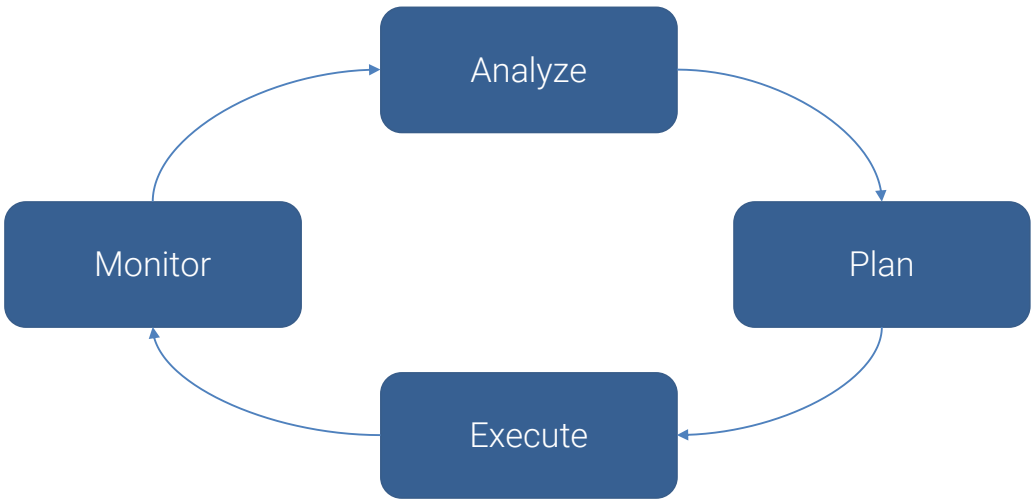
What are basic Operators?

Remember
Control- and MAPE-Loops

Control Loop



MAPE-Loop

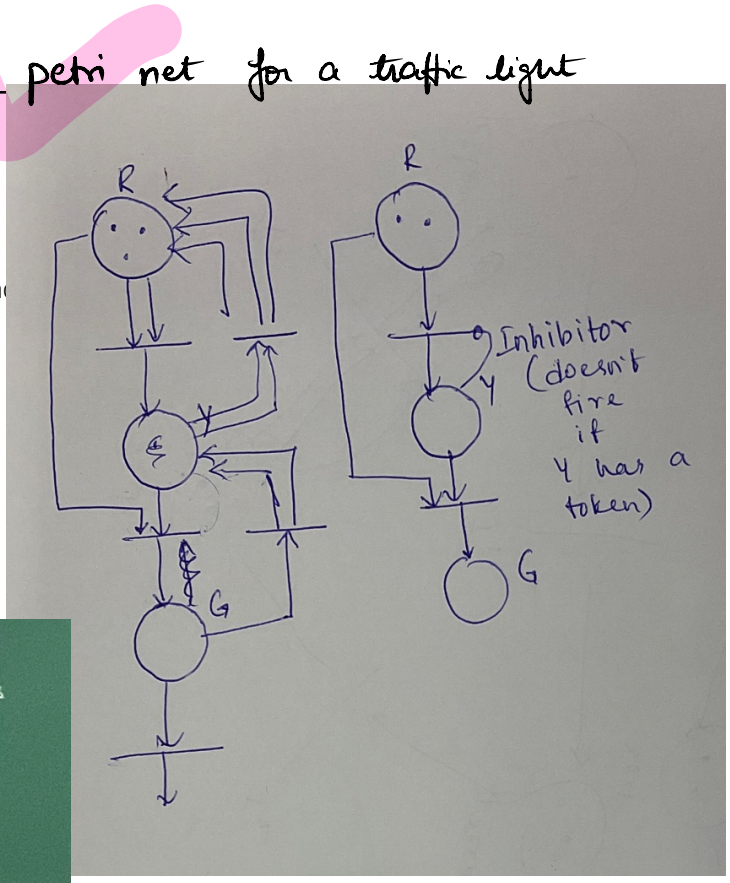


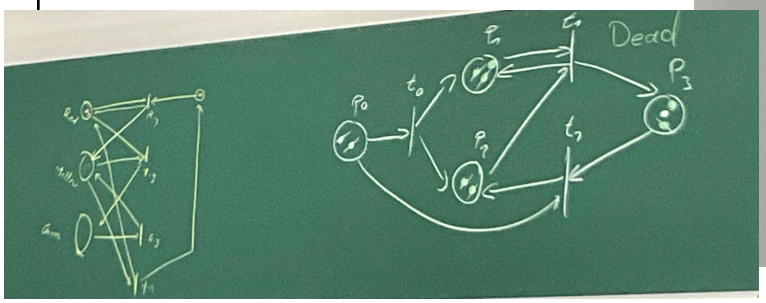
Petri Nets

Online-Materials

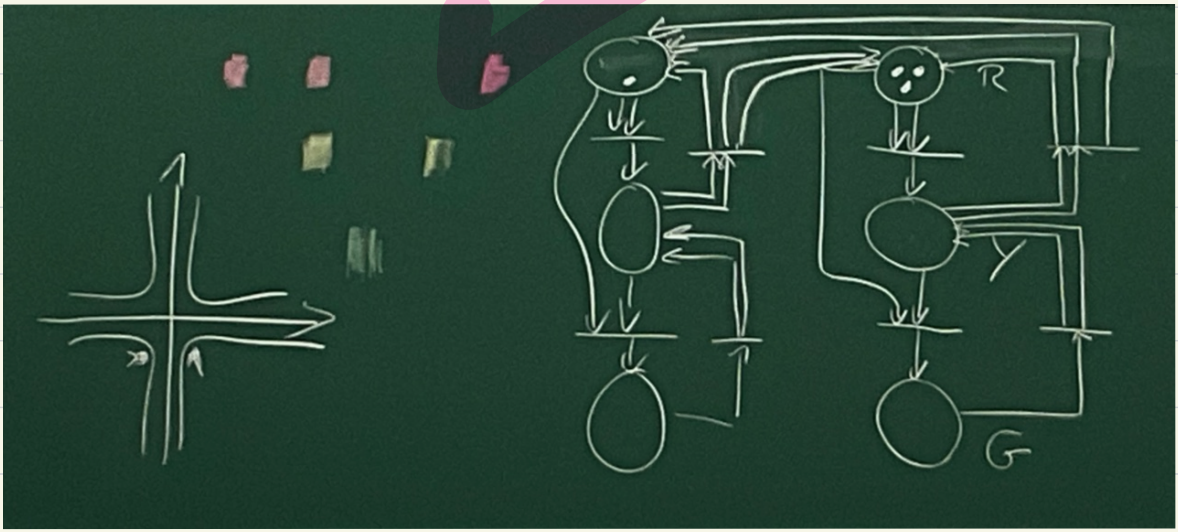
Introduction videos to Petri Nets can be foun

https://youtu.be/GCsVxWh995o
https://youtu.be/WGSAi9-QUwk
https://youtu.be/1IPOlE0PvQY

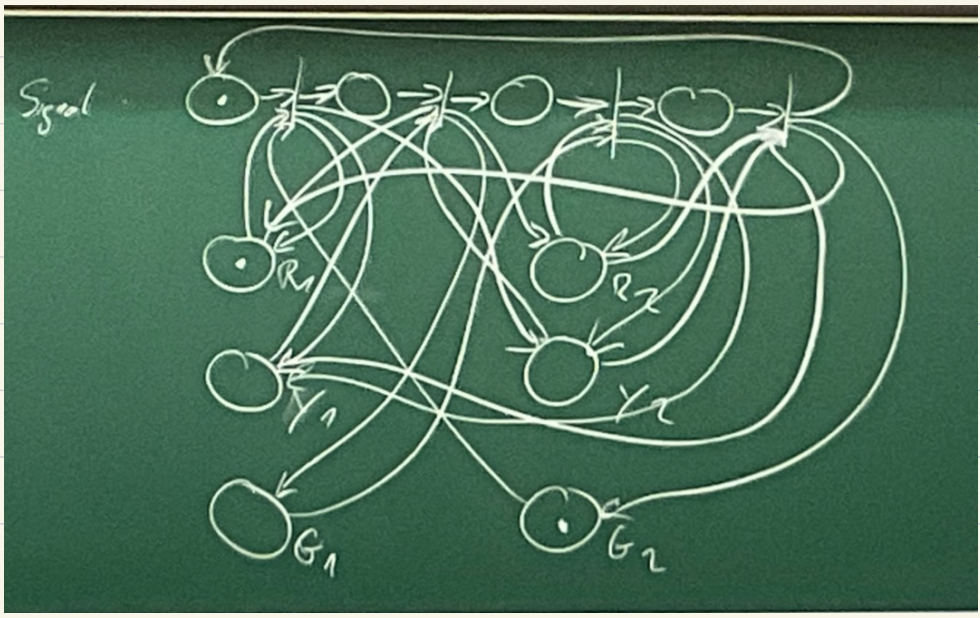


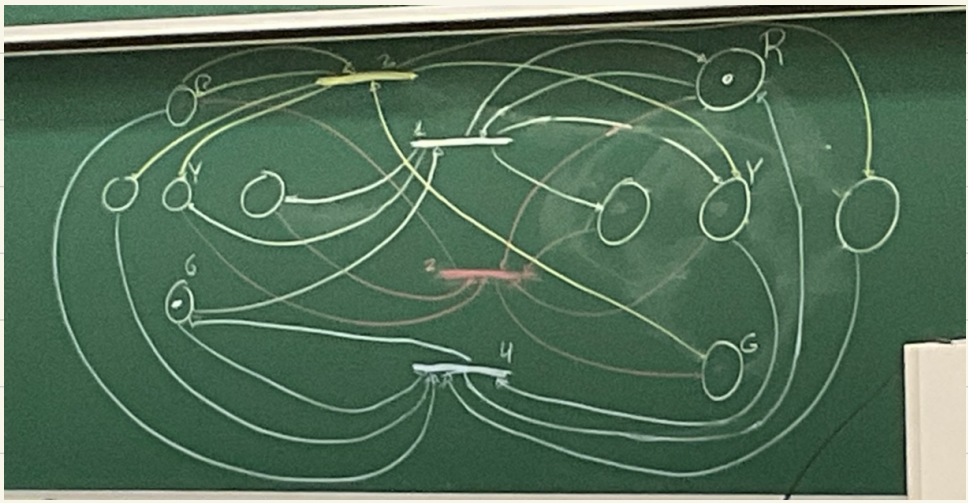


2 traffic lights

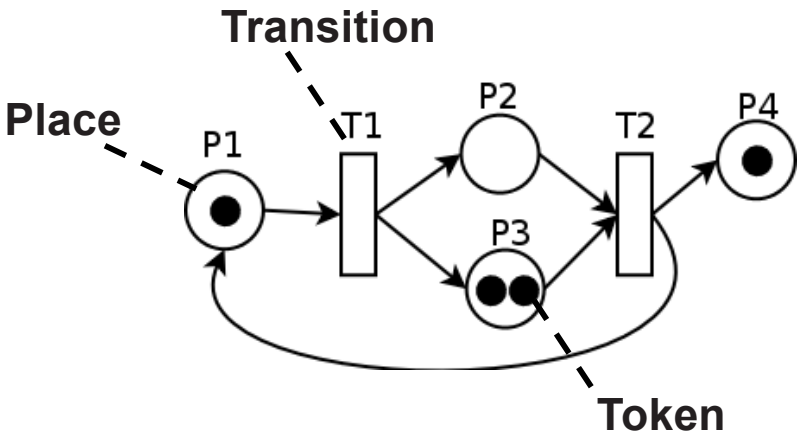


synchronized ↓



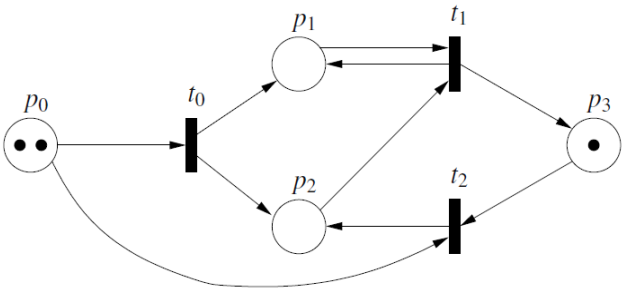

R
Y
G
R
Y
G
2
1
2
4

Elements



Petri nets

Is transition t_2 of the petri net shown below enabled? (Capacity of places is ∞)
What is the marking after t_2 fires?



Exercise

Model a Traffic Light using Petri Nets

Analysis

Terms:

A transition is dead, if it is not enabled in any marking.

A transition t is potentially fireable, when there is at least one marking reachable, which enables t.

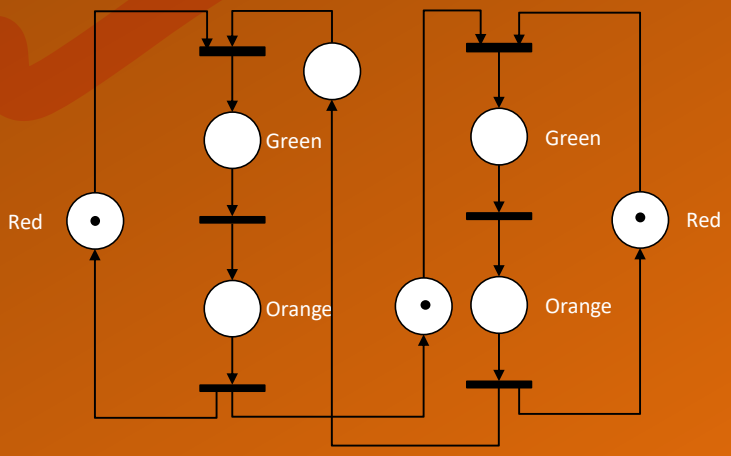
A petri net is dead, if no transition is able to fire in a given marking.

A petri net is quasi alive, if not dead under any future marking.

A petri net is alive, when all transitions are potentially fireable for all future markings.

Exercise

Quasi alive or alive?

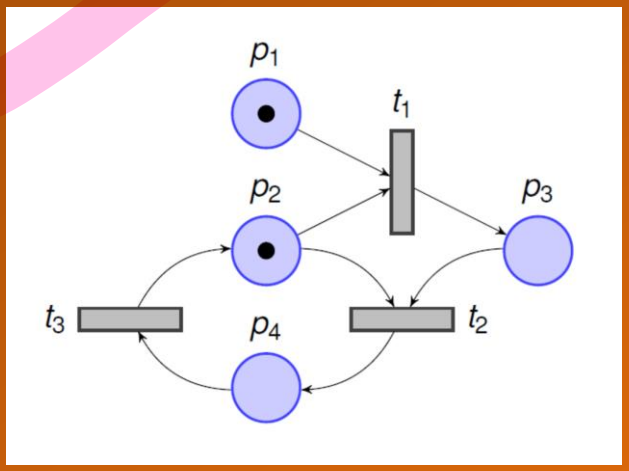


alive

all are potentially fireable

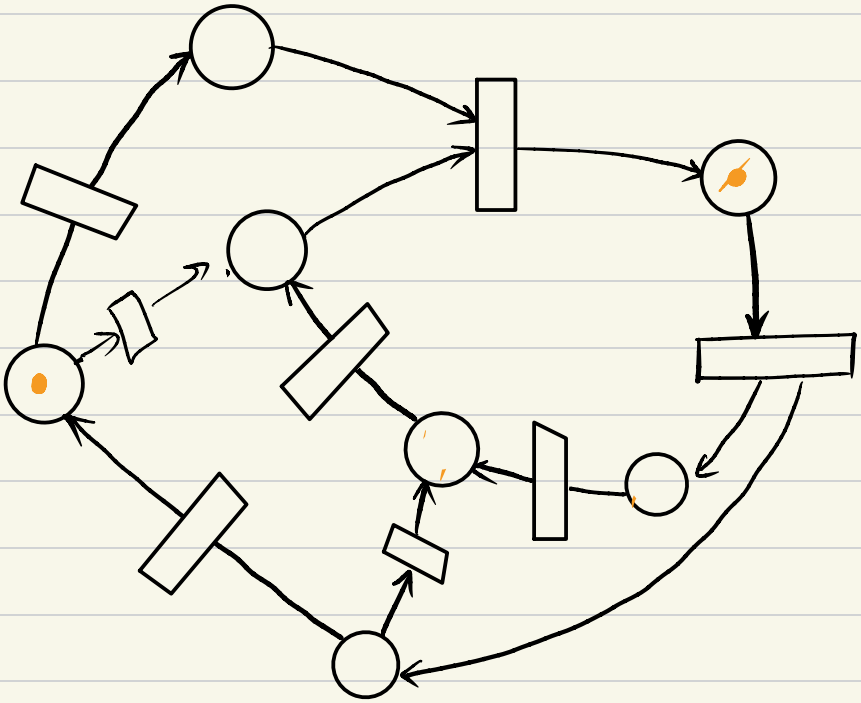
Exercise

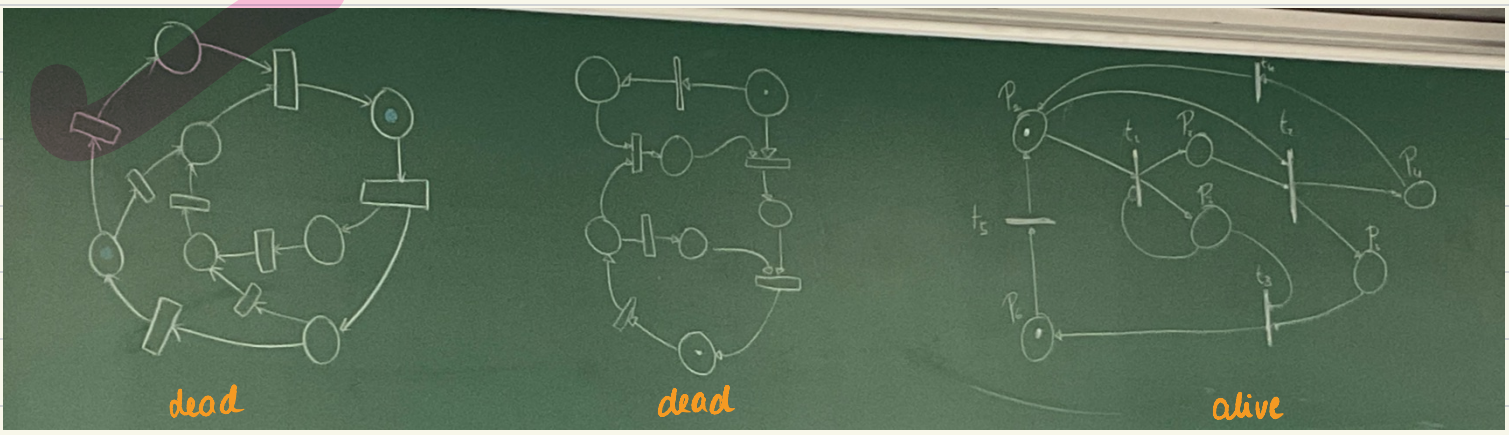
Quasi alive or alive?

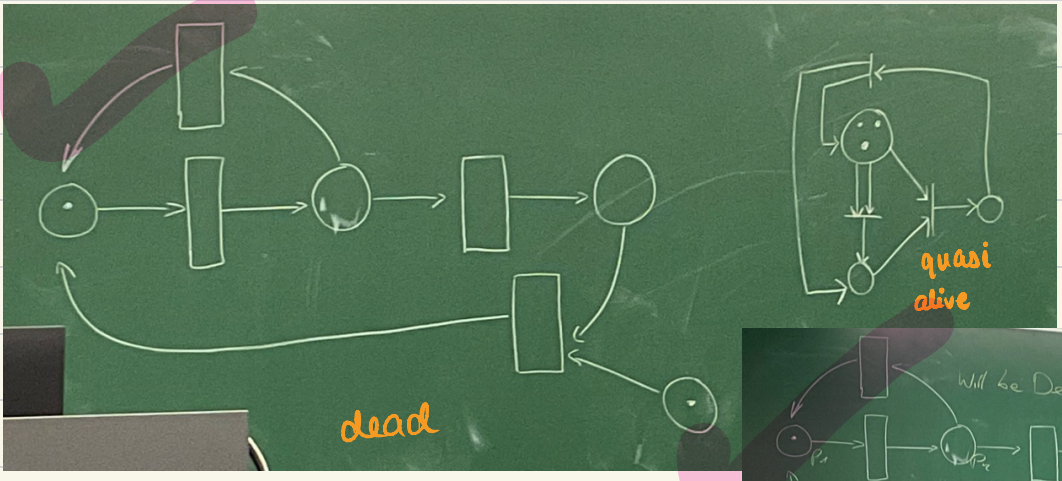


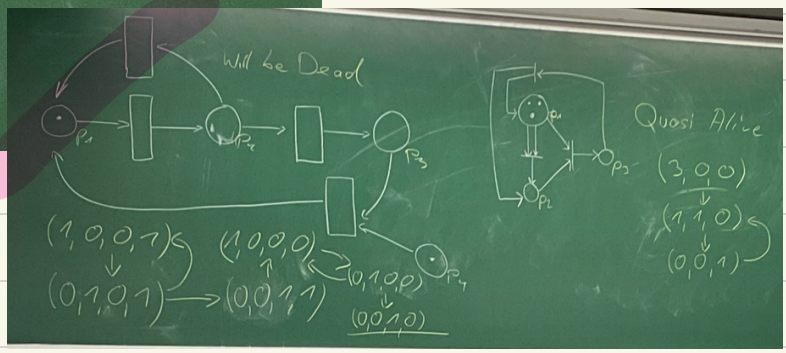
dead

after p_1 is fired it can't be fired again.




dead
dead
alive

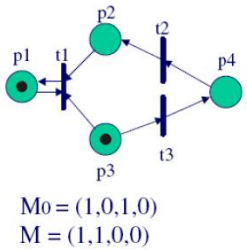

quasi
alive
dead

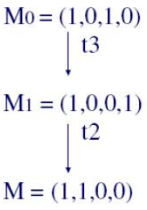

Will be Dead
Quasi Alive
(3,0,0)
(1,1,0)
(0,0,1)
(1,0,0,1)
(0,1,0,1)
(1,0,0,0)
(0,0,1,1)
(0,1,0,0)
(0,0,1,0)

Reachability Graph

Firing sequence for a petri net G and initial marking M0 is a sequence of transitions $t_0, \dots, t_n$, such that $M_0 \xrightarrow[G,t_0]{} M_1 \xrightarrow[G,t_1]{} \dots \xrightarrow[G,t_n]{} M_n$

Reachability graph: The reachability graph of G is the transition relation $\xrightarrow[G]{}$ restricted to its reachable markings $R(G)$. It is the state space of the net.

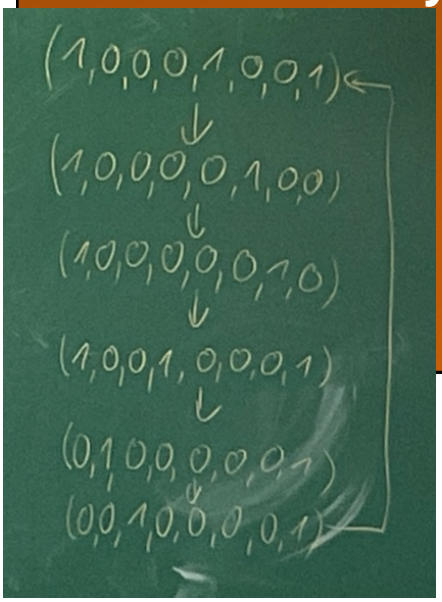




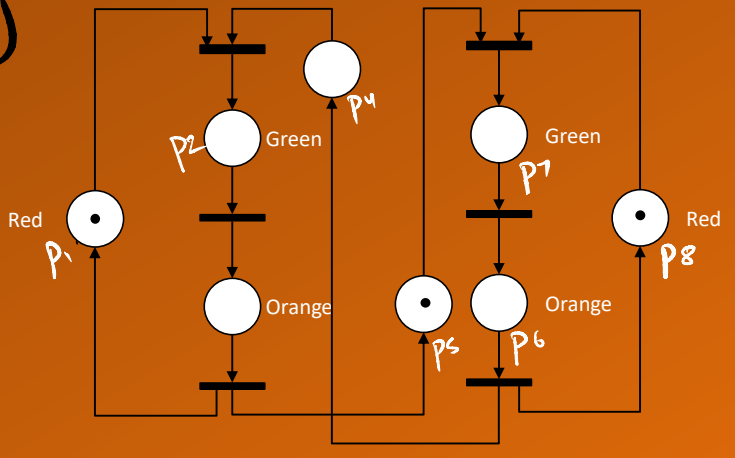
Nassar, K., & Casavant, A. (2008). Analysis of timed Petri nets for reachability in construction applications. *Journal of Civil Engineering and Management*, *14*(3), 189-198.

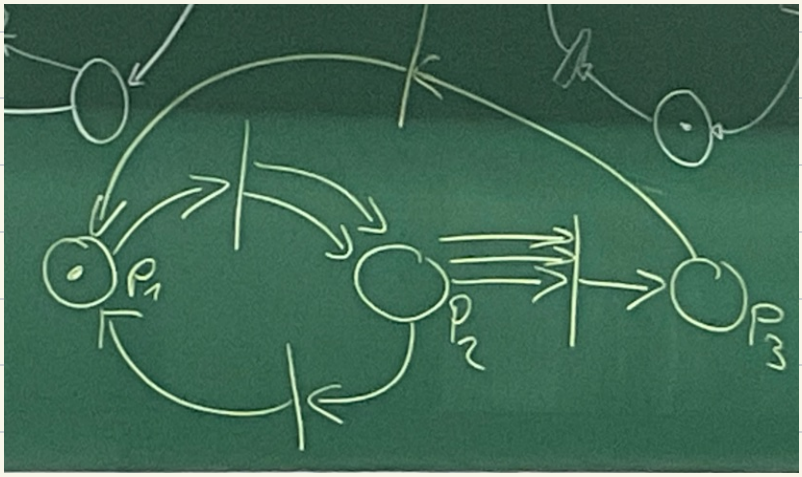
$M_0 = (10001001)$
$M_1 = (10000010)$
$M_2 = (10000100)$
$M_3 = (10010000)$
$M_4 = (\quad)$

Draw the Reachability



Exercise





$M_0 = (1\ 0\ 0)$

$M_1 = (0\ 2\ 0)$

$M_2 = (1\ 1\ 0)$

$M_3 = (0\ 3\ 0)$

$M_4 = (1\ 2\ 0)$

$M_5 = (0\ 4\ 0)$

$M_6 = (1\ 0\ 1)$

$M_7 = (0\ 2\ 1)$

$M_8 = (1\ 2\ 0)$

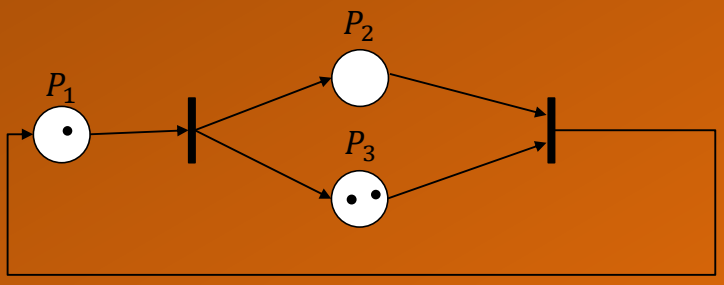
$M_4 = (0\ 0\ 1)$

$M_5 = (1\ 0\ 0)$

$M_0 = (1\ 0\ 2)$
$M_1 = (0\ 1\ 3)$
$M_2 = (1\ 0\ 2)$

Exercise

Draw the Reachability Graph



Group Discussion

Why are Petri Nets commonly suggested for analysing embedded systems?

Where are Limitations of Petri Nets?

Exercise

For home:
Google the Dining Philosophers Problem and Model it with Petri Nets

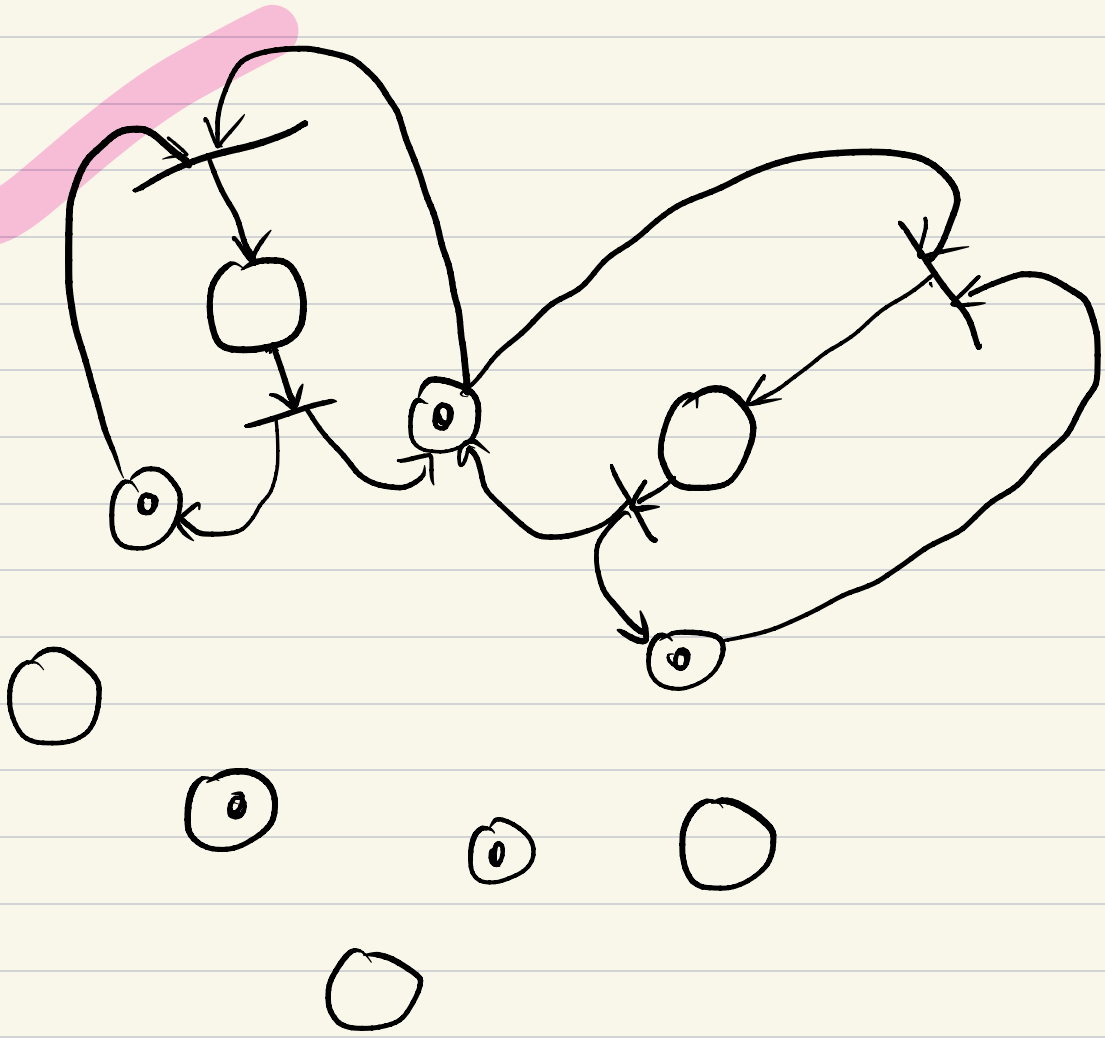
Questions for Self-Assessment

What are the basic concepts of automata theory?
What is a petri net?
When does a petri net fire?
Are petri nets deterministic?
When is a petri net dead, alive, quasi alive?
What is a reachability graph?
What are shortcomings of petri nets?
Why are petri nets often suggested for modelling embedded systems?

Literature

[Hopcroft et al. 2006] Introduction to Automata Theory, Languages, and Computation. 3rd Edition, Pearson Education, 2006.

[Reisig 1991] Reisig, W.: Petri Nets and Algebraic Specifications. In: Theoretical Computer Science, 80(1), 1991, pp. 1-34.



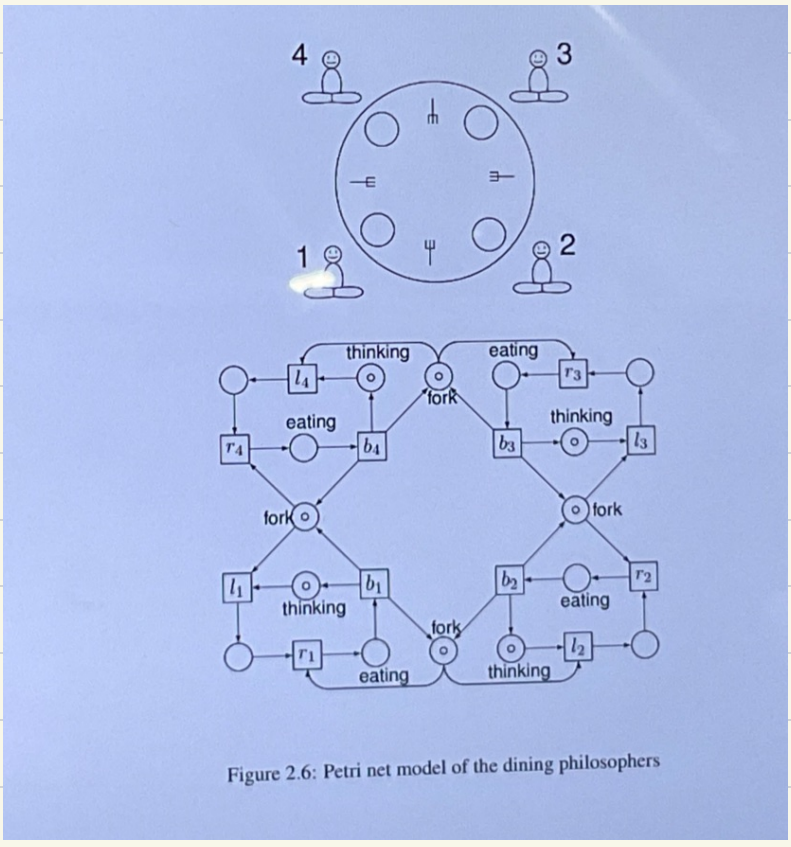
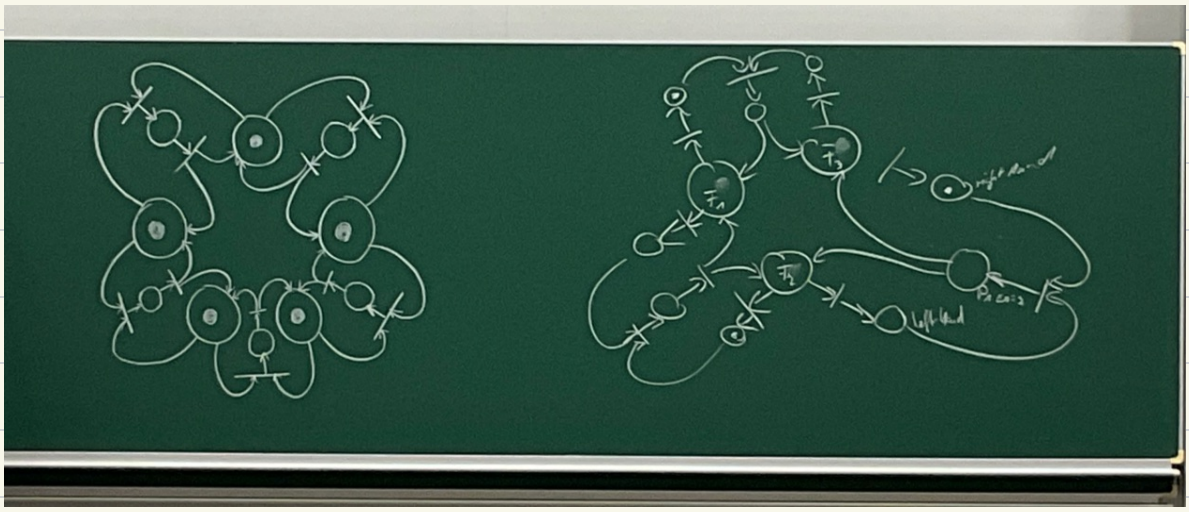
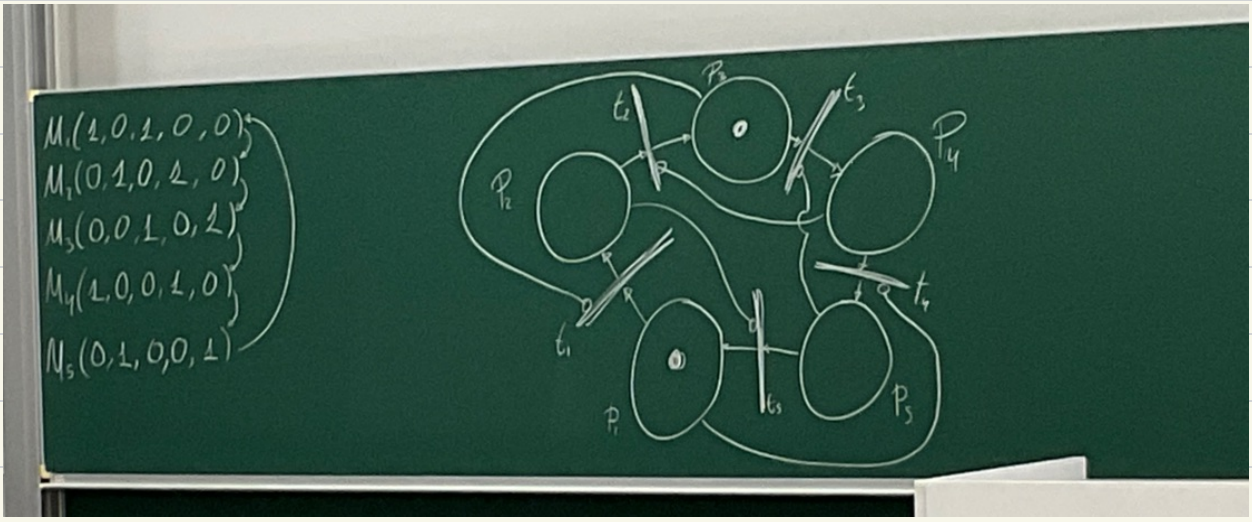
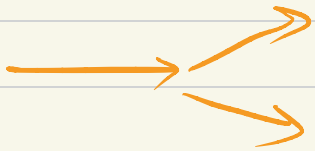


Figure 2.6: Petri net model of the dining philosophers

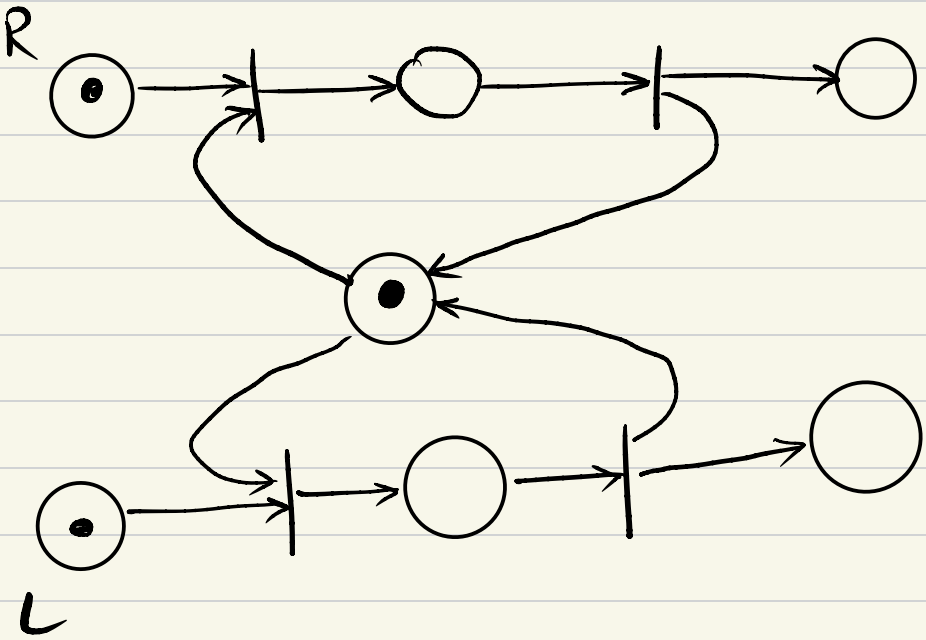
Q) Model Petri nets for Diving Philosophers Problem

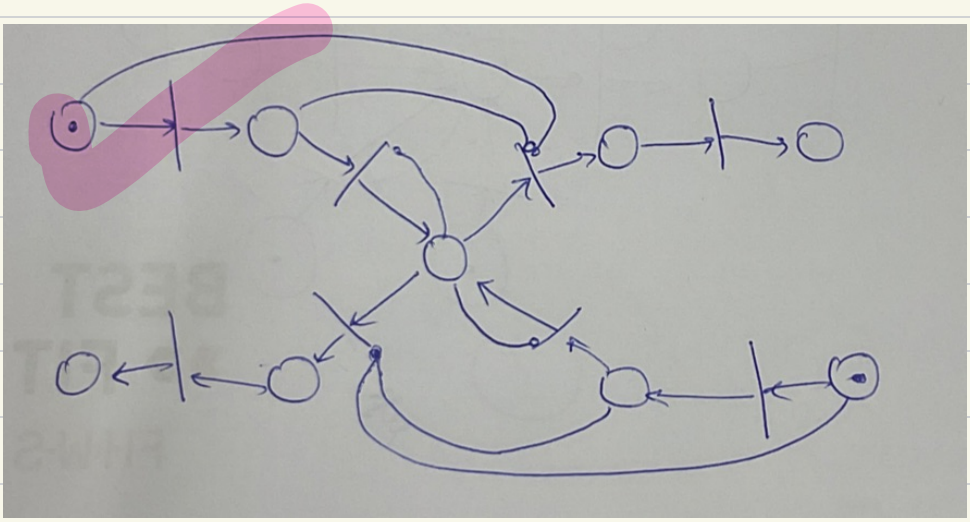


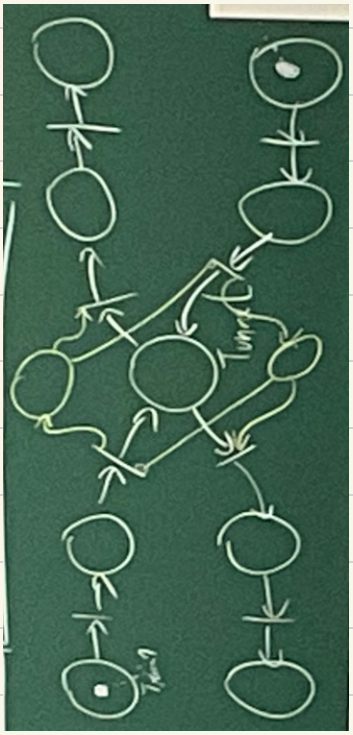


Q Petri nets for 

two trains enter a tunnel and only one can pass

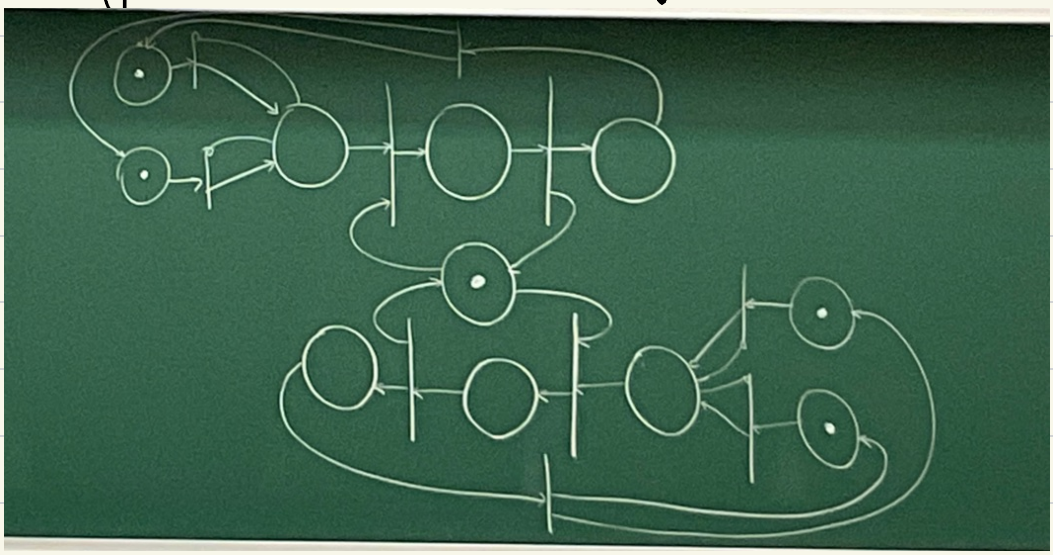






Q Two trains meet at a junction and have an option to go either left or right.

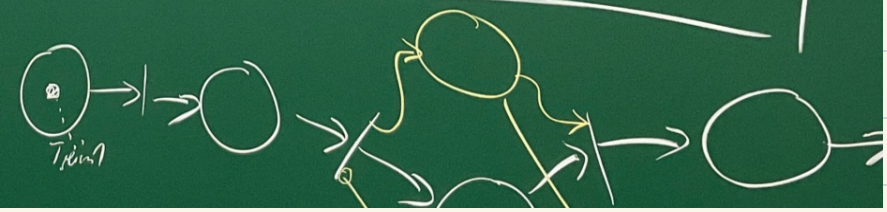
(wrong answer probably)

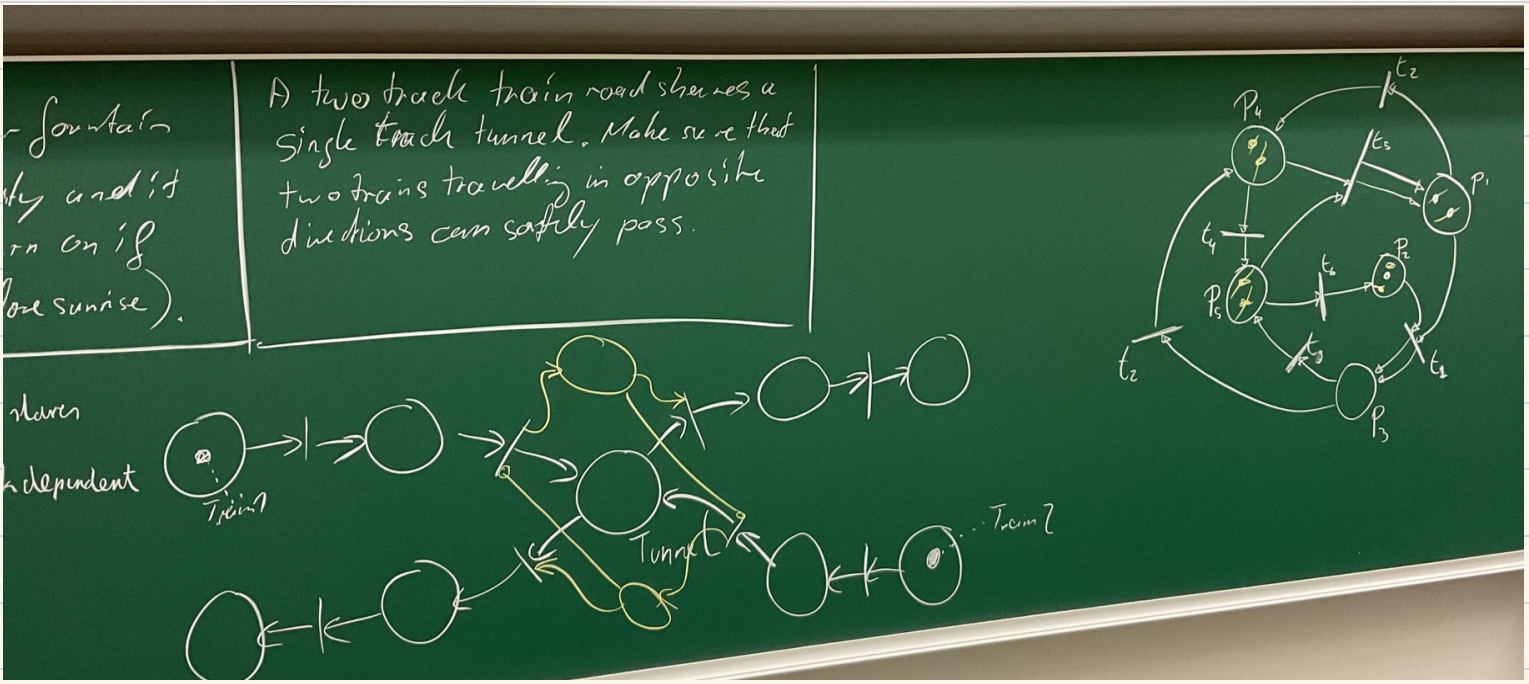


You specify a water fountain. The water fountain shall turn on if someone is in its vicinity and it is between 7am and 1am. Lights shall turn on if movement is detected and it is after dark (before sunrise).

A two track train road shares a single track tunnel. Make sure that two trains travelling in opposite directions can safely pass.

A system of devices is running with an independent clock. It consists of a master and two slaves that measure temperature, send the data and receive an endpoint. Once both of them do their job they can go again. Master device is clock dependent. Master device is choosing the slave at random. Slaves don't depend on the clock.





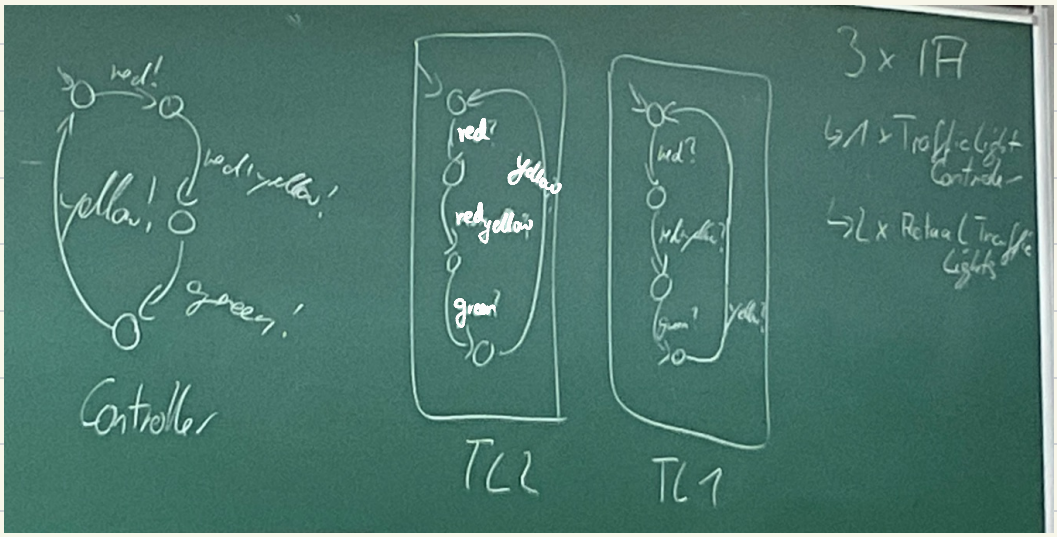
Interface Automata

? input

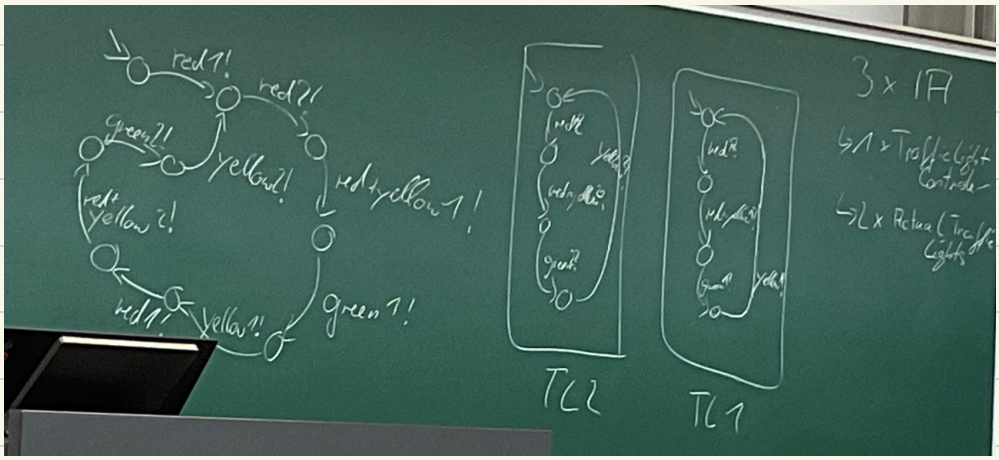
! output

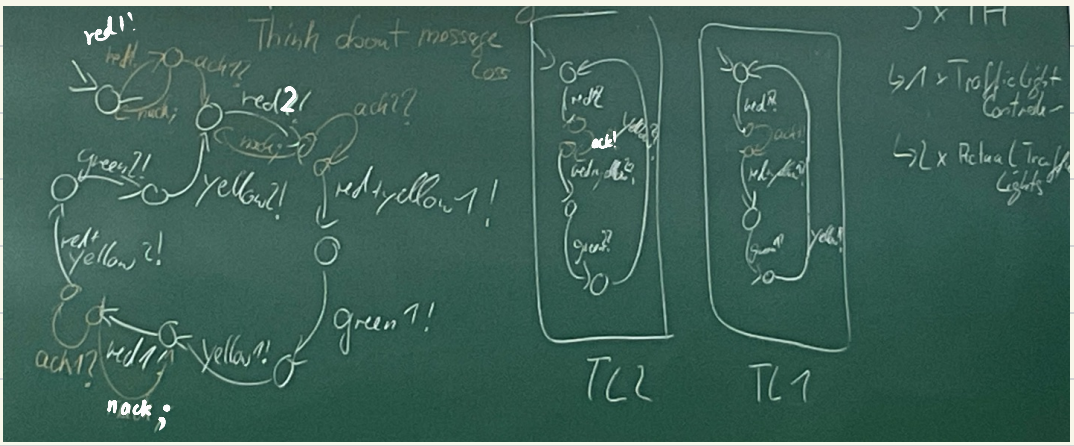
Model a traffic light (1 traffic control, 2 actual traffic lights)

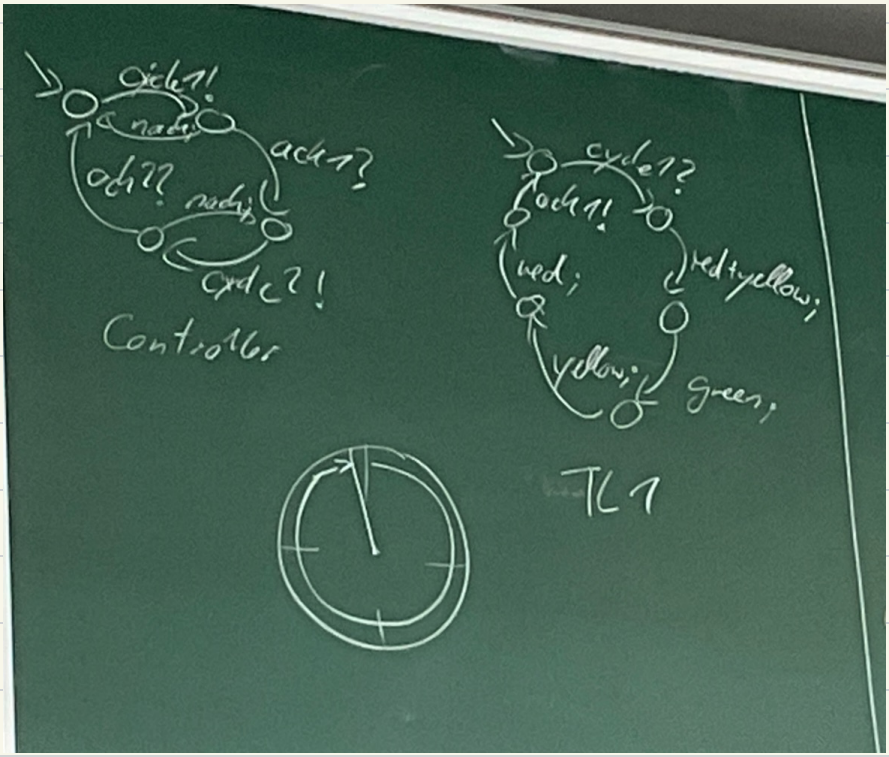
wrong ans



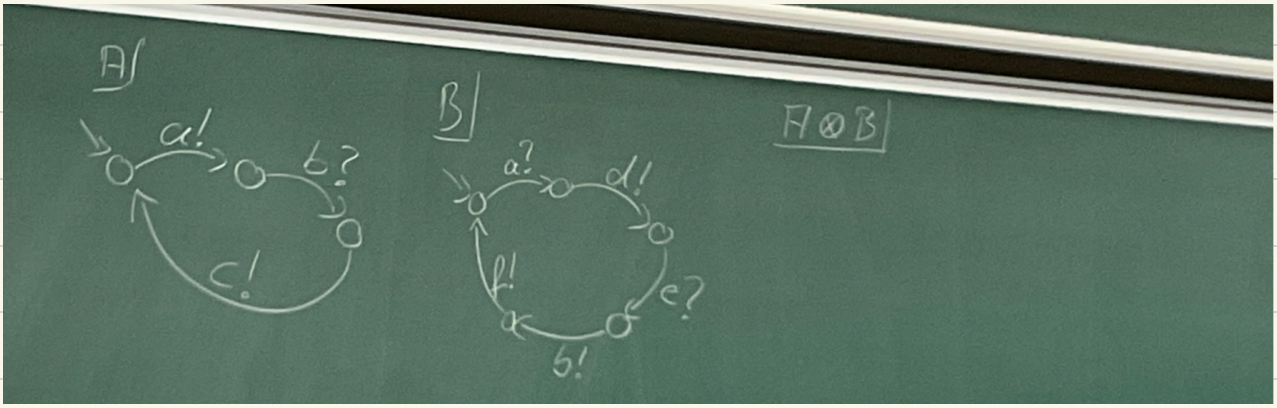
not completely correct

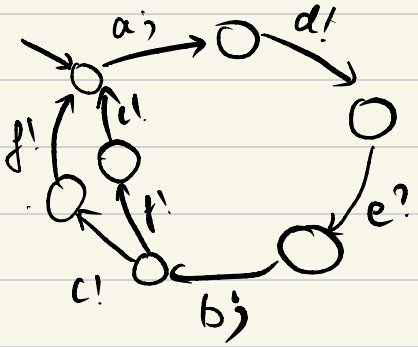




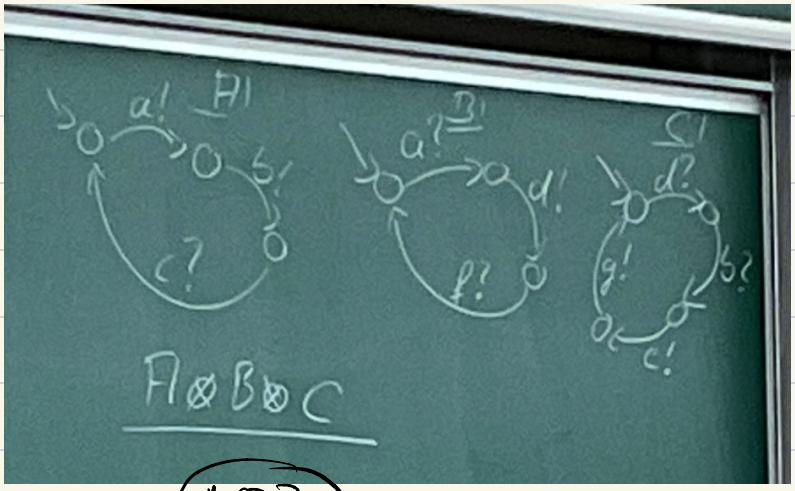

cycle1!
ack1?
ack1?
nack1!
cycle1?!
Controller
cycle1?
ack1!
red;
red+yellow;
yellow;
green;
TL1

9

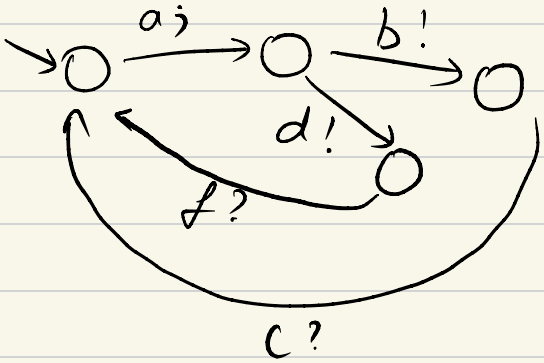

A
a!
b?
c!
B
a?
d!
e?
f!
b!
A⊗B


a?
d!
g!
d!
f!
e?
c!
b?

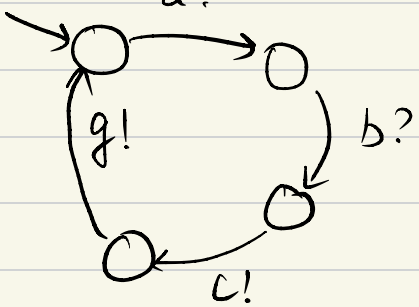
9

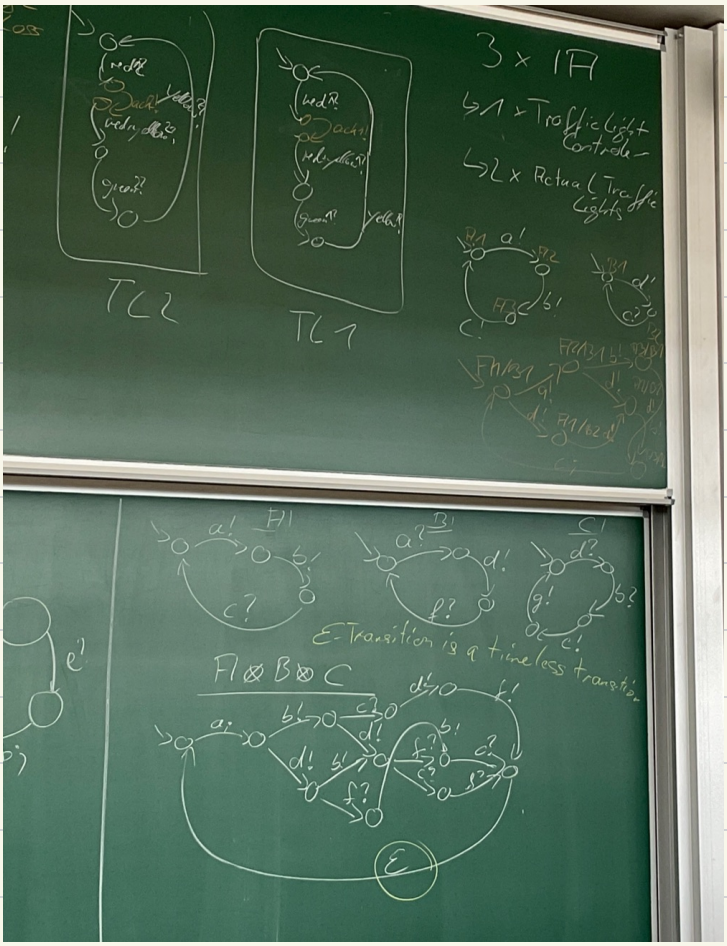

a!
b!
c?
a?
d!
f!
d?
g!
b?
c!
A⊗B⊗C

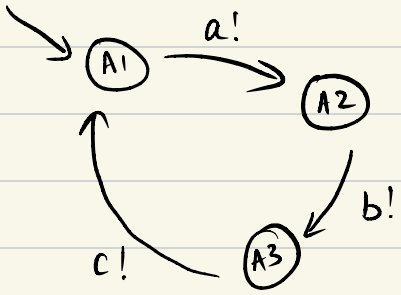
A⊗B

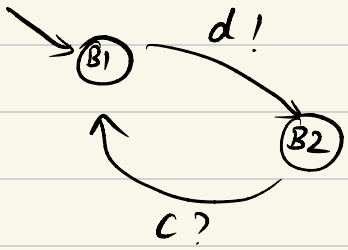

a;
b!
d!
f?
c?

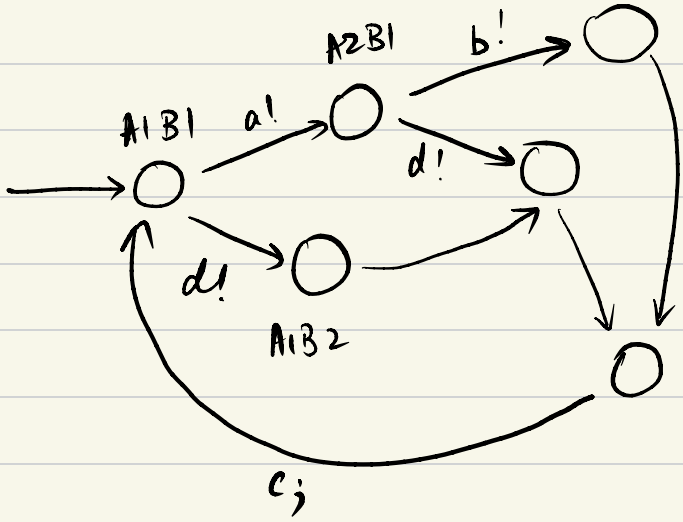
C


d?
g!
b?
c!


3 × IA
↳1 × Traffic Light Controller
↳2 × Return (Traffic Lights)
TL2
TL1
E-Transition is a timeless transition
A⊗B⊗C


A1
a!
A2
b!
A3
c!


B1
d!
B2
c?


A2B1
b!
A1B1
a!
d!
d!
A1B2
c;

Exercises Interface Automata

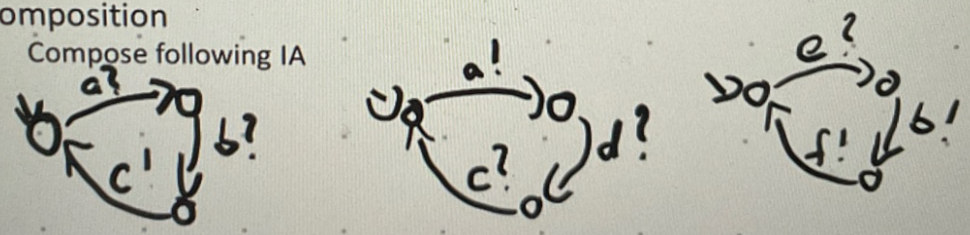
Modelling

Access to a parking garage is controlled by gates. When a car approaches, the entrance gate checks with the garage management system whether a parking space is available. If this is the case, the gate opens and issues the parking ticket. Otherwise, the entrance gate will retry the check every 10 seconds. When leaving the garage, the exit barrier reads the parking ticket and checks with the garage management system whether the parking ticket has been paid. If this is the case, the gate opens. If this is not the case, the barrier remains closed and the ticket is returned.

Marian Daun

Composition

Compose following IA



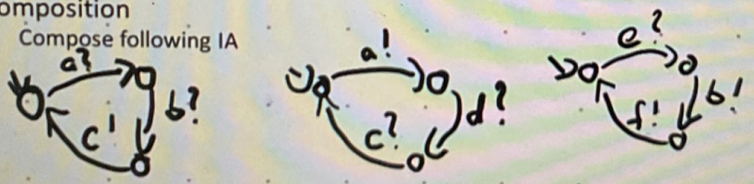
Exercises Interface Automata

Modelling

Access to a parking garage is controlled by gates. When a car approaches, the entrance gate checks with the garage management system whether a parking space is available. If this is the case, the gate opens and issues the parking ticket. Otherwise, the entrance gate will retry the check every 10 seconds. When leaving the garage, the exit barrier reads the parking ticket and checks with the garage management system whether the parking ticket has been paid. If this is the case, the gate opens. If this is not the case, the barrier remains closed and the ticket is returned.

Composition

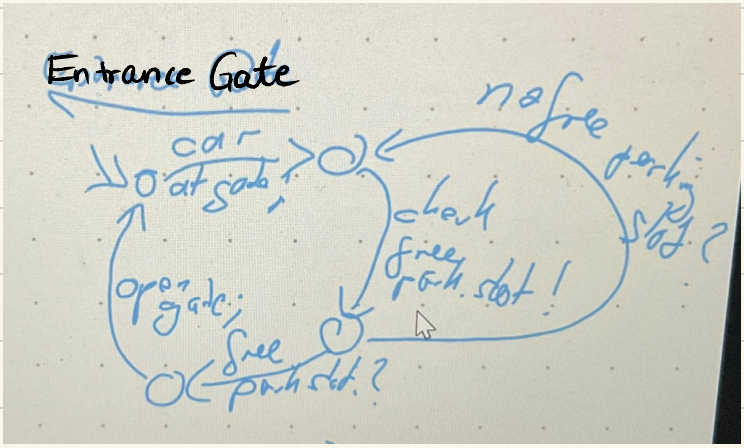
Compose following IA

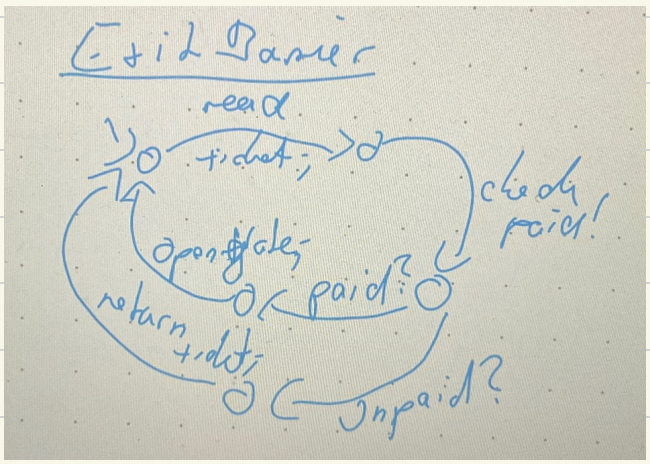


Entrance Gate

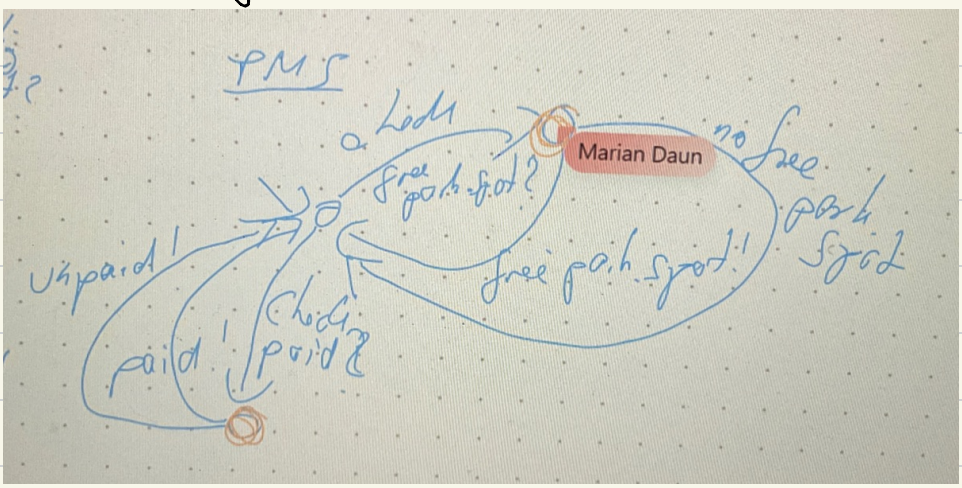
car at gate?

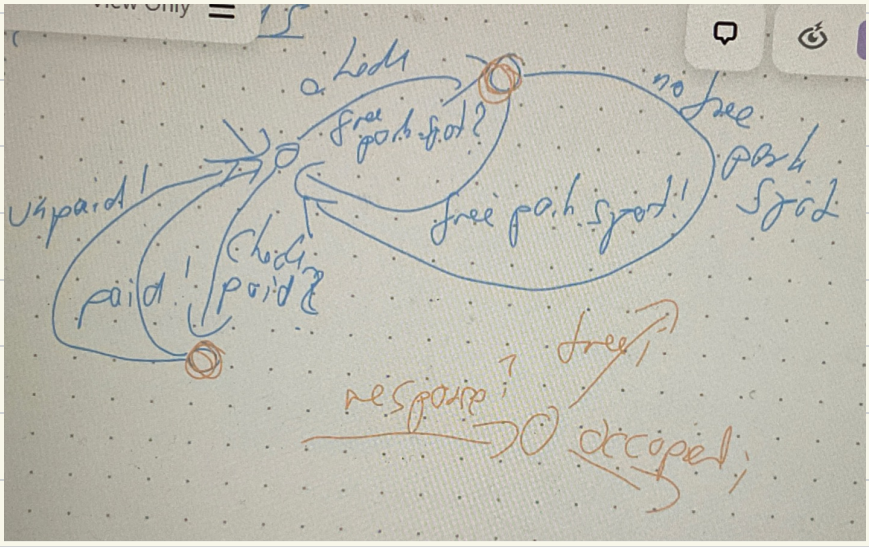
Marian Daun



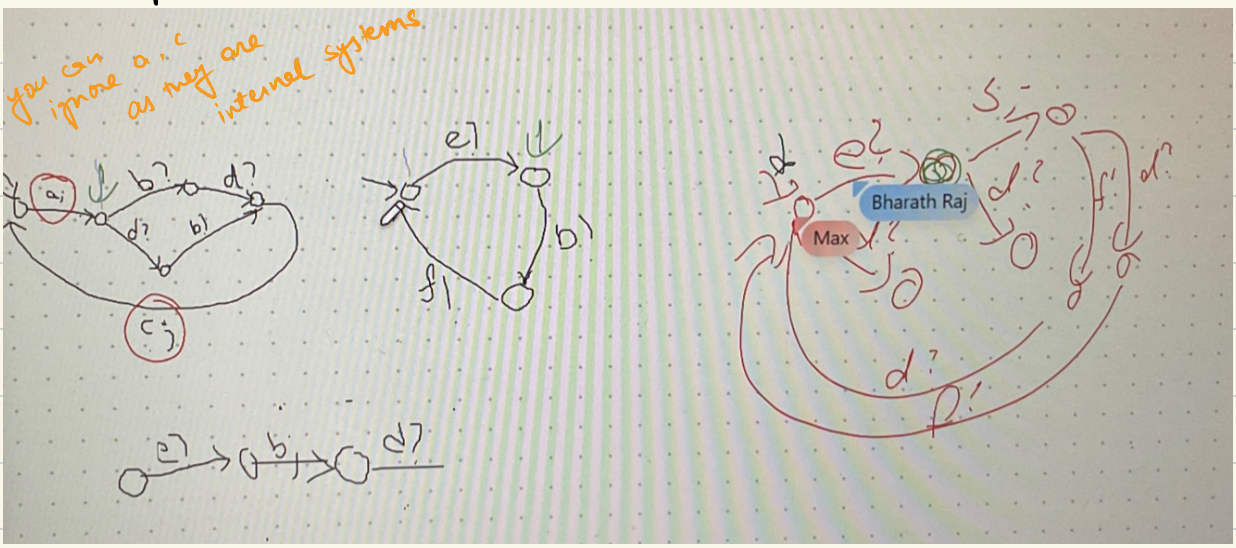


parking management system

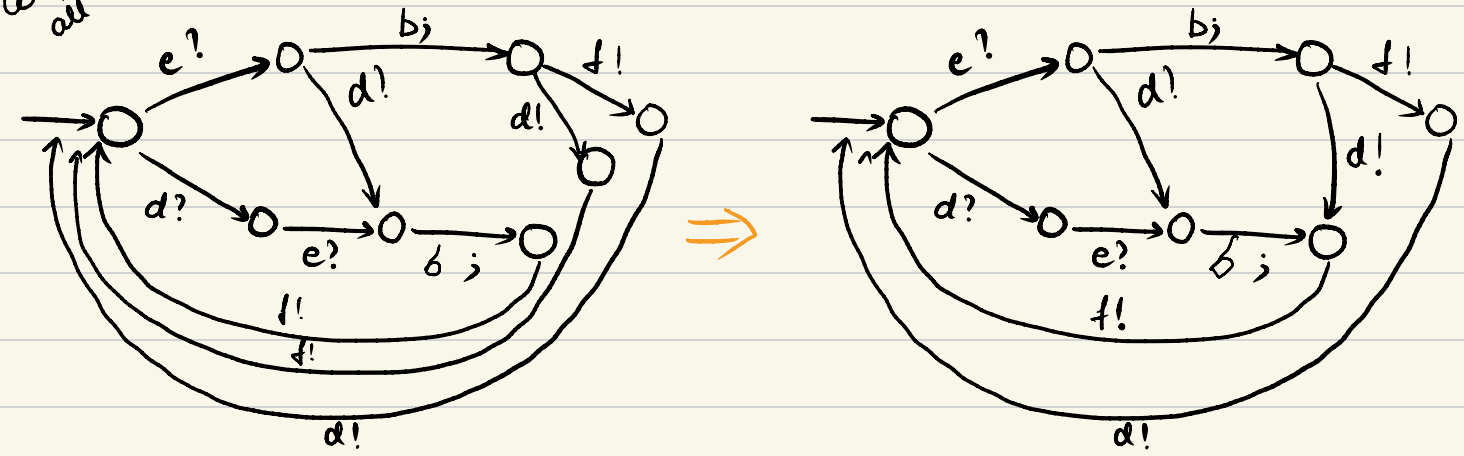




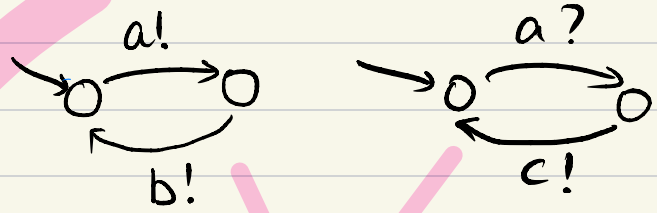
Composition

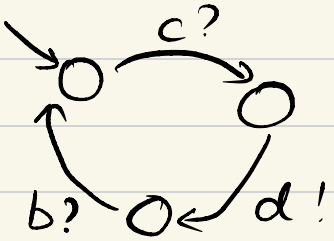


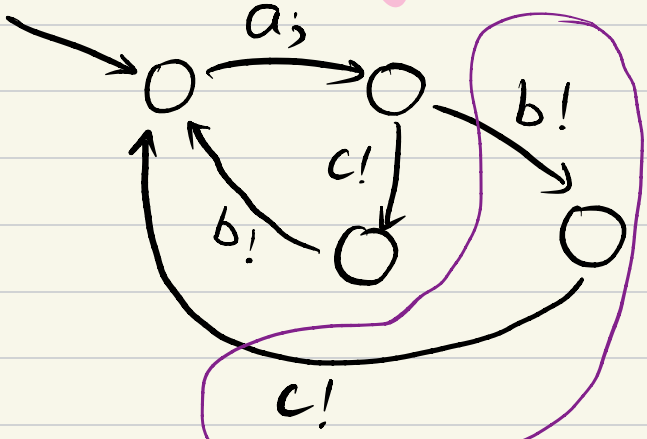
composition of all 3 loops:

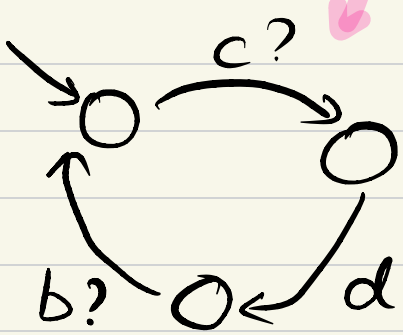


g)

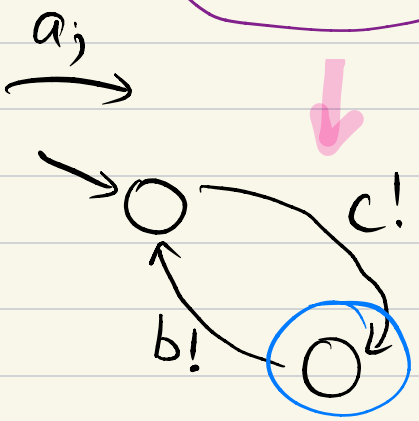


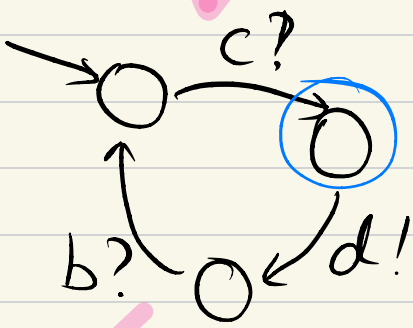


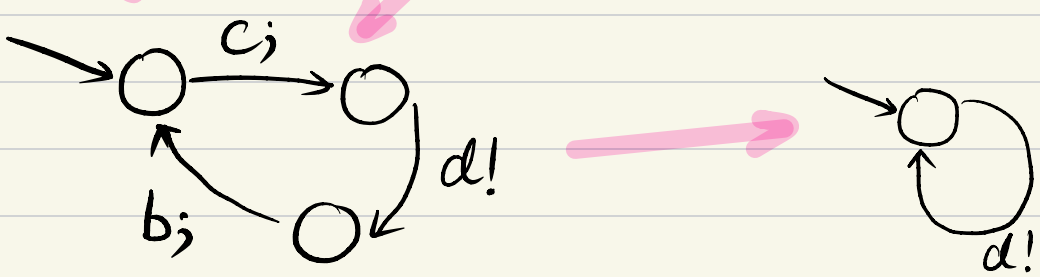




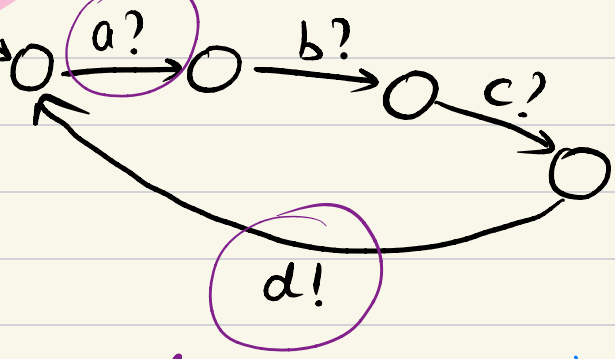
cancelled because b needs to be after c



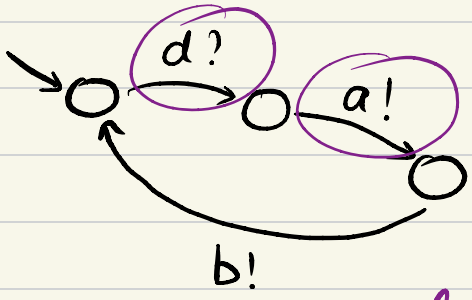




g)



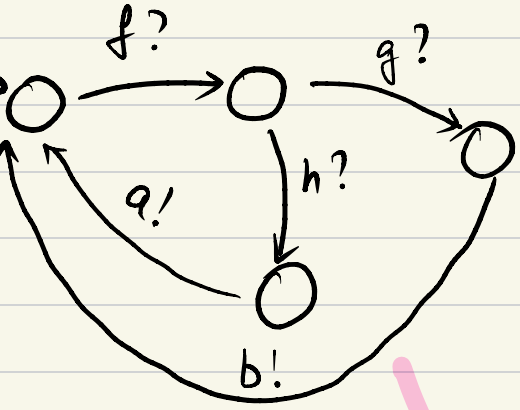
a < d

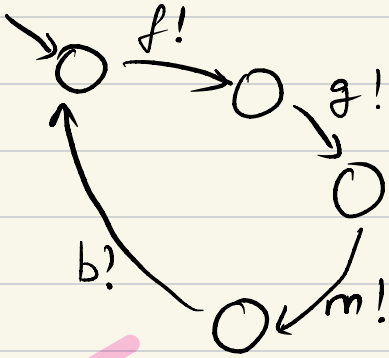


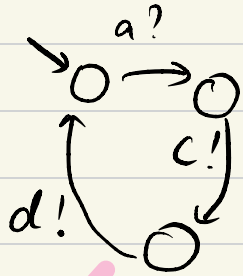
d < a

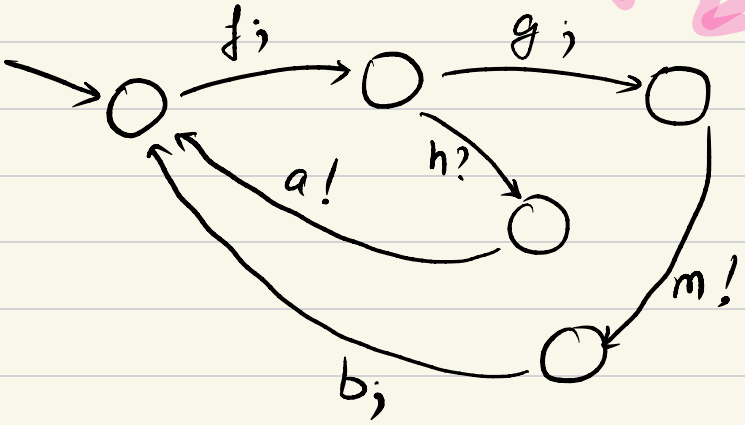
not composable

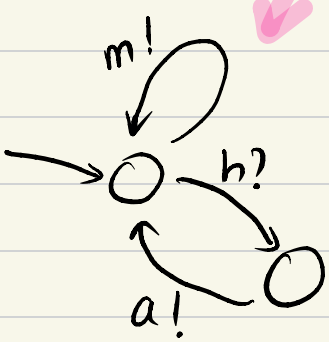
g)

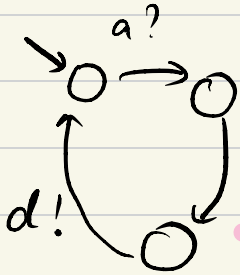


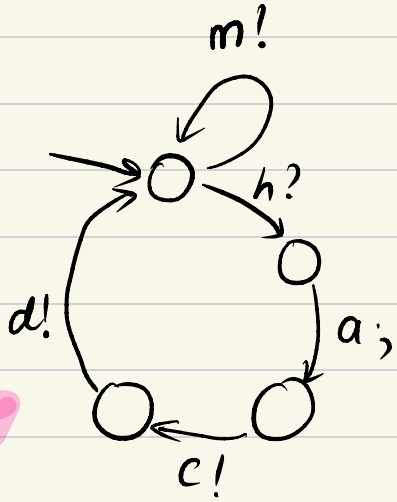




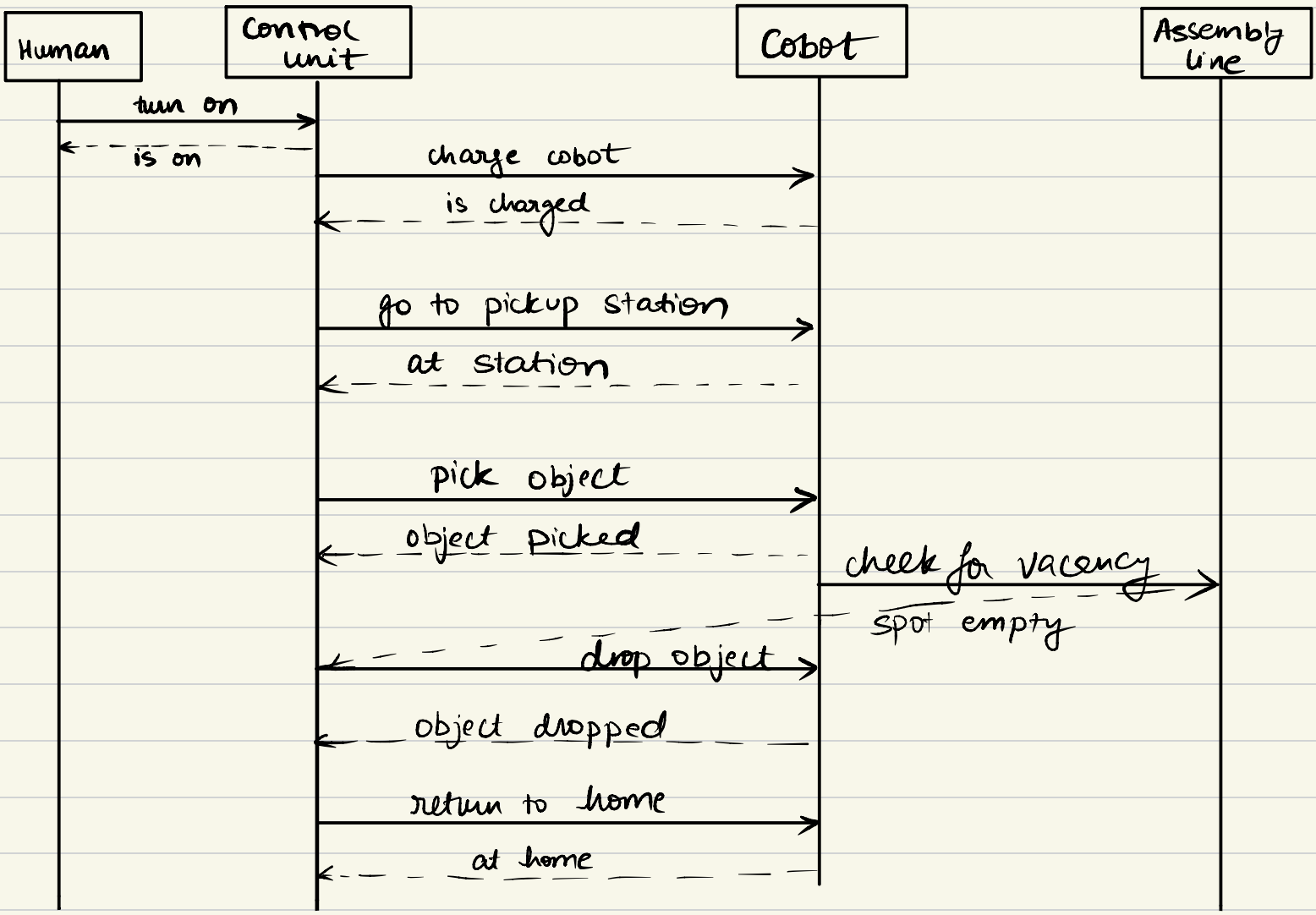








Sequence Diagram of a Cobot



4) a<b
a<c
b<d
c<d

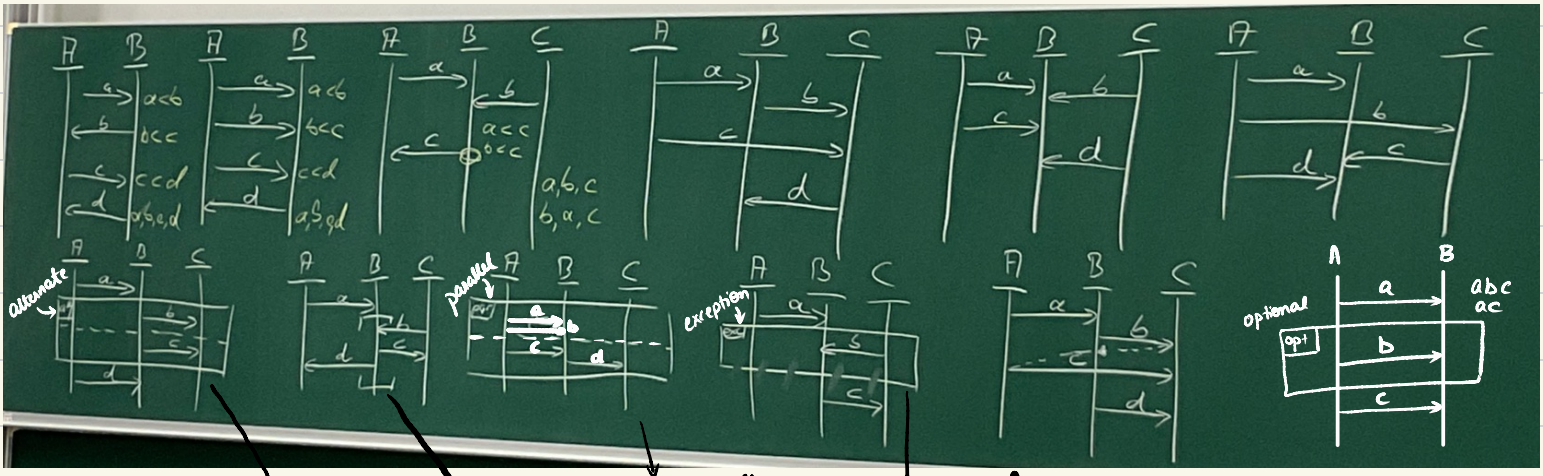
abcd
acbd

5) a<c
b<d

acbd bacd
abcd badc
abdc
bdac

6) a<b
a<d
b<c
b<d

abcd abdc



7) a<b
a<d
a<c

abd
acd
adb
adc

box has a choice: choose either above/below dotted line

8) a<d
a<c

abcd bacd
abdc badc
acbd
acdb
adbc
adcb

in this region order doesn't matter

9) a<b
c<d

abcd
acbd
acdb
cdab
cadb
cabd

both boxes can occur but you don't know which one

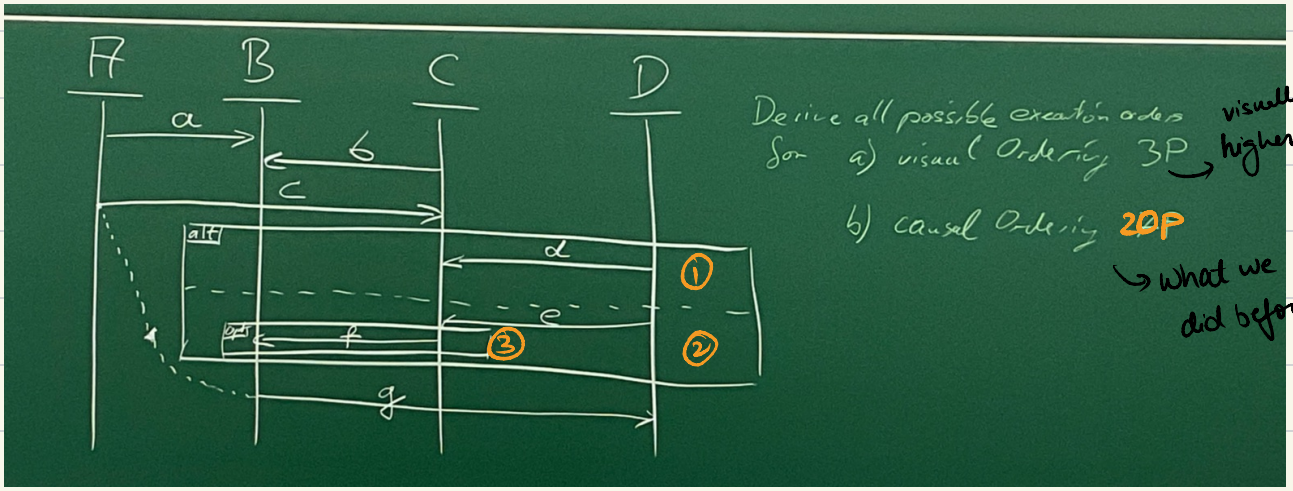
10) a<c
ab
b
ac

if b occurs, execution stops

11) a<d
a<b
a<c
b<d
b<c

abcd
abdc

8)



a) abcdg
abcefg
abceg

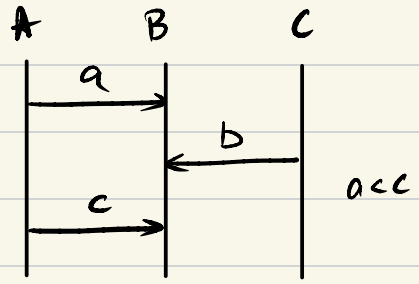
b) a<c
a<g
b<g
c<g
e<f
c<f

(1)
a,b,c,d,g
a,c,b,d,g
a,d,b,c,g
a,d,c,b,g
a,b,c,g,d
a,c,b,f,d
a,c,g,b,d

a,c,g,d,b
a,c,d,g,b
a,c,d,b,g
b,a,c,d,g
b,a,d,c,g
b,a,c,g,d

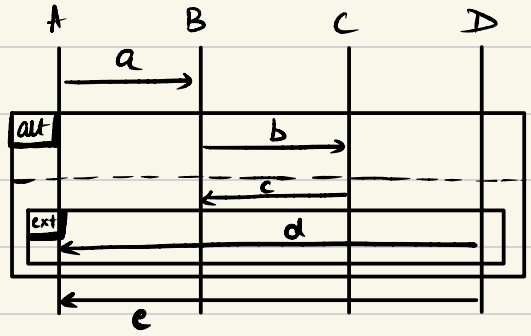
b,d,a,c,g
d,a,b,c,g
d,b,a,c,g
d,a,c,b,g
d,a,c,g,b

Q)



abc
bac
acb

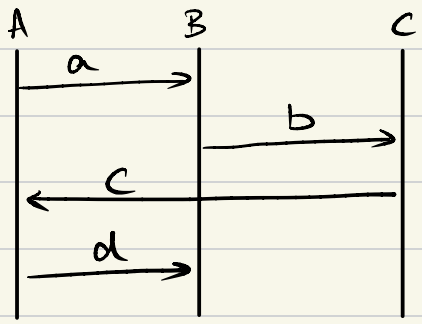
Q)



a<b
b<c
d<e

abe aeb
eab
acd d
ace cd
cad ad cae
eac cea eca

Q)



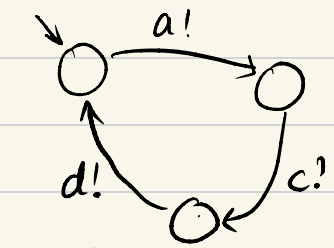
Translate into IA

a<b
b<c
a<d
c<d

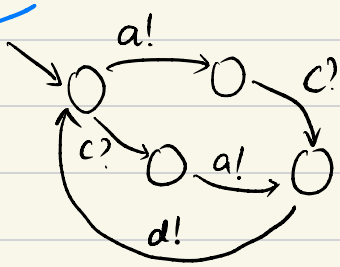
abcd

3 lifelines ⇒ 3 IA

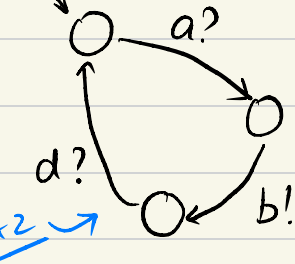
A]



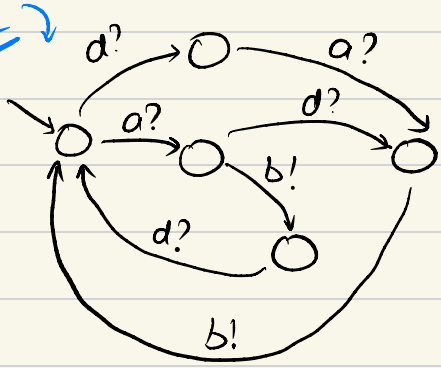
Alt 2
Alt 1



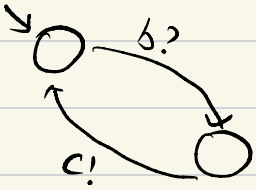
B]



Alt 2
Alt 1

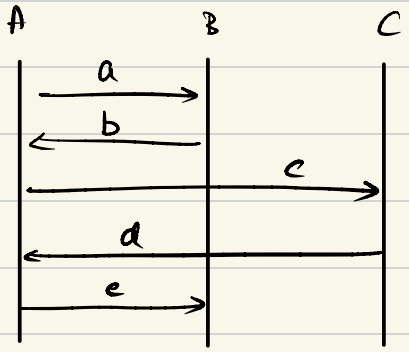


C]



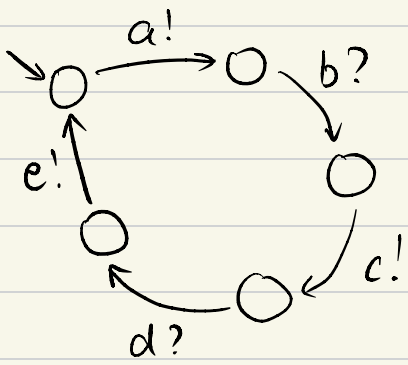
Alt 2 is enough for the exam.

Q)

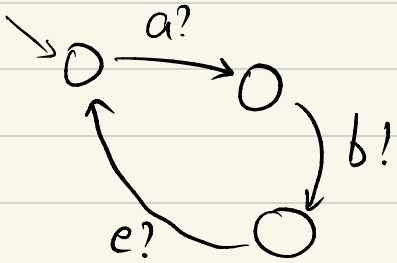


Translate into IA

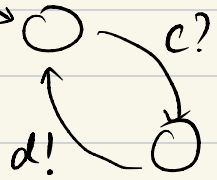
A]



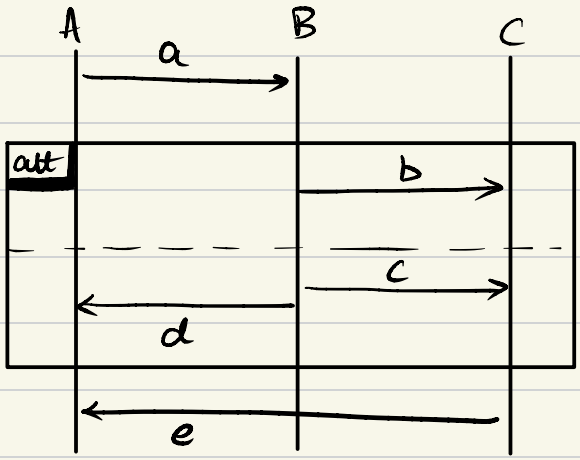
B]



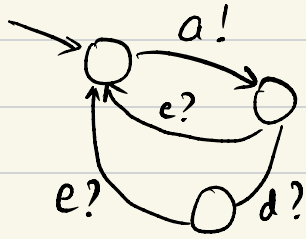
C]



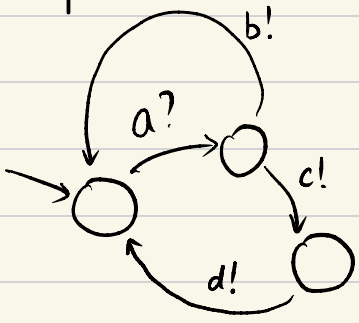
Q)



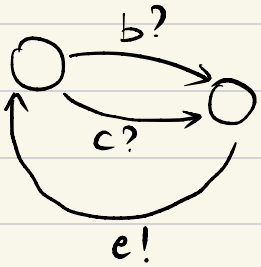
A]



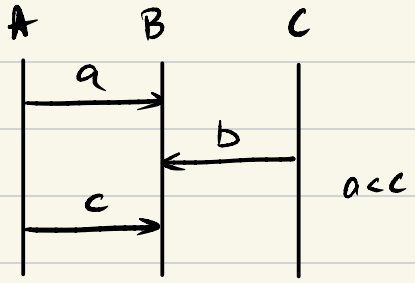
B]



C]



Q)

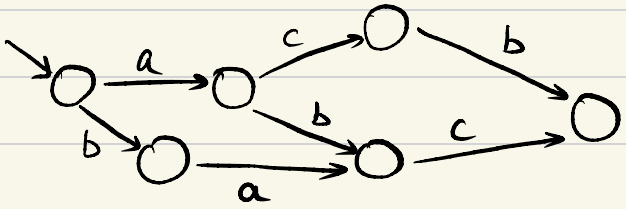


abc
bac
acb

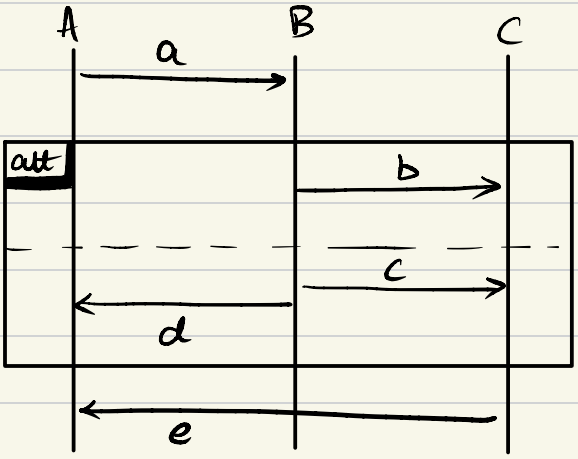
acc

finite state machine

Transform into FSM

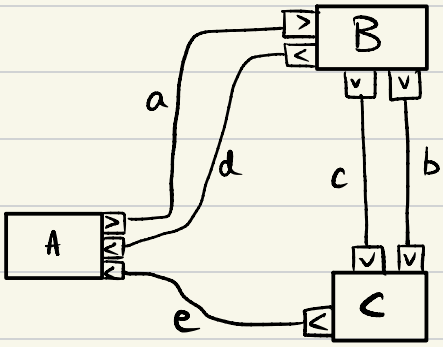


Q)



internal block diagram

Transform into IBD



EMBEDDED SYSTEM

15P System layers MAPE loop

15P { BDD, IBD, Content

81/90 for 1.0

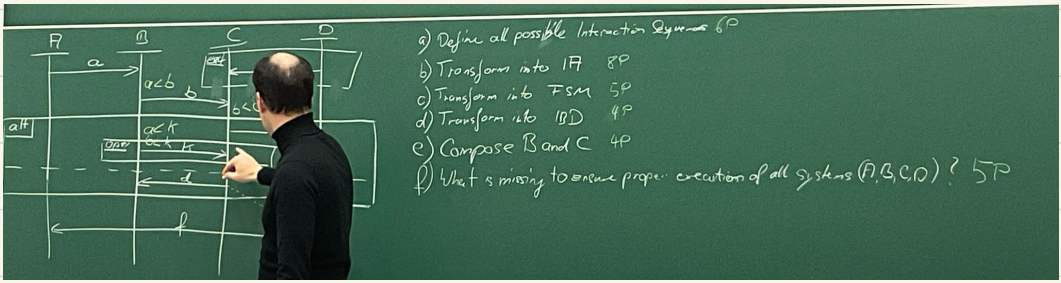
15 Petri nets — Model | Analyse

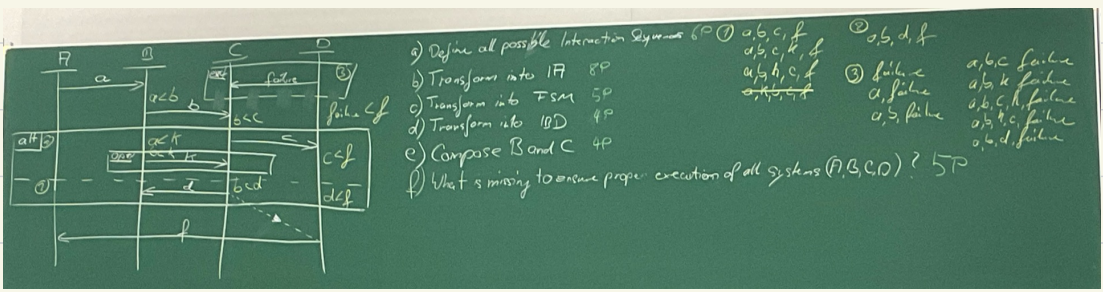
15 IA — Model | Compose | transform to SD

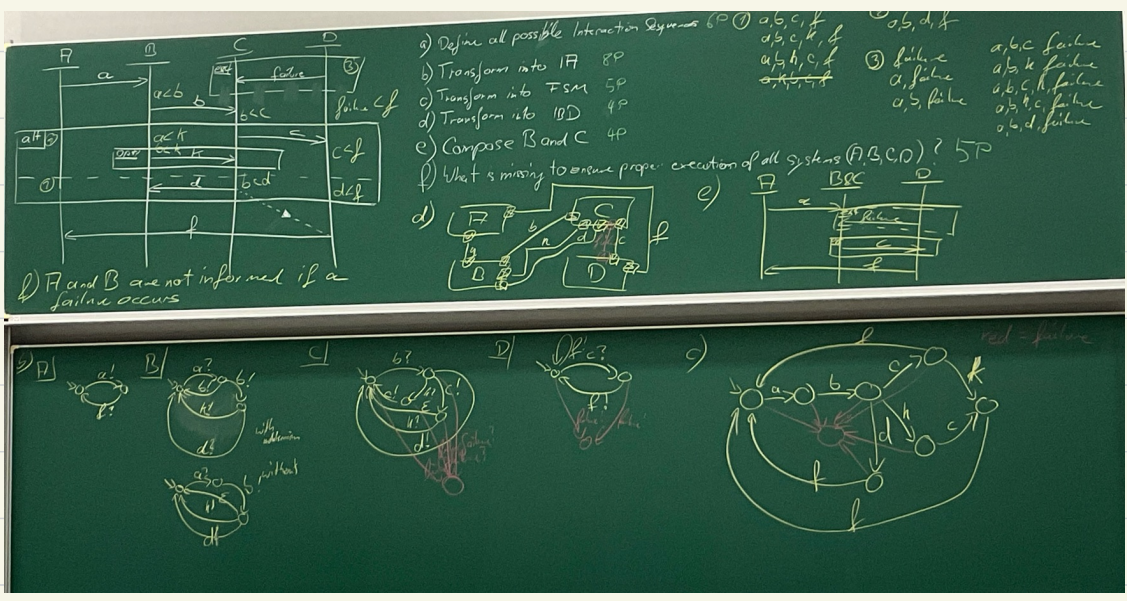
15 Sequence diagrams — Model | Analysis | transform to IA, FSM, IBD

15 field buses

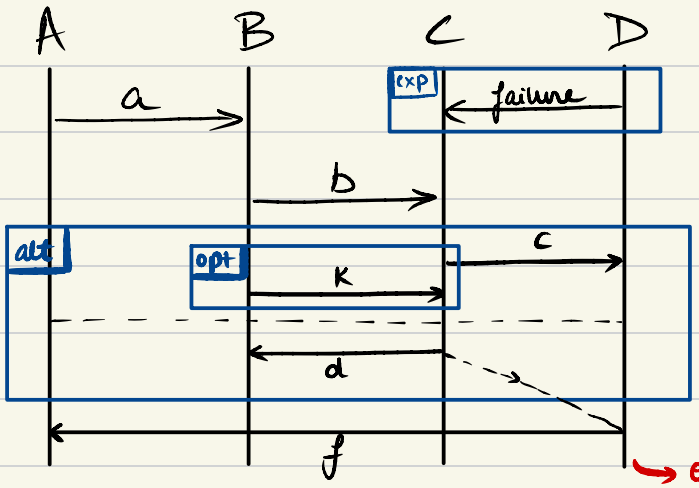
3 big modelling questions maybe + conversions


A
B
C
D
a
a<b
b
b<c
alt
opt
a<k
k
d
f
a) Define all possible Interaction Sequences 6P
b) Transform into IA 8P
c) Transform into FSM 5P
d) Transform into IBD 4P
e) Compose B and C 4P
f) What is missing to ensure proper execution of all systems (A,B,C,D)? 5P


A
B
C
D
a
a<b
b
b<c
failure
alt
opt
a<k
k
d
b<c!
c!f
d!f
f
a) Define all possible Interaction Sequences 6P
b) Transform into IA 8P
c) Transform into FSM 5P
d) Transform into IBD 4P
e) Compose B and C 4P
f) What is missing to ensure proper execution of all systems (A,B,C,D)? 5P
① a,b,c,f
a,b,c,k,f
a,b,k,c,f
② a,b,d,f
③ failure
a, failure
a,b, failure
a,b,c failure
a,b,k failure
a,b,c,k failure
a,b,k,c failure
a,b,d failure


A
B
C
D
a
a<b
b
b<c
failure
alt
opt
a<k
k
d
b<c!
c!f
d!f
f
a) Define all possible Interaction Sequences 6P
b) Transform into IA 8P
c) Transform into FSM 5P
d) Transform into IBD 4P
e) Compose B and C 4P
f) What is missing to ensure proper execution of all systems (A,B,C,D)? 5P
① a,b,c,f
a,b,c,k,f
a,b,k,c,f
② a,b,d,f
③ failure
a, failure
a,b, failure
a,b,c failure
a,b,k failure
a,b,c,k failure
a,b,k,c failure
a,b,d failure
d)
e)
B&C
failure
f) A and B are not informed if a failure occurs

a)
A
a!
b!
B
a?
b!
k!
d!
c!
b?
c!
d!
D
c?
f!
failure!
c)
f
a
b
c
k
d
h
c
k
k
red = failure



$a<k$
$a<b$ $\quad c<f$
$b<c$
$b<k$
$failure<f$
$d<f$
$b<d$

→ only look at D for failure order. failure occurs before f. BUT c and d dont have any order w.r.t failure

1) Define all possible Interaction system (SP)

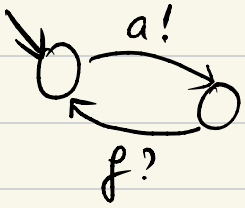
abcf
abckf
abkcf

failure
a failure
ab failure
abc failure
abck failure
abkc failure
abk failure

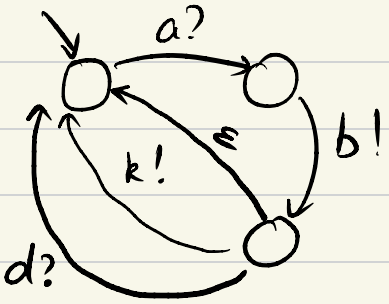
abdf
abd failure

2) Transform into IA

A]

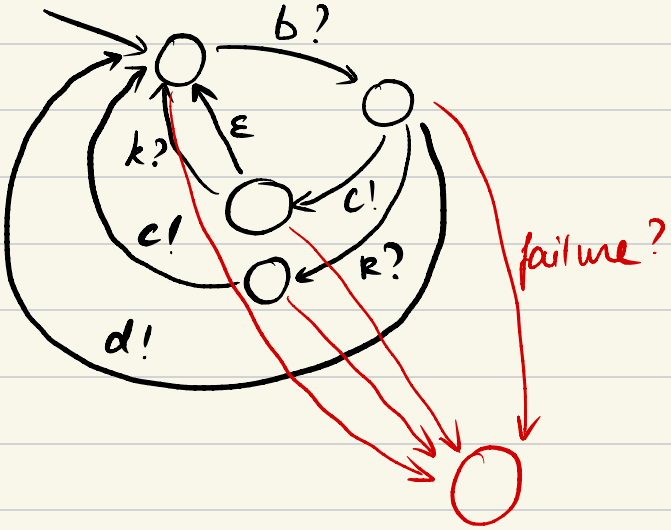


B]

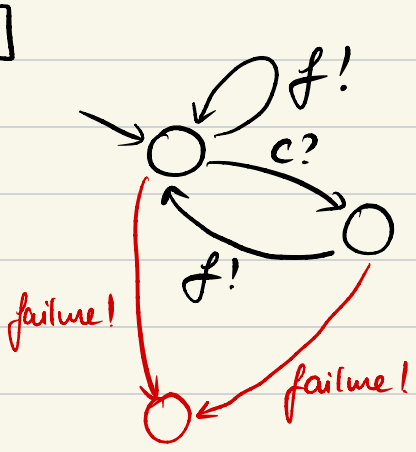


ε - goes directly, takes no time

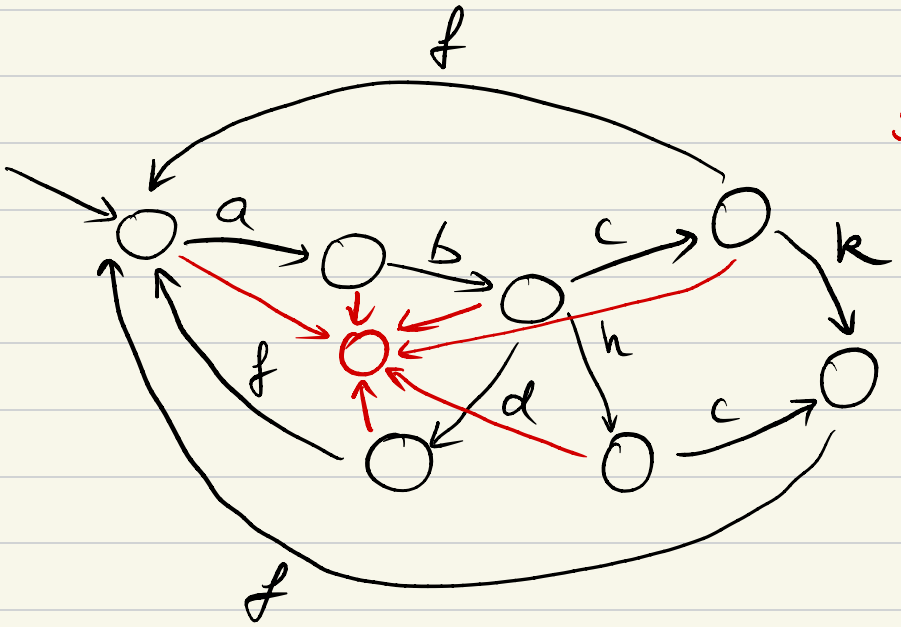
C]



D]

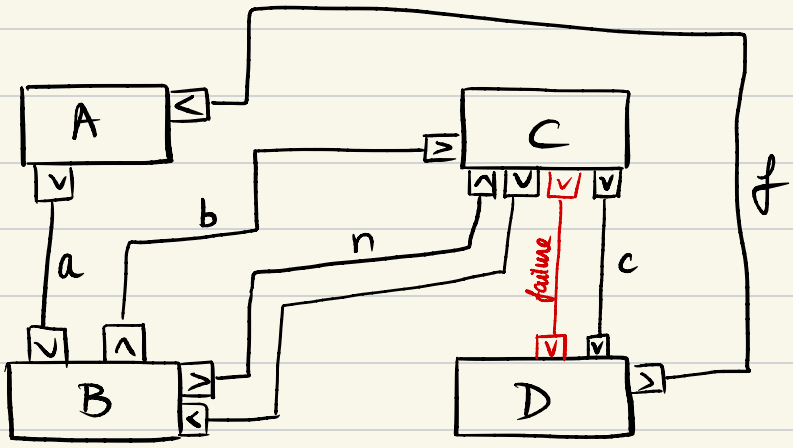


3)

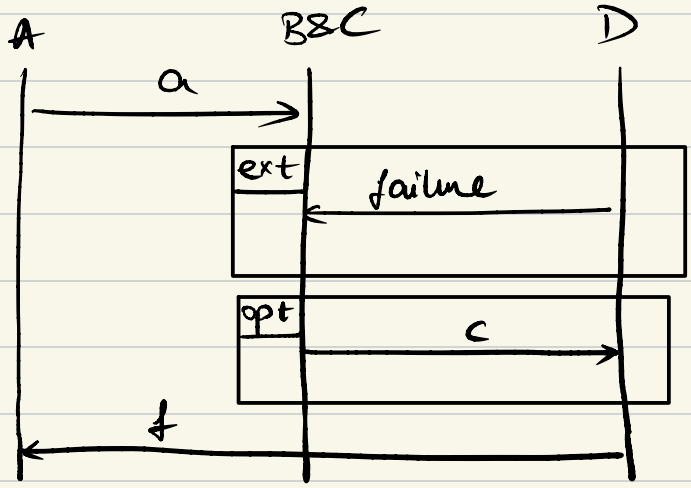


red = failure

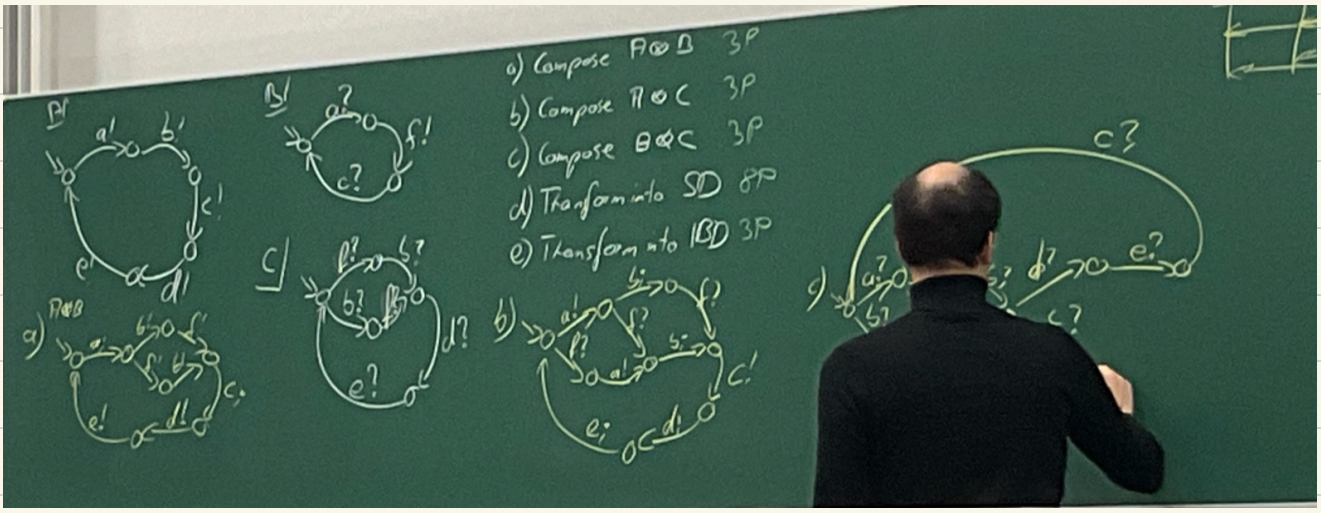
4)



5)

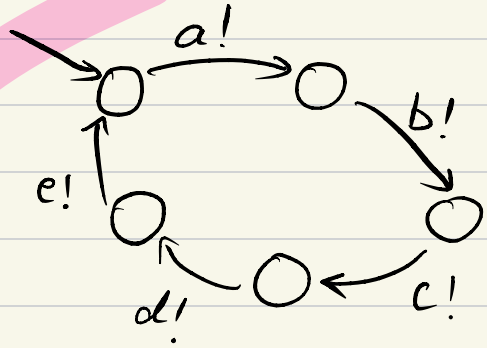


6) A and B are not informed if a failure occurs.

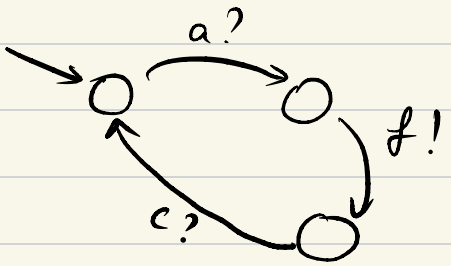

a) Compose A ⊗ B 3P
b) Compose Π ⊗ C 3P
c) Compose Θ ⊗ C 3P
d) Transform into SD 8P
e) Transform into IBD 3P

Q

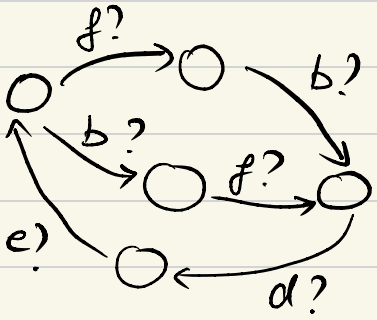
A]



B]



C]



① Compose A ⊗ B

② Compose A ⊗ C

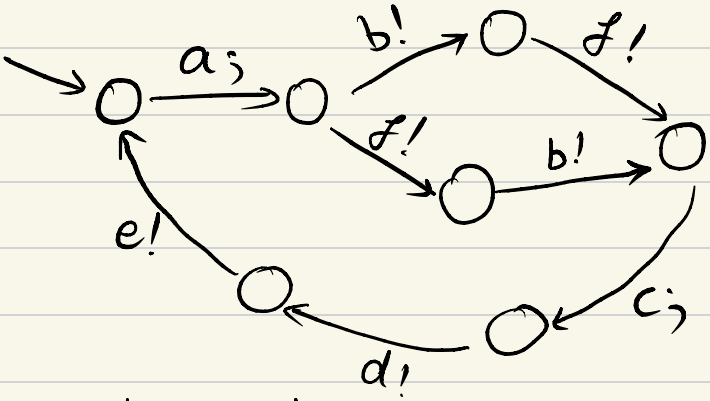
③ Compose B ⊗ C

④ Transform into SD

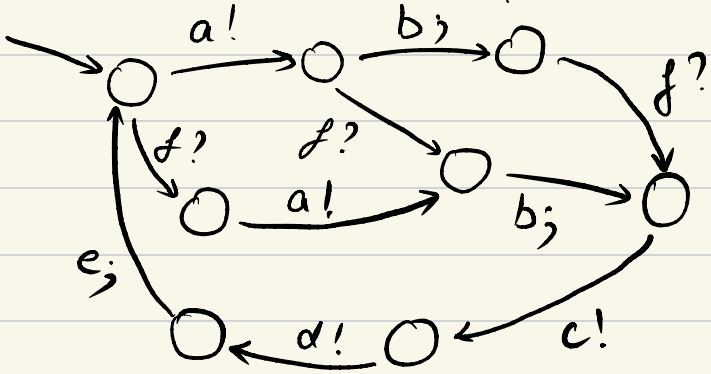
⑤ Transform into IBD

Ans

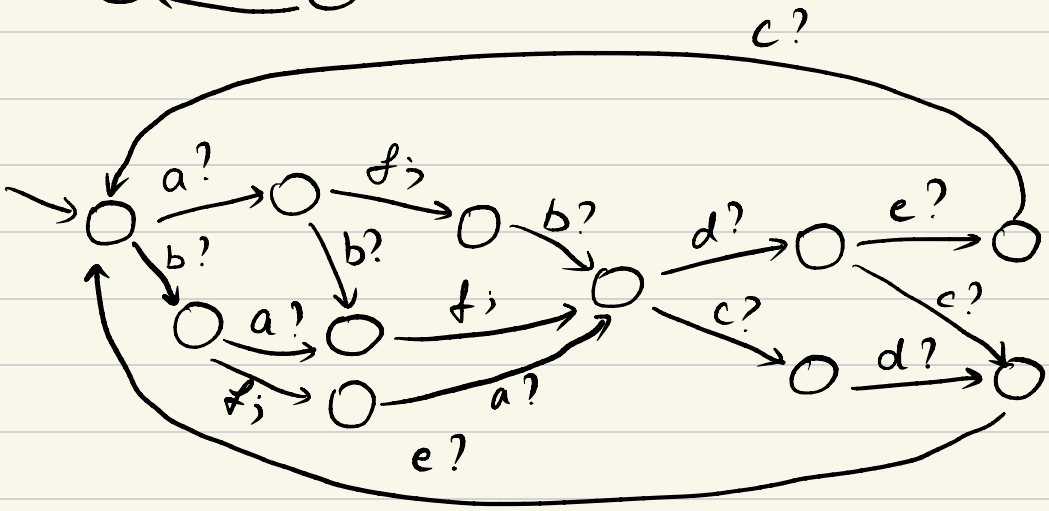
①

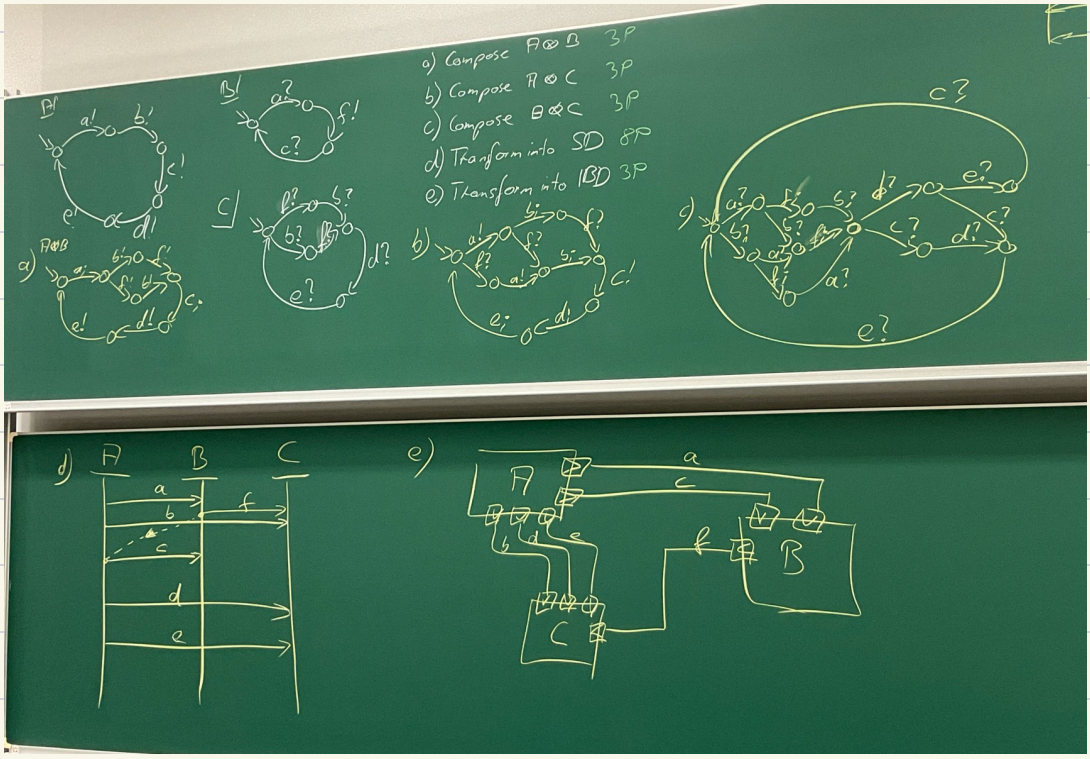


②



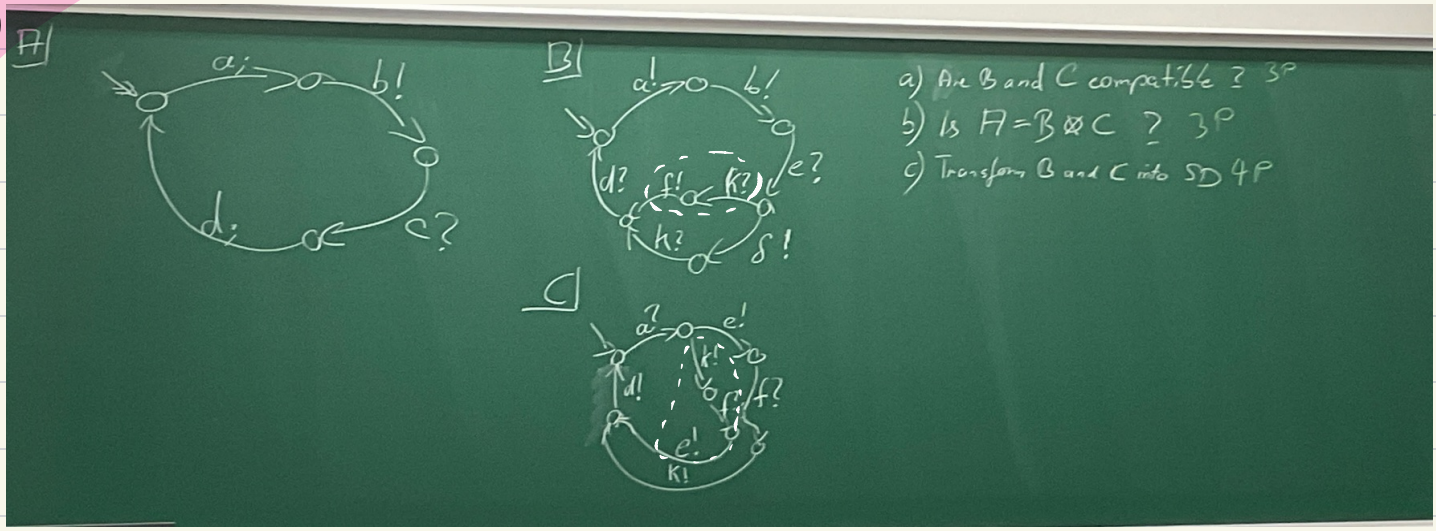
③



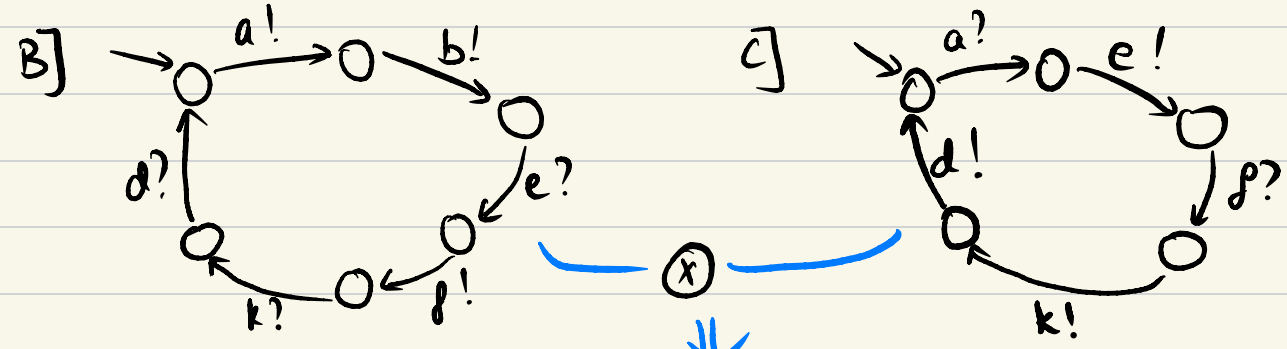

a) Compose A⊗B 3P
b) Compose A⊗C 3P
c) Compose B⊗C 3P
d) Transform into SD 8P
e) Transform into IBD 3P
A
B
C
a) A⊗B
b)
c)
d)
e)

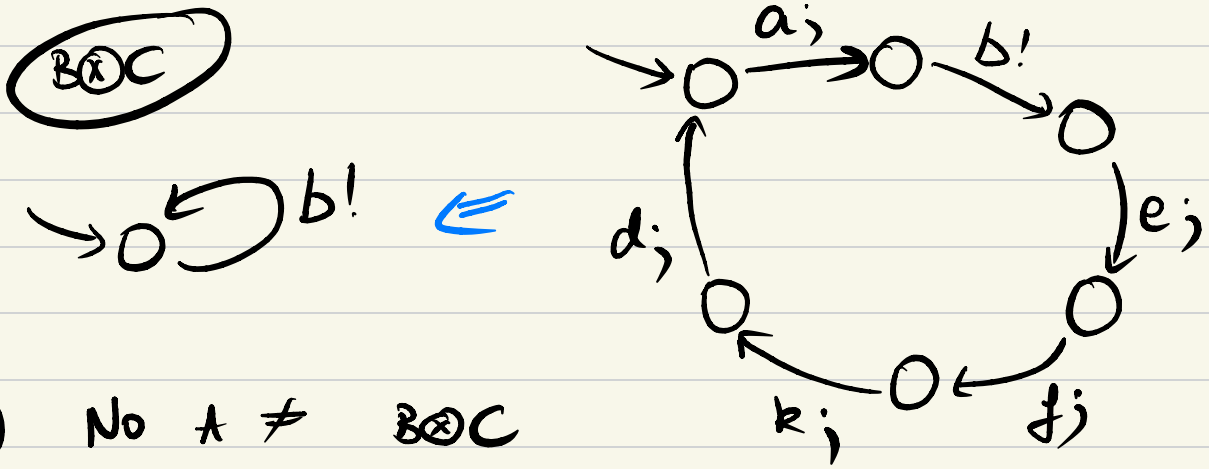
d)
A
B
C
a
b
c
d
e
f
e)
A
B
C
a
c
b
d
e
f

8)


A]
a;
b!
d;
c?
B]
a!
b!
d?
f!
k!
e?
h?
g!
C]
a?
e!
k!
d!
f?
e!
k!
a) Are B and C compatible ? 3P
b) Is A = B ⊗ C ? 3P
c) Transform B and C into SD 4P

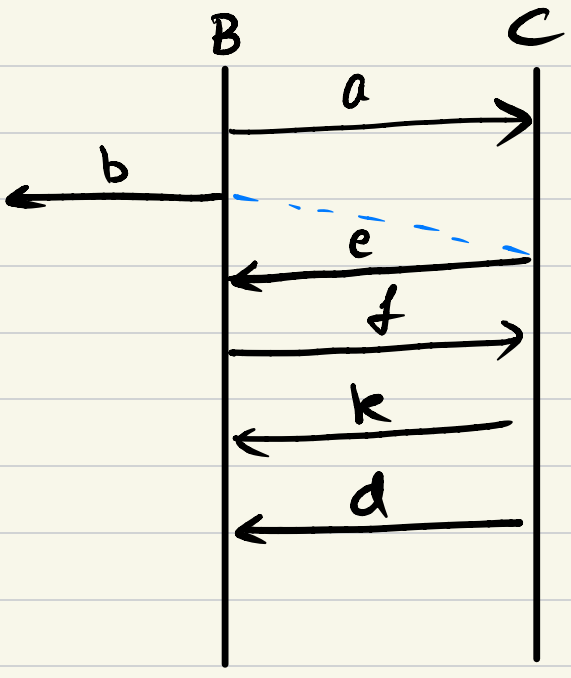
(a)


B]
a!
b!
d?
e?
k?
g!
C]
a?
e!
d!
g?
k!
X

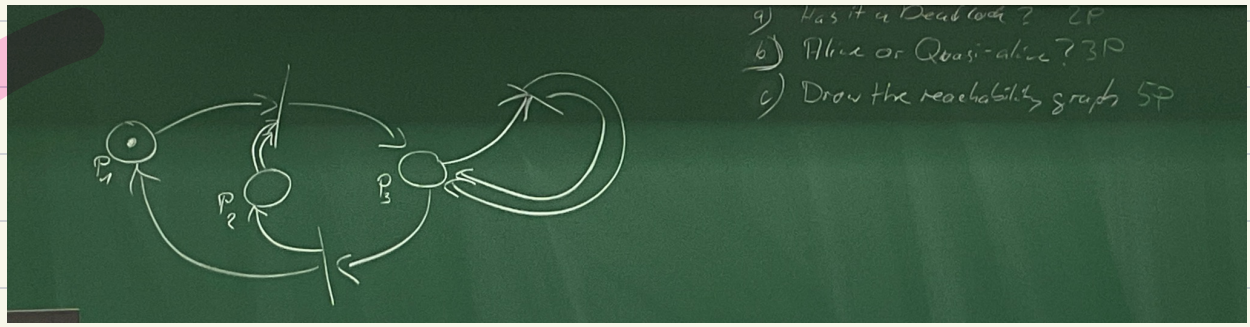

B⊗C
b!
a;
b!
e;
d;
k;
f;

(b) No A ≠ B⊗C

(c)


B
C
a
b
e
f
k
d

9)

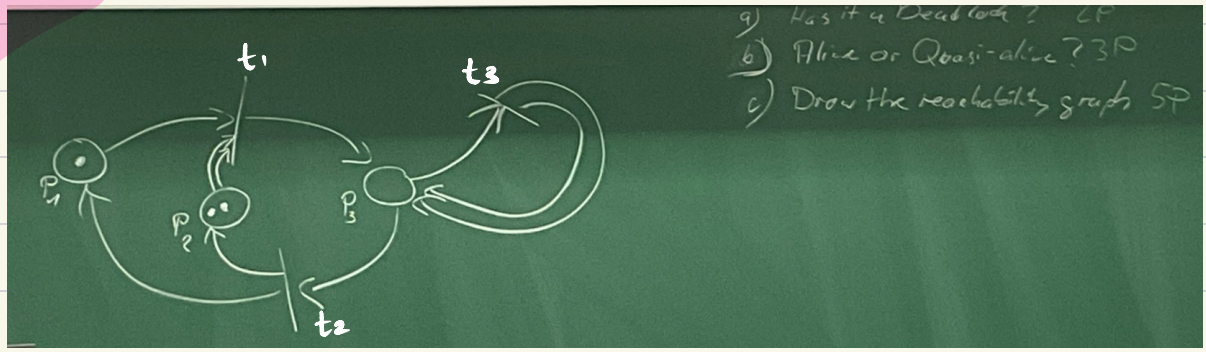


(a) Fully dead - it has a deadlock

(b) Dead -

(c) 100

8



(a) Yes, after t_1, t_2

(b) No, it will be dead
if there's even one possibility to reach a dead loop, its dead, not quasialive

(c) $(1,2,0)$ impossible due to infinity

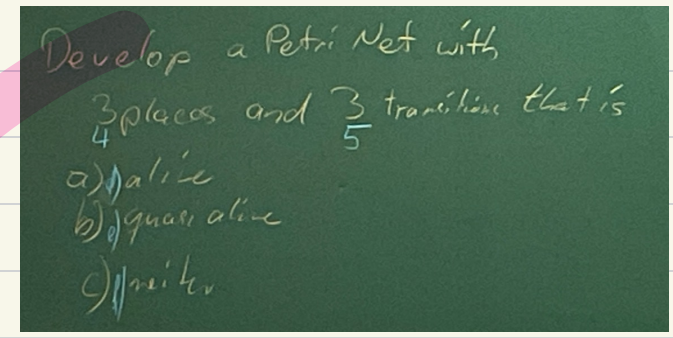
$(0,0,1) \rightarrow (0,0,2) \rightarrow (0,0,3) \rightarrow (0,0,4) \rightarrow \dots$

$(1,1,0) \rightarrow (1,1,1) \rightarrow (1,1,2) \rightarrow (1,1,3) \rightarrow \dots$

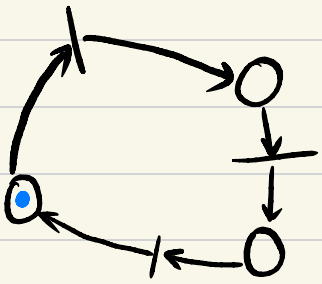
$(2,2,0)$ ....

$(1,0,1) \rightarrow (1,0,2) \rightarrow (1,0,3) \rightarrow \dots$

g)



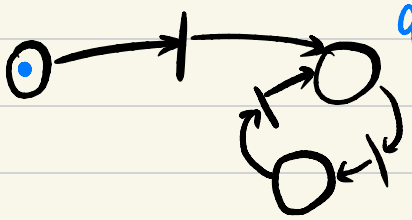
a)



alive

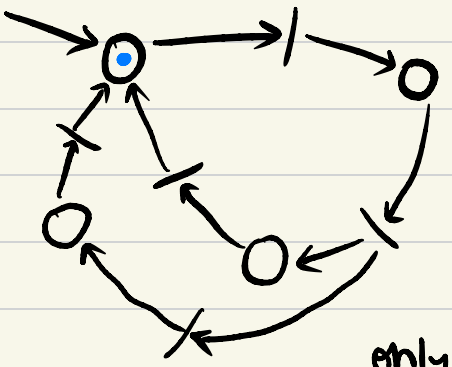
c)

b)



quasi alive

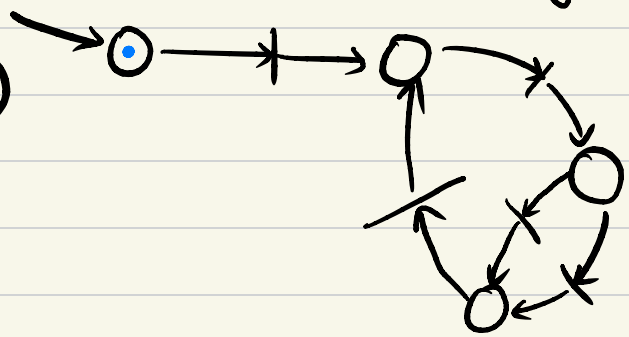
d)



alive

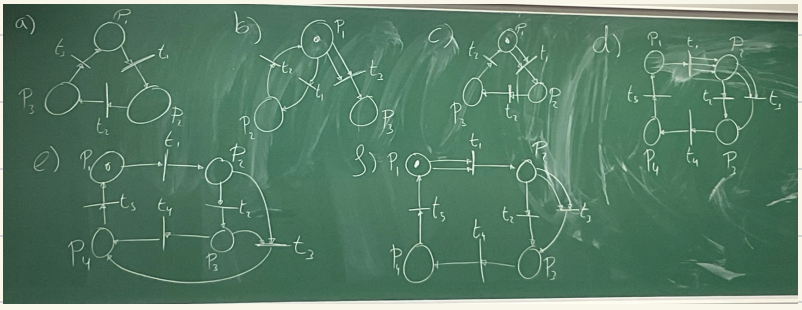
only transition can generate, not a place

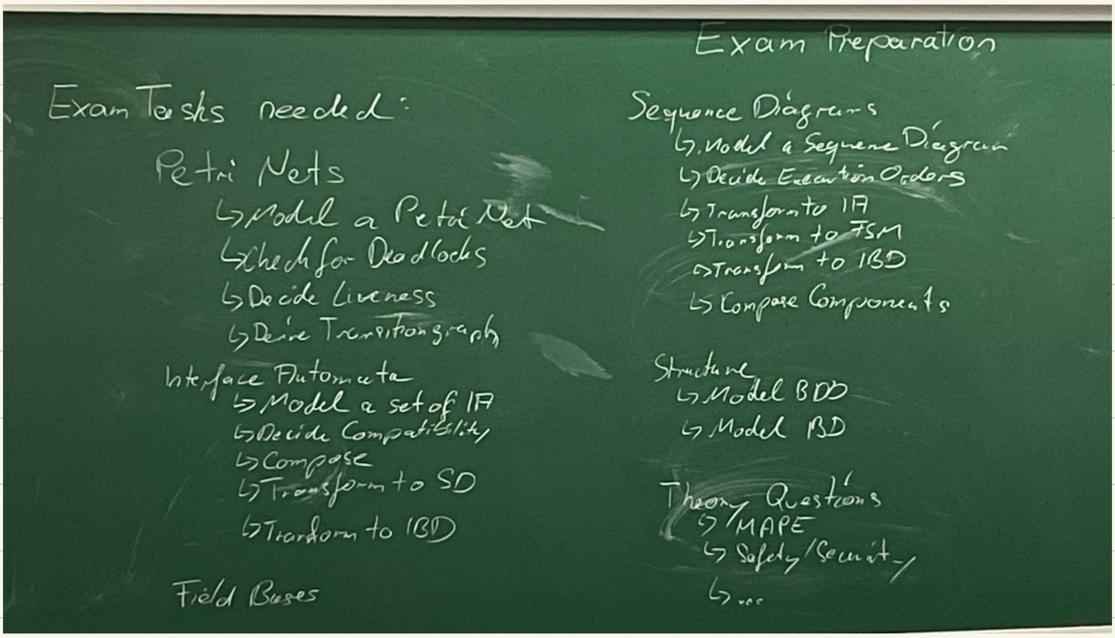
e)

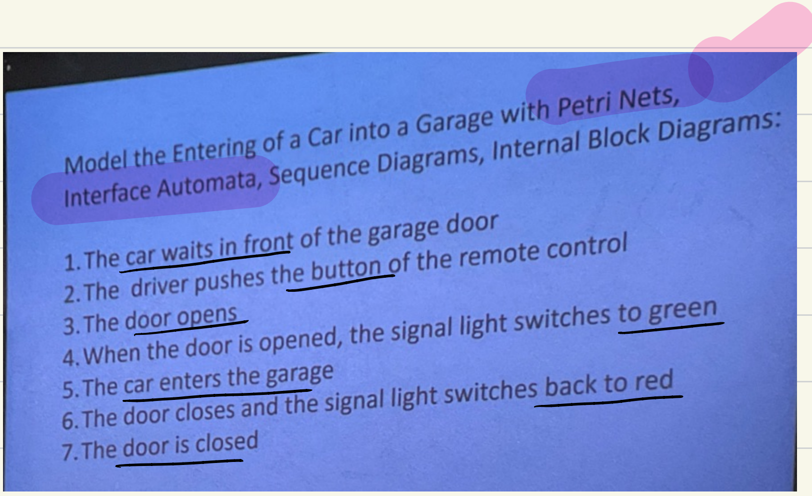


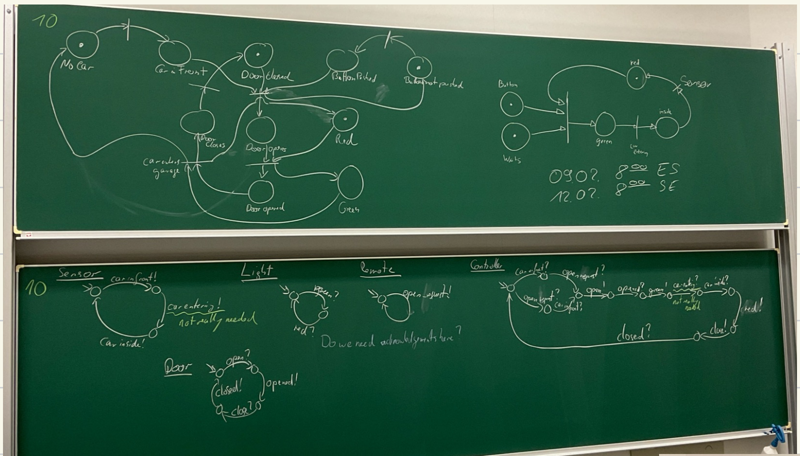
quasi alive

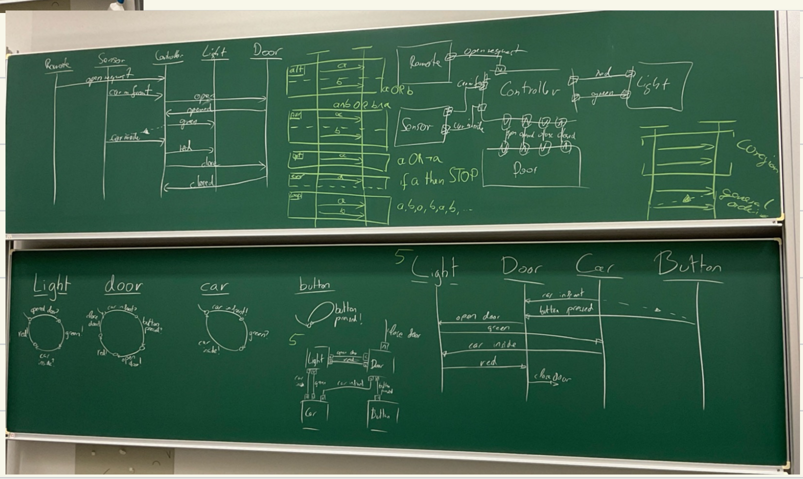
f)

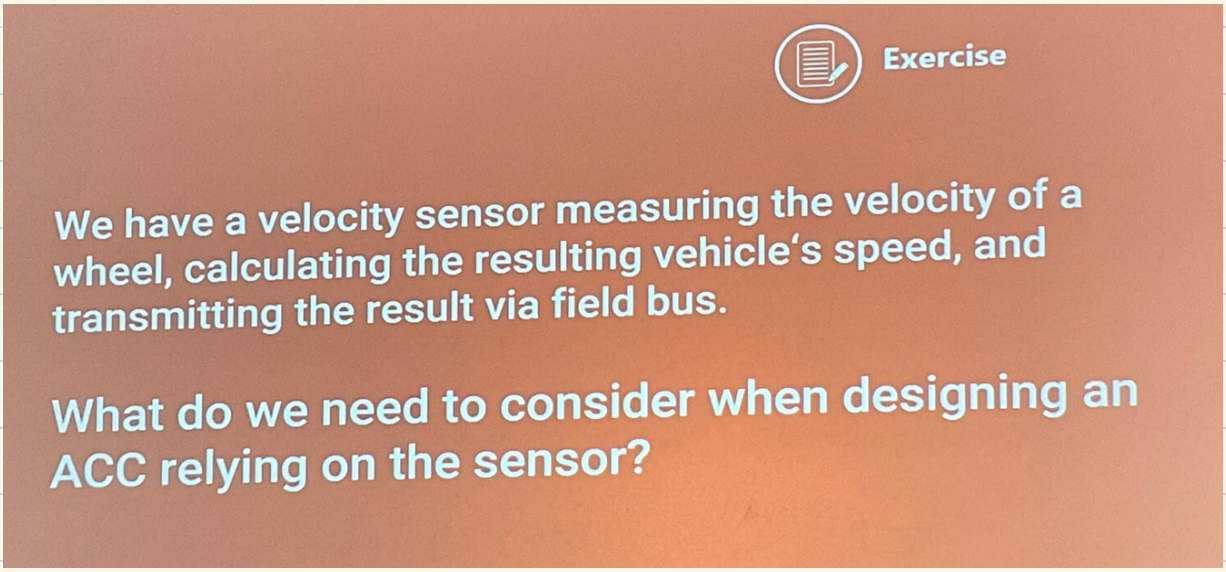



Exam Preparation
Exam Tasks needed:
Petri Nets
↳ Model a Petri Net
↳ Check for Deadlocks
↳ Decide Liveness
↳ Derive Transitiongraph
Interface Automata
↳ Model a set of IA
↳ Decide Compatibility
↳ Compose
↳ Transform to SD
↳ Transform to IBD
Field Buses
Sequence Diagrams
↳ Model a Sequence Diagram
↳ Decide Execution Orders
↳ Transform to IA
↳ Transform to FSM
↳ Transform to IBD
↳ Compare Components
Structure
↳ Model BDD
↳ Model IBD
Theory Questions
↳ MAPE
↳ Safety/Security
↳ ...


Model the Entering of a Car into a Garage with Petri Nets, Interface Automata, Sequence Diagrams, Internal Block Diagrams:
1. The car waits in front of the garage door
2. The driver pushes the button of the remote control
3. The door opens
4. When the door is opened, the signal light switches to green
5. The car enters the garage
6. The door closes and the signal light switches back to red
7. The door is closed


Sensor
Light
Remote
Door


Light
Door
Car
Button



→ accuracy and precision of the sensor

→ reliability

→ low noise, low reflection

→ good shielding

→ good environment / weather conditions

→ low cost

→ sensor placement and response time / real time constraints

→ cable arrangement

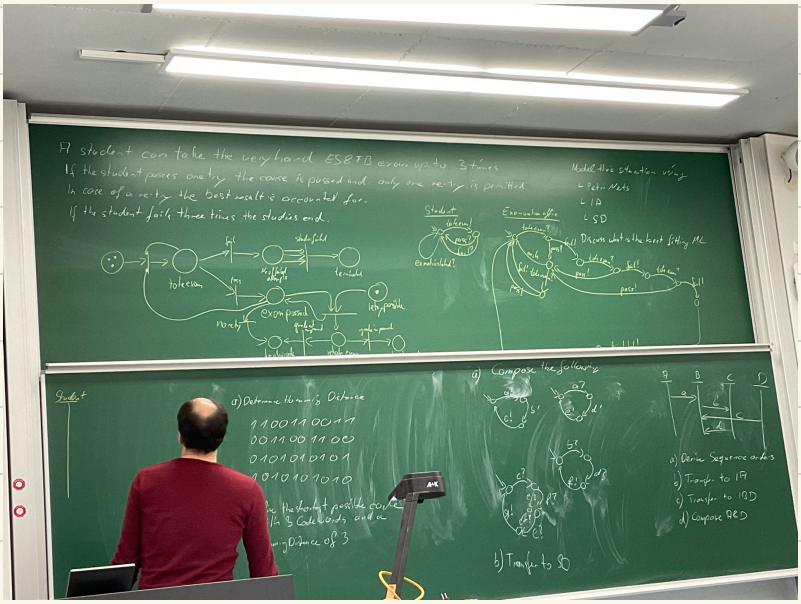
→ mode of data transmission

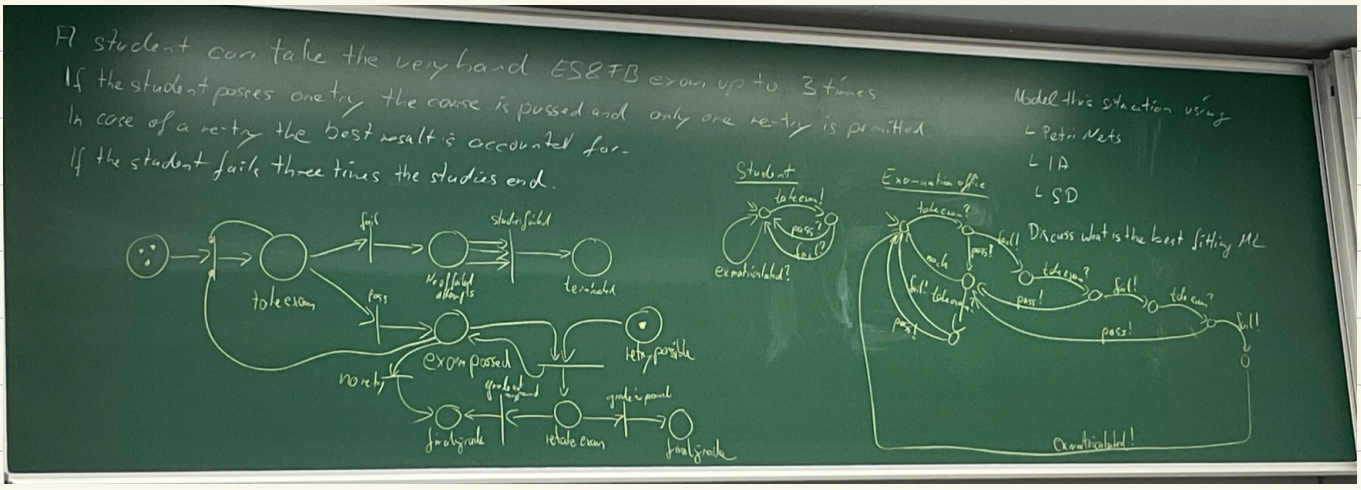
9

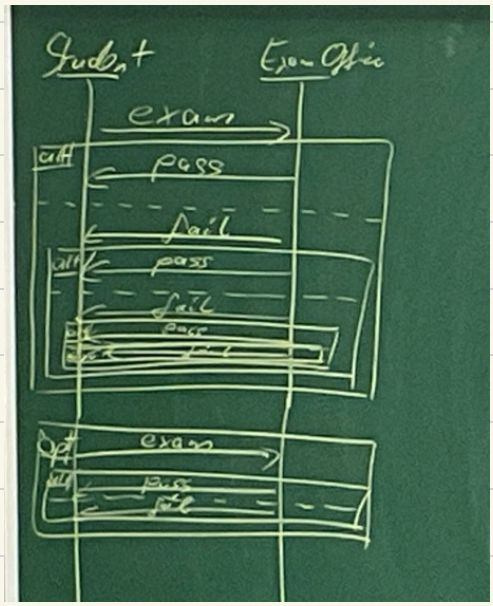
A student can take the very hard ES&FB exam up to 3 times
If the student passes one try the course is passed and only one re-try is permitted
In case of a re-try the best result is accounted for.
If the student fails three times the studies end.

Model this situation using
- Petri Nets
- IA
- SD

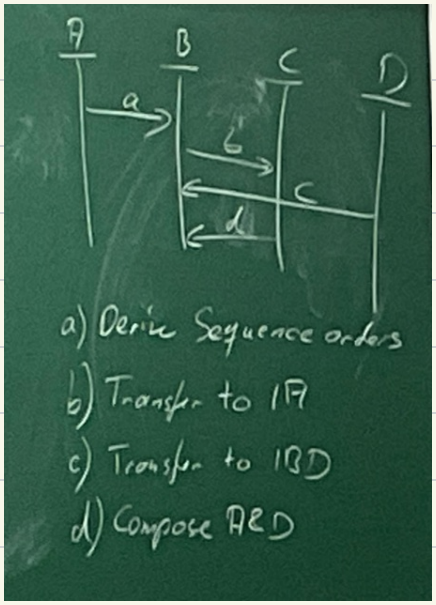
Discuss what is the best fitting ML







Q



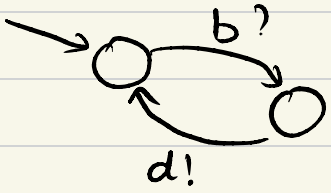
a < b
b < d

@ abcd
acbd
cabd

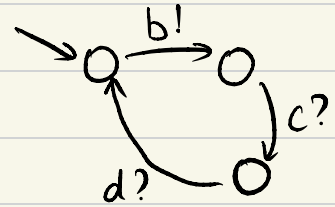
ⓑ

A]

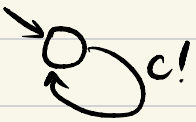
C]



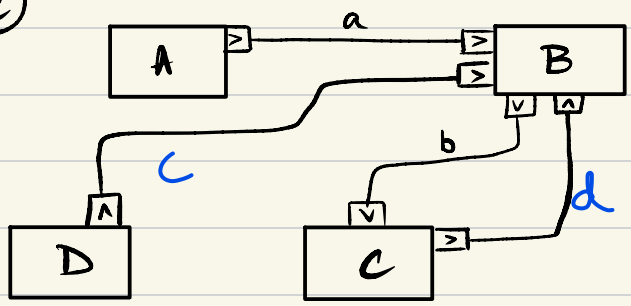
B]



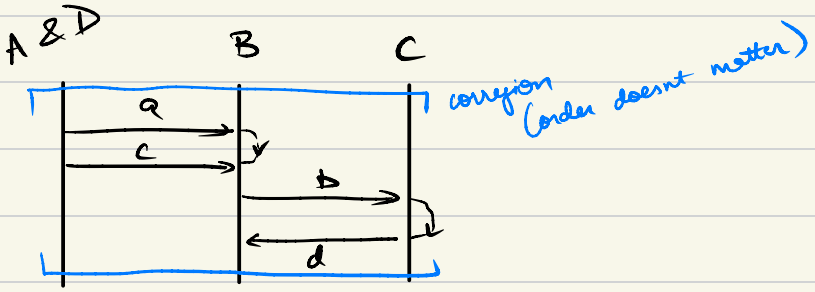
D]

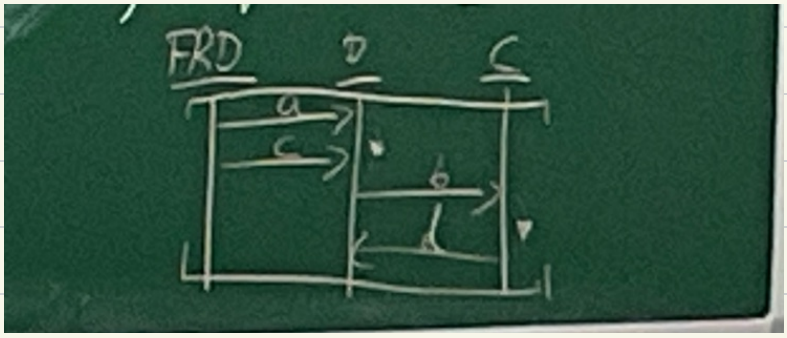


ⓒ

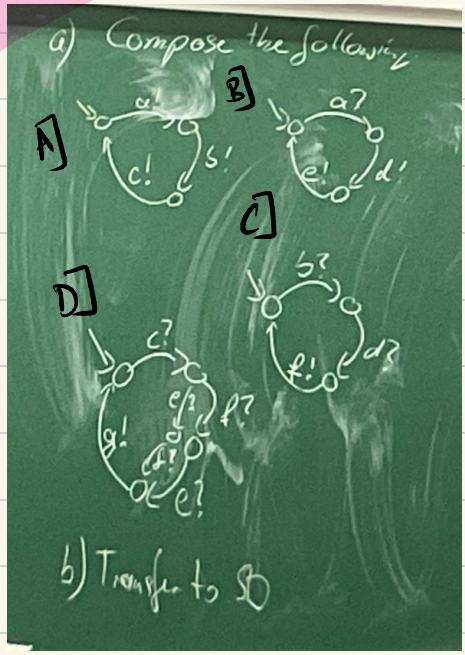


ⓓ

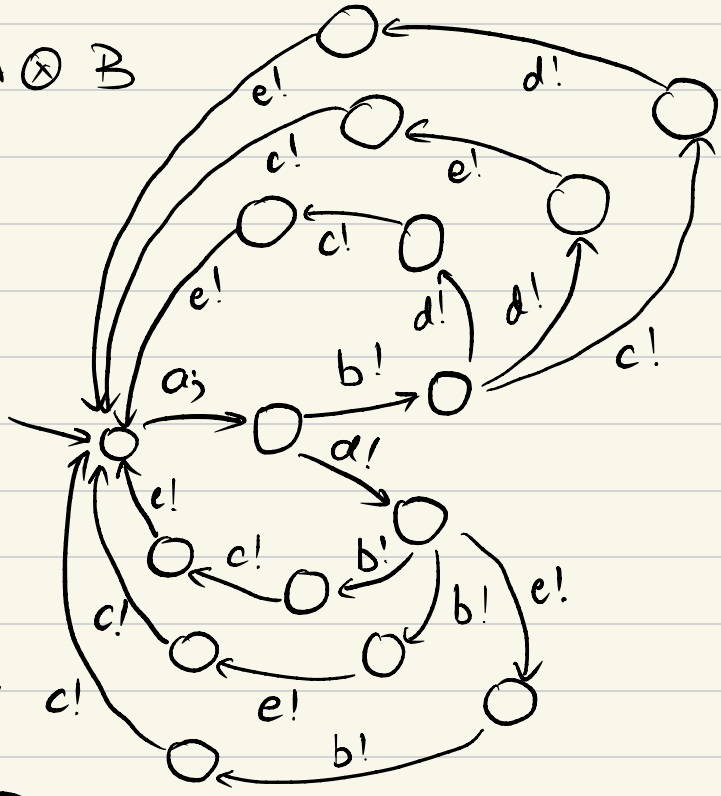


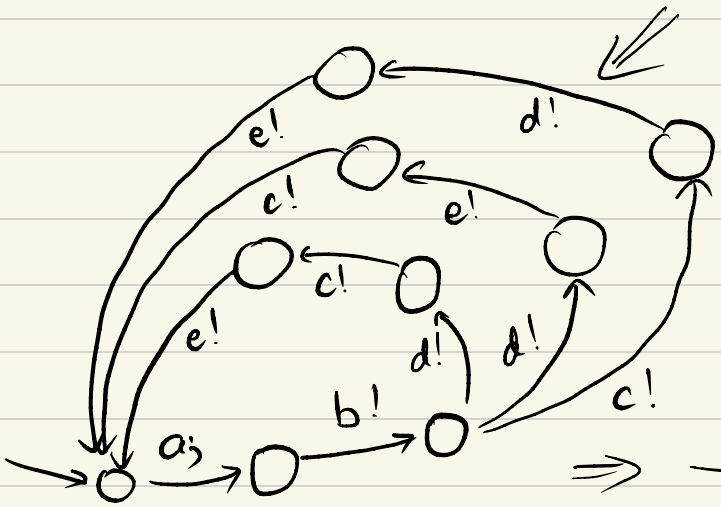


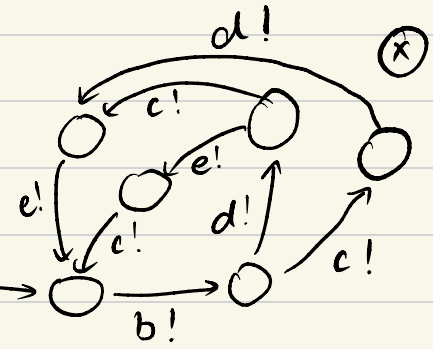
9

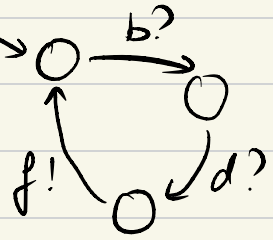

a) Compose the following
A]
B]
C]
D]
b) Transfer to SO

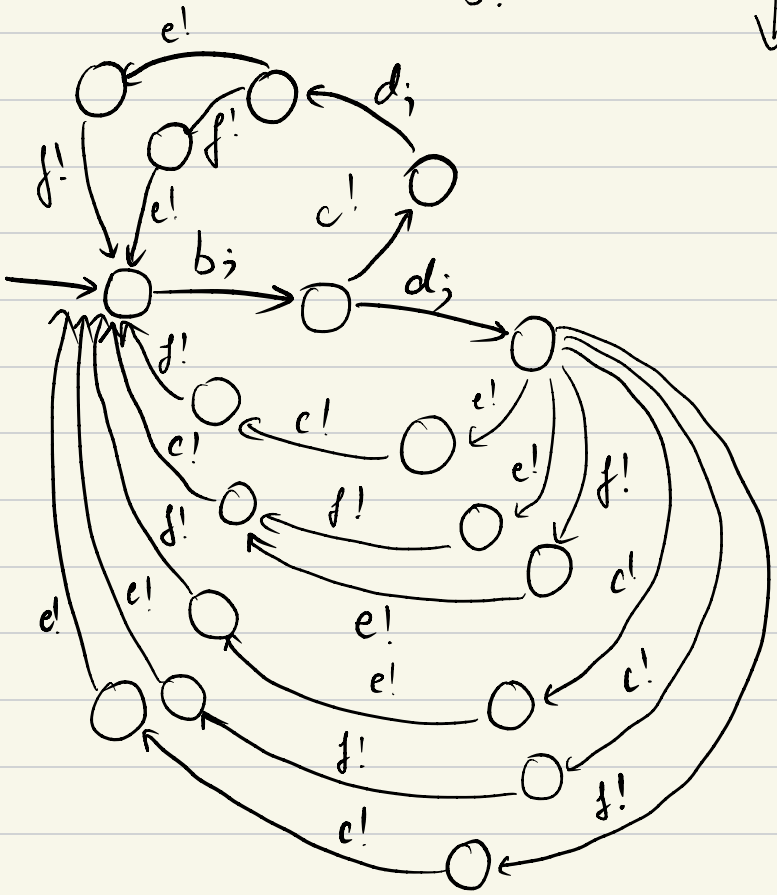
A ⊗ B


e!
d!
c!
e!
c!
e!
d!
d!
c!
a;
b!
a!
e!
c!
b!
c!
b!
e!
c!
e!
b!

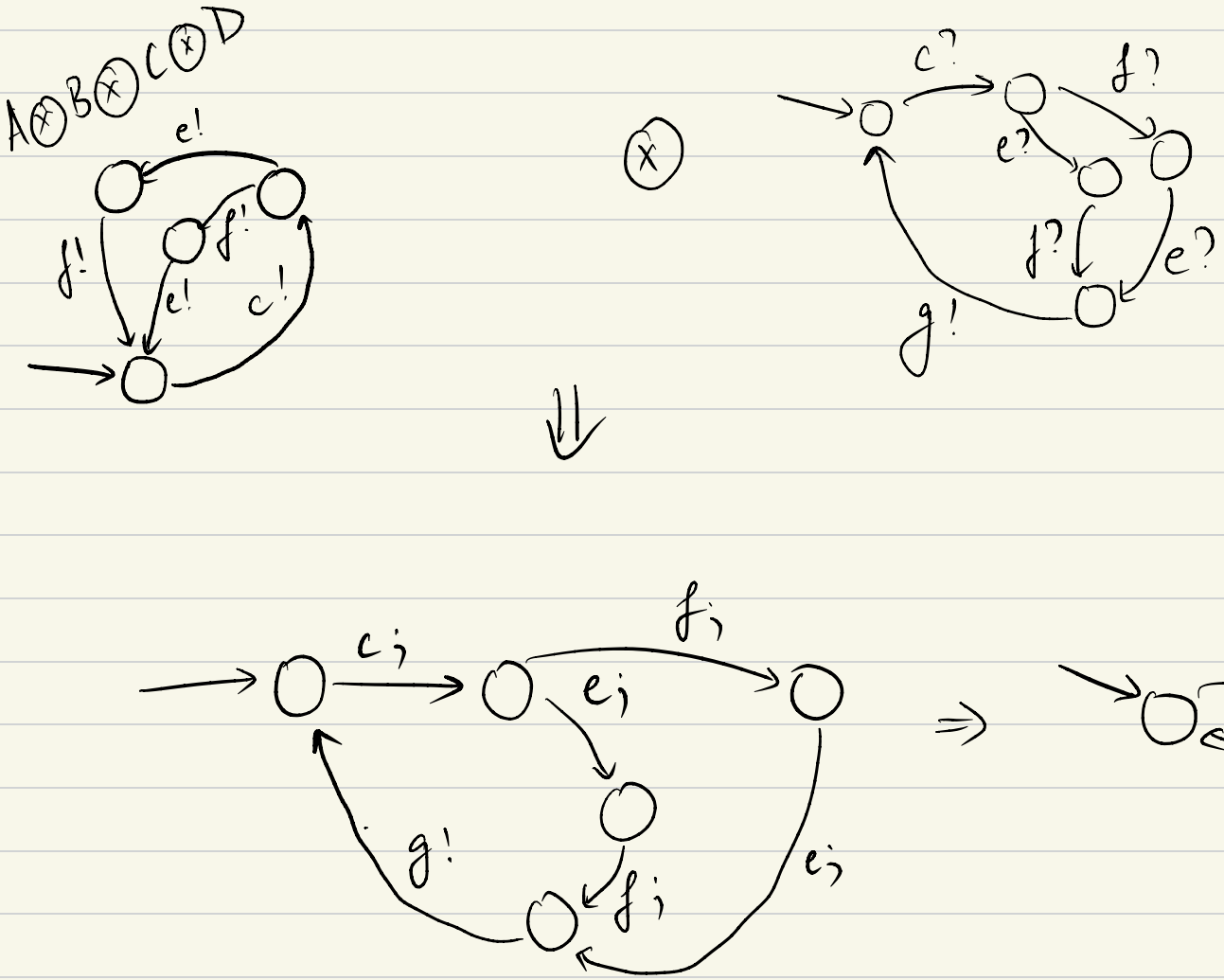

e!
d!
c!
e!
c!
e!
d!
d!
c!
a;
b!


d!
c!
e!
e!
c!
d!
c!
b!

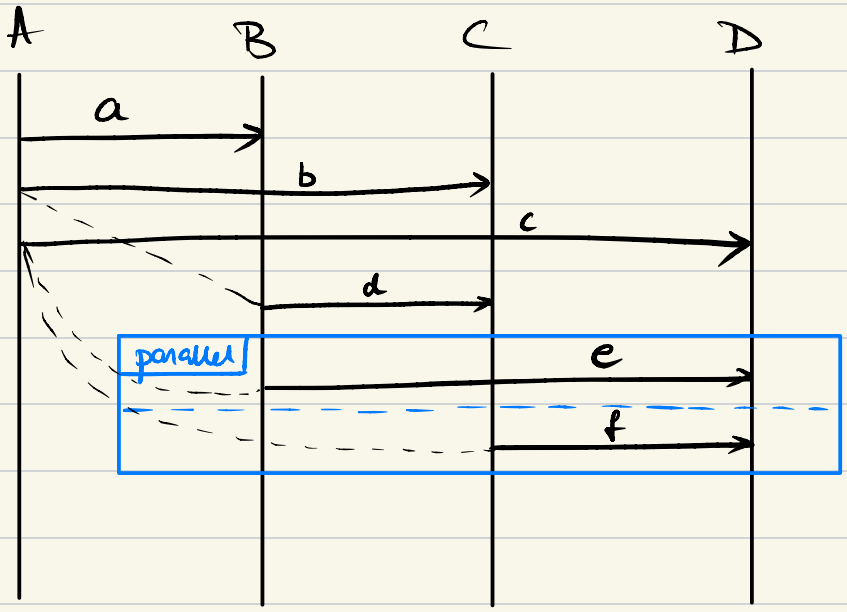

b?
f!
d?


e!
d;
f!
f!
e!
c!
b;
d;
f!
c!
c!
e!
e!
f!
f!
c!
e!
e!
e!
e!
c!
f!
f!
c!

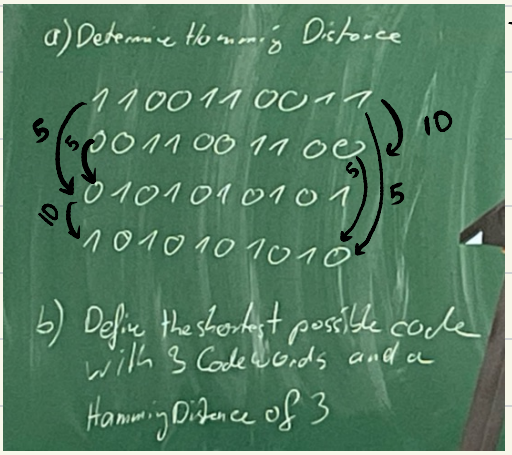
A⊗B⊗C



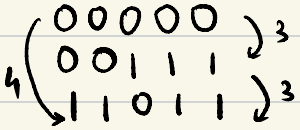
(b) transfer to SD



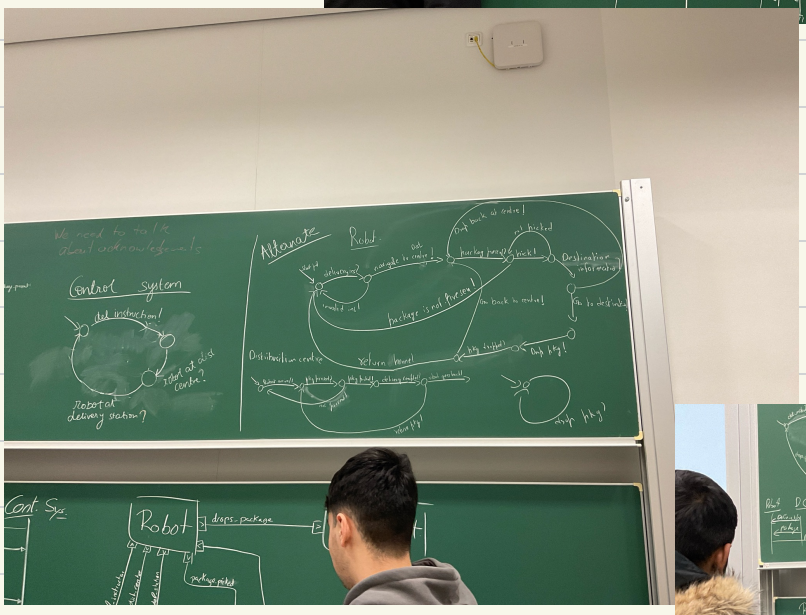
Q

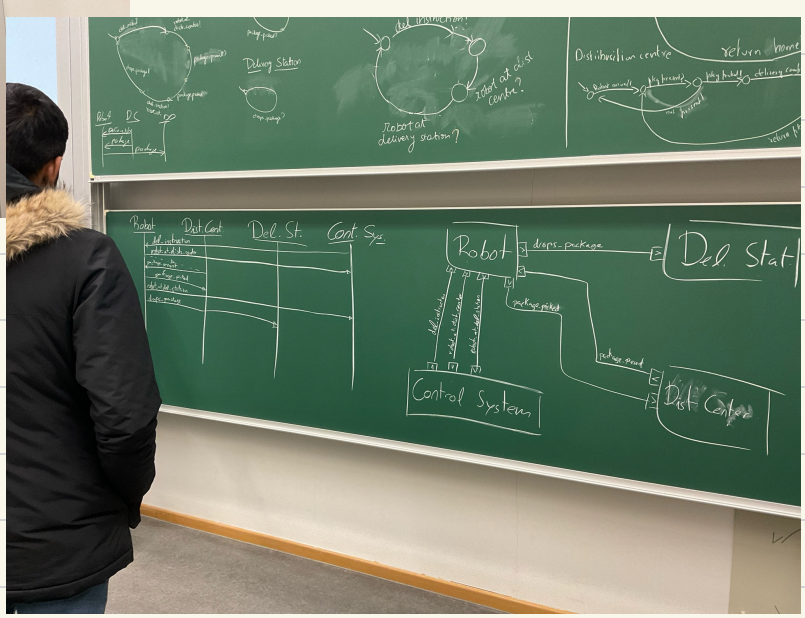


→ hamming dist = 5




Robot
Control System
Alternate
Robot
Distribution centre
Delivery Station
Robot
Dist. Cent
Del. St.
Cont. Sys
Robot
drops_package
Del Stat


Control System
Alternate
Robot
Distribution centre
Cont. Sys
Robot
drops_package


Delivery Station
Distribution centre
Robot
Dist. Cent
Del. St.
Cont. Sys
Robot
drops_package
Del Stat
Control System
Dist. Centre

Sample Exam Q.