Final Project Announcement

Project Logistics

- 3 students per team (1,2,3)
- 3 projects (a,b,c)
- 3 jobs
  - Requirement (R)
  - Development (D)
  - Validation (V)
- Student 1: a.R+b.D+c.V
- Student 2: b.R+c.D+a.V
- Student 3: c.R+a.D+b.V

Projects

1. Elevator
2. Banking System
3. Painkiller Injection System

Elevator

- A building with 3 floors and a basement (-1 1 2 3 floor)
- 2 elevators (should be coordinated)
- Interfaces
  - Button panels and display inside each elevator
  - Button panels and display on each floor
- System Events
  - Door open, door closed
  - Elevator arrive at each floor

Banking system

- A database containing all account data
  - Checking & saving accounts
- An interface for APP
  - transfer money to other people
  - open/close account
- An interface for ATM
  - deposit & withdraw cash
  - query account details

Painkiller Injection System

- Patients need painkiller after surgery
- There are limits on
  - The total amount per day 3ml
  - Amount in a short period 1ml/hr
- Baseline
  - 0.01-0.1ml/min
- Bolus
  - 0.2-0.5ml/shot
- Interface
  - For physician
  - Patient button

Requirement

- Requirement document
- UML diagrams
  - Collaborate with development guy/gal
- Model of system environment for validation
  - Collaborate with validation guy/gal
- Traceability report
  - Collaborate with both development and validation
  - Focus on requirement
- User Manual

Development

- Detailed UML diagrams reflecting actual design
  - Collaborate with requirement guy/gal
- Implementation of the design
- Traceability report
  - Collaborate with both requirement and validation
  - Focus on specification, model translation and code

Validation

- Validation planning and execution
- Risk Management
- Testing
- Model checking
- Traceability
  - Collaborate with other two guys/gals
  - Focus on test case, models in model checking

Job Allocation

- Finish job allocation at the following link by Sun Mar. 17th

Name1(1R2D3V)	**Name 2(2R3D1V)**	**Name 3(3R1D2V)**

- 2-student teams should use the following job allocation (must choose Painkiller)

Name1(1R3D1V)	**Name 2(3R1D3V)**

- Changes in team composition and job allocation must be formally announced to the instructor team via email

Group meetings

- Weekly meetings (Required)
- Meeting report for each project
  - The requirement guy/gal for each project is the organizer of the meeting.
  - For each group member, summarize works done in the previous week
  - Summary of the meeting
  - For each group member, propose action items for the next week
  - Due every Fri at the end of day, starting next week
- This is an important traceability document
- Please be specific
  - “continue working on project” does not work

Office Hours

- Zhihao Jiang (jiangzhh@)
  - Mon 7:30-8pm SIST 3-424
- TA
  - Thu 7pm-8pm SIST 1B-105
  - 何沛霖(hepl2023@)
  - 王文滔(wangwt2023@)

Why do we need models?

- Prediction
  - We know the low-level mechanisms but we want to understand how they affect higher-level behaviors
  - Use simulation instead of testing on the real system

- Explain the data
  - Make assumptions and use our knowledge to explain mechanisms that we don’t understand

- Classification
  - i.e. definitions, machine learning algorithms

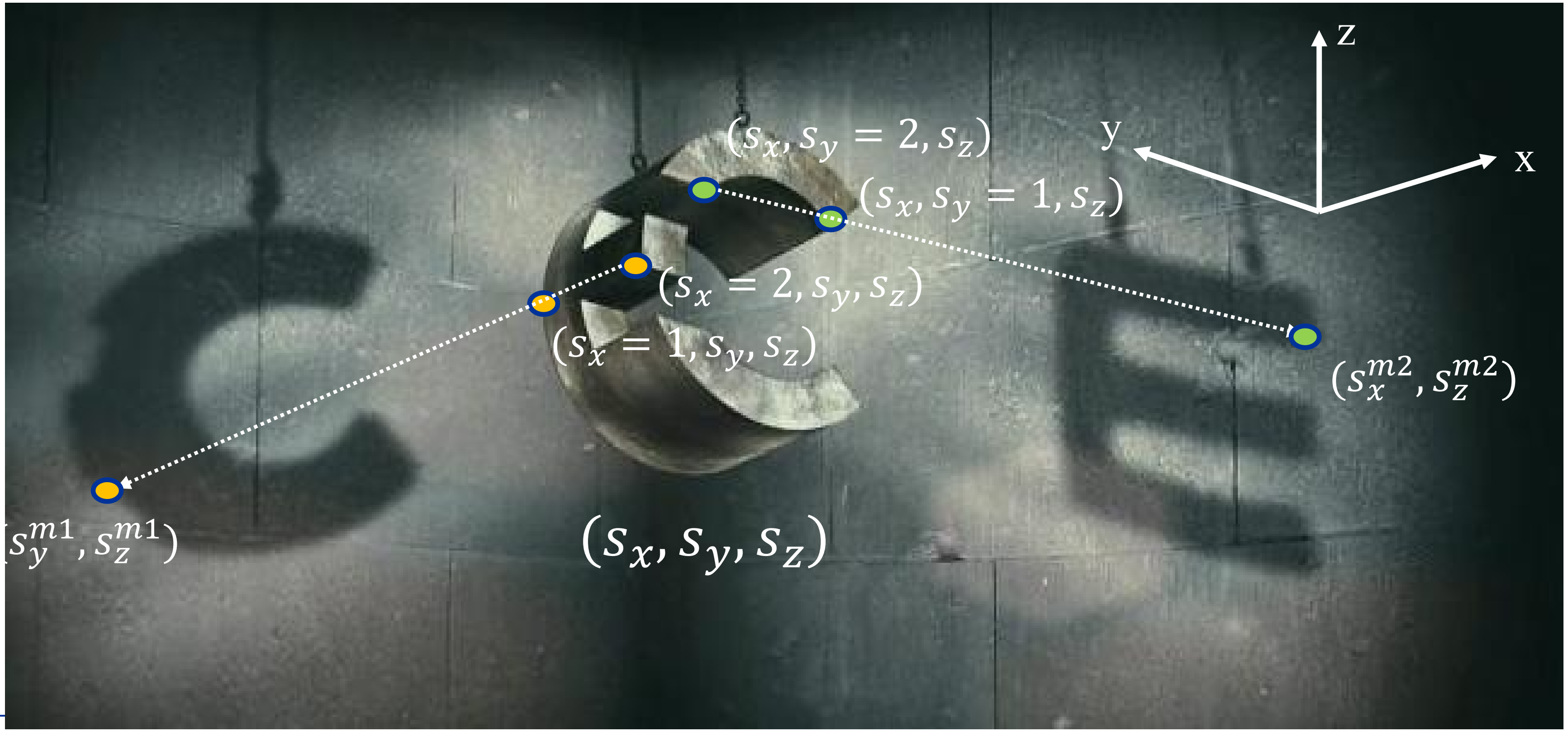
What are models?

- A system: $(S,\ I, T,\ O)$
  - S: States $s_1, s_2 \ldots s_n$
  - I: Inputs (could be $\emptyset$)
  - T: Transitions $S \times I \times S$
  - O: Observations $f(S_o)$, $S_o \subseteq S$

- Model of the system $(S^m,\ I^m, T^m)$
  - S^m: Abstraction/interpolation of S
    - Much fewer state variables
  - I^{m}: abstraction of I (could be $\emptyset$)
  - T^m: Transitions $S^m \times I^m \times S^m$

Abstraction – removal of state variables

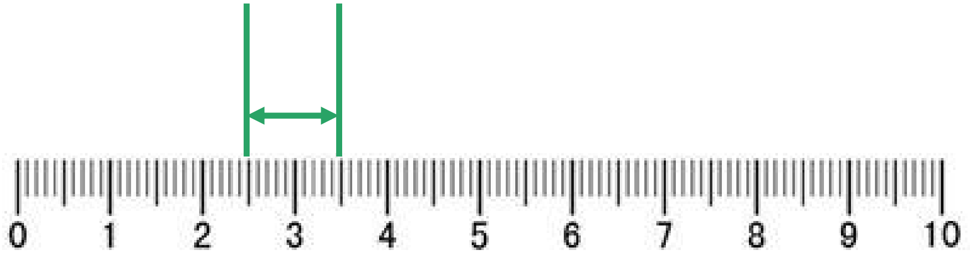
- States (s_x, s_y, s_z) *are abstracted to* (s_y^{m1}, s_z^{m1})
  - $(s_x, s_y, s_z) \rightarrow (s_y^{m1}, s_z^{m1})$
- Loss of information



Abstraction: Approximation of state variable values

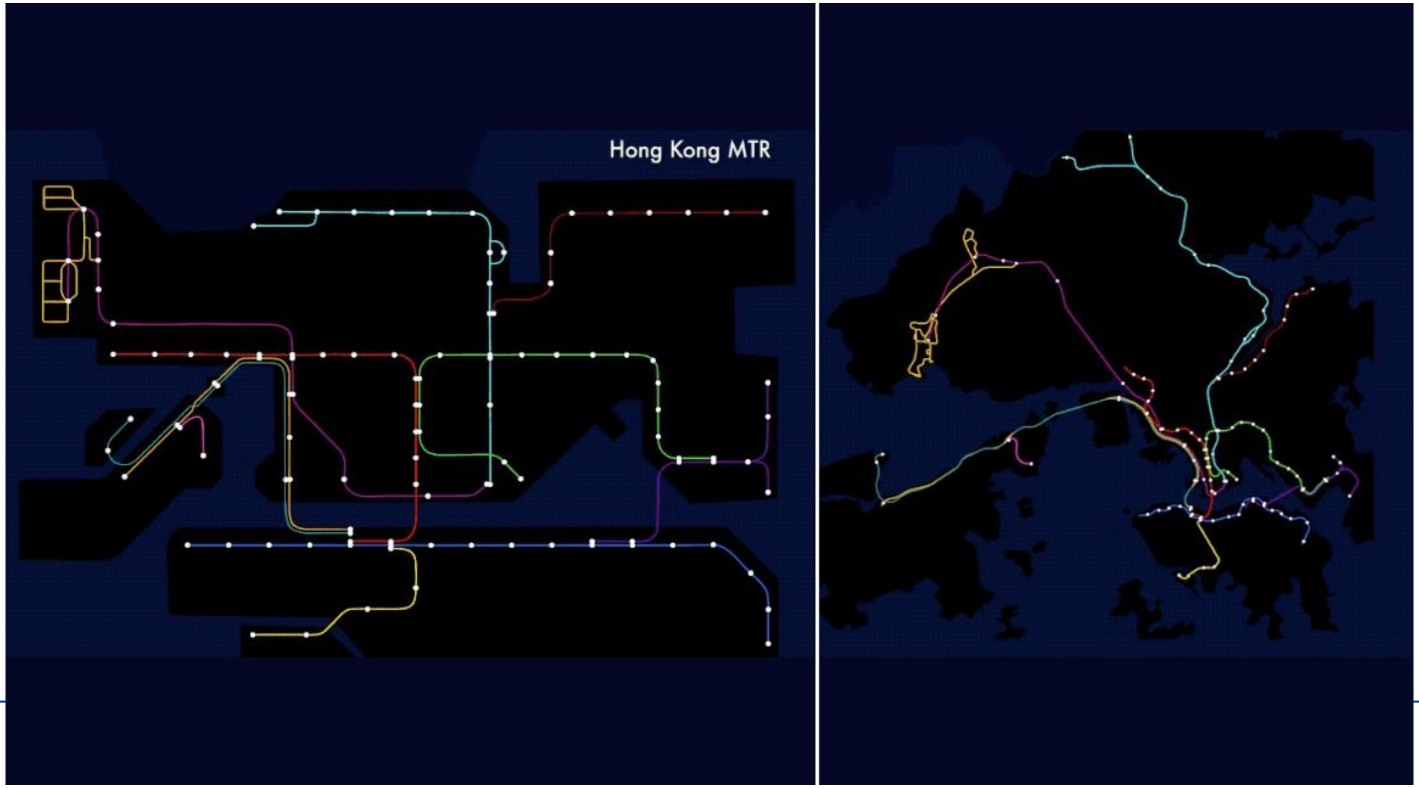
- Irrational numbers
  - $\pi \approx 3.1415$
  - $\sqrt{2} \approx 1.414$

- Approximation is another way of abstraction

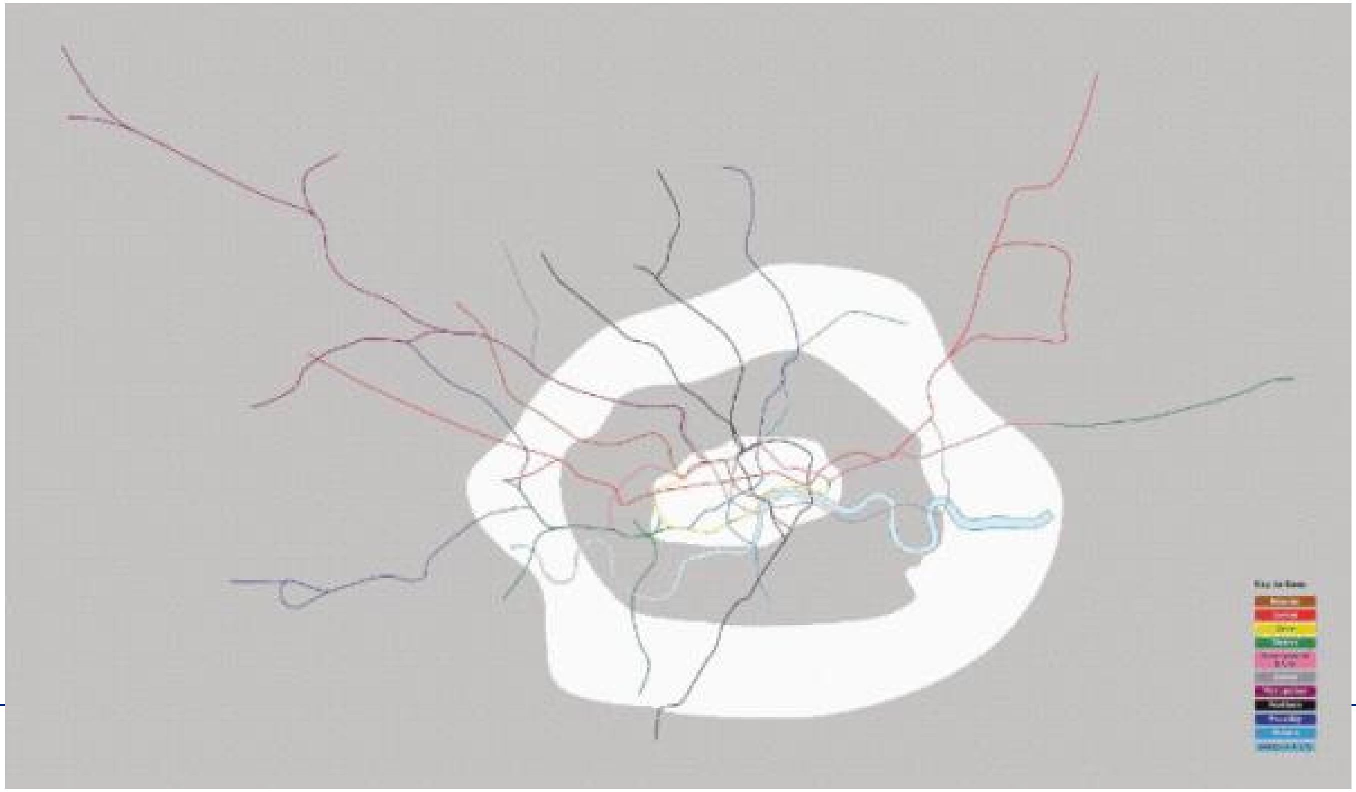


Interpolation: extracting interpretable information

- Locational information -> topological information
- $S^m = f(S_p),\ S_p \subseteq S$



More Interpolation: London MTR



What is considered as a “good” model?

- Accuracy
  - All models are wrong!
  - Error accumulates over time
  - Initial condition of the model cannot be determined due to limited observability
- Generality
  - The capability to explain not only training data, but also testing data
- Identifiability
  - Model parameters can be identified from data
- Interpretability
  - S^m are meaningful and interpretable by human

Newton vs. Einstein

- Newtonian physics is suitable for macro level objects at low speed
- $L = L_0\sqrt{1 - \frac{v^2}{c^2}}$
- A model can only be “good” within the context of its designated application
- The definition of “goodness” is changing over time

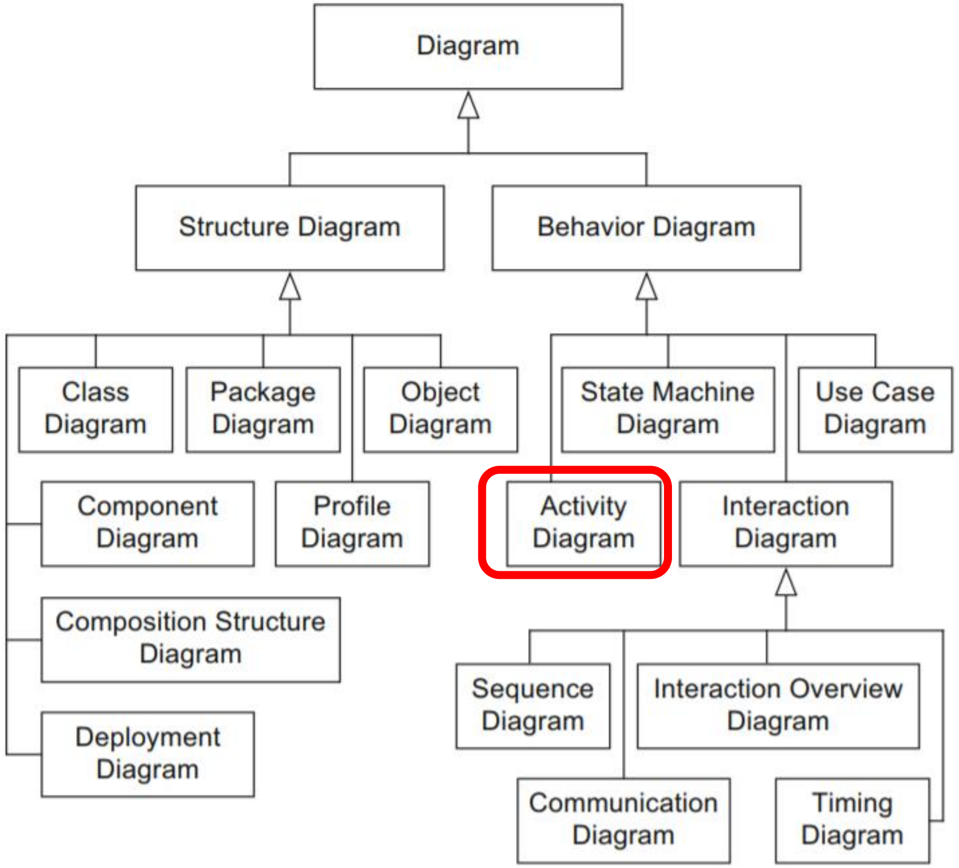
Modeling methodologies

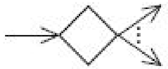
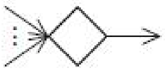
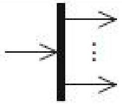
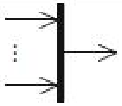
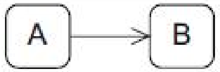
- Bottom-up modeling
  - "White-box" model
  - Using first principles
  - Pros:
    - Interpretable
    - Convincing
  - Cons:
    - State space explosion
    - Difficult to be general
    - Low identifiability

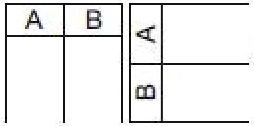
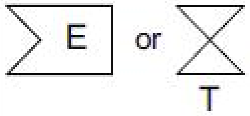
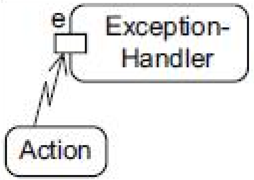
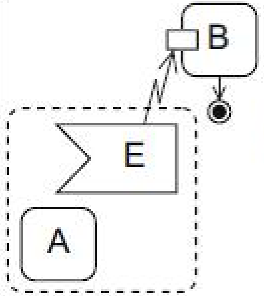
- Data driven models (i.e. Neural networks)
  - "Black-box" model
  - From observable data
  - Pros:
    - No need to know domain knowledge
  - Cons:
    - Large and uninterpretable S^m
    - Depends highly on the quality and quantity of data

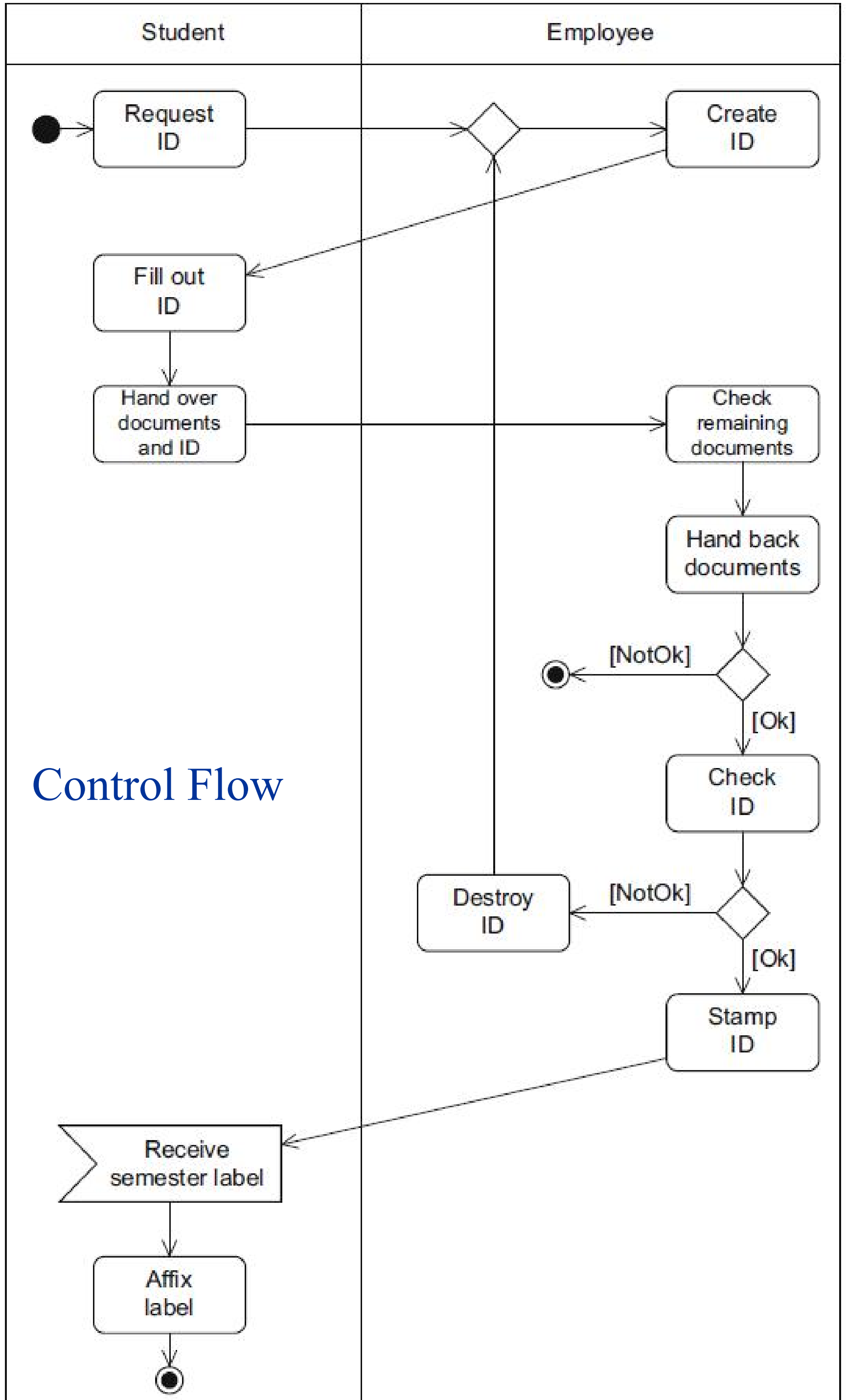
Lecture 6: UML Part 3

UML Diagrams

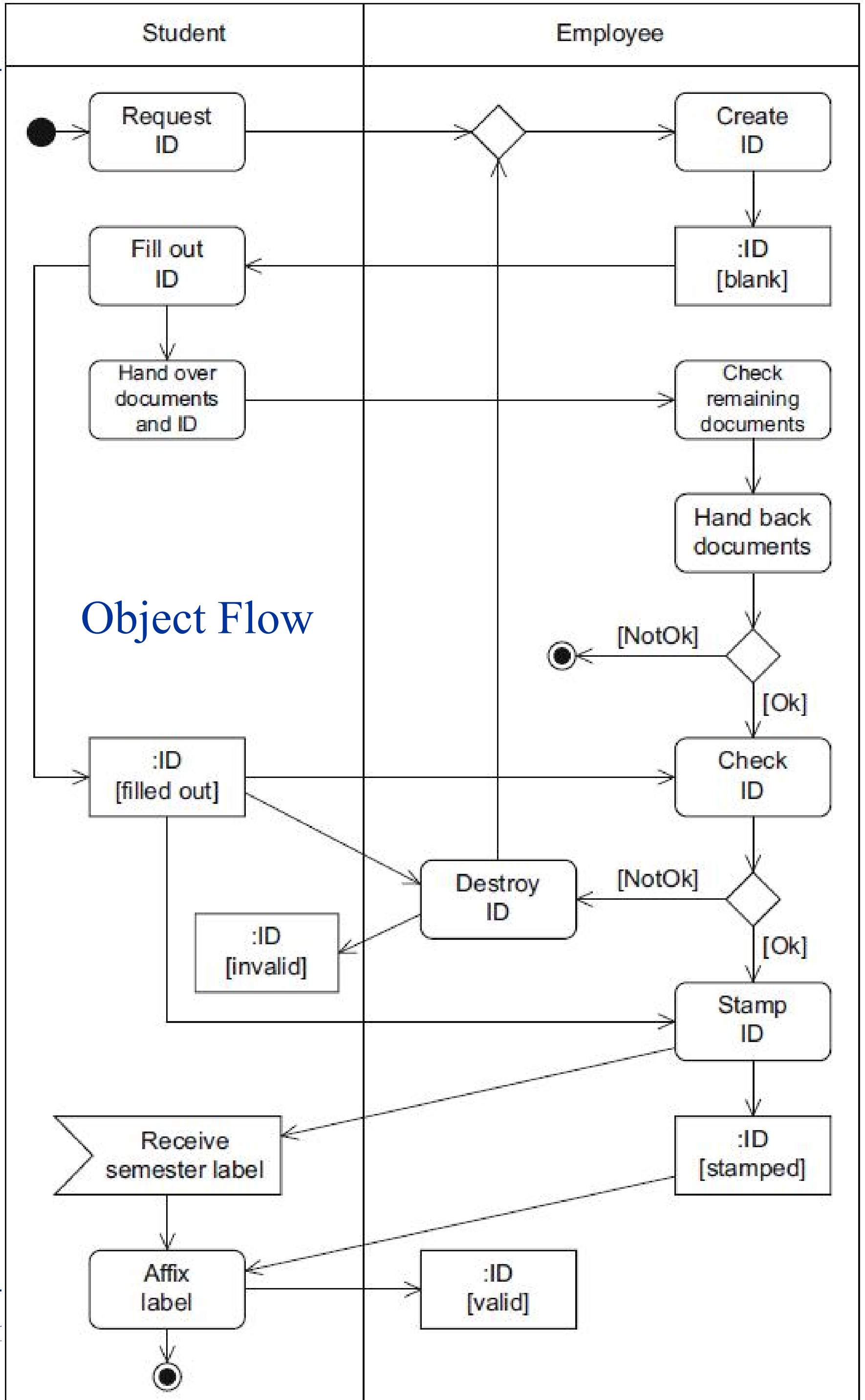


Name	*Notation*	*Description*
Action node		Actions are atomic, i.e., they cannot be broken down further
Activity node		Activities can be broken down further
Initial node		Start of the execution of an activity
Activity final node		End of ALL execution paths of an activity
Flow final node		End of ONE execution path of an activity
Decision node		Splitting of one execution path into two or more alternative execution paths
Merge node		Merging of two or more alternative execution paths into one execution path
Parallelization node		Splitting of one execution path into two or more concurrent execution paths
Synchronization node		Merging of two or more concurrent execution paths into one execution path
Edge		Connection between the nodes of an activity
Call behavior action		Action A refers to an activity of the same name
Object node		Contains data and objects that are created, changed, and read
Parameters for activities		Contain data and objects as input and output parameters
Parameters for actions (pins)		Contain data and objects as input and output parameters

Name	*Notation*	*Description*
Partition		Grouping of nodes and edges within an activity
Send signal action		Transmission of a signal to a receiver
Asynchronous accept (time) event action		Wait for an event E or a time event T
Exception handler		Exception handler is executed instead of the action in the event of an error e
Interruptible activity region		Flow continues on a different path if event E is detected

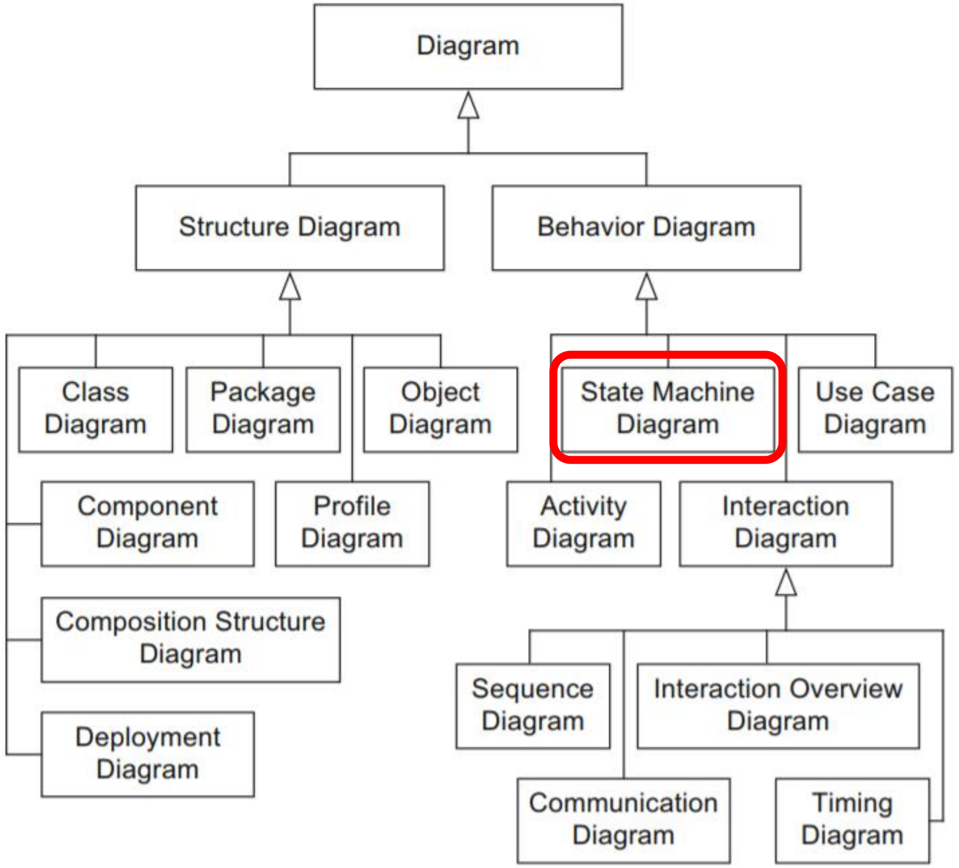


Control Flow



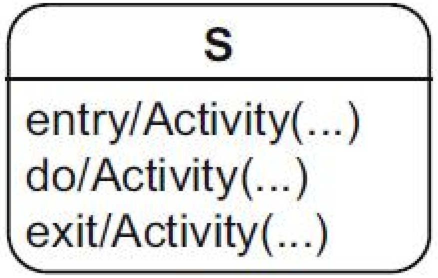
Object Flow

UML Diagrams

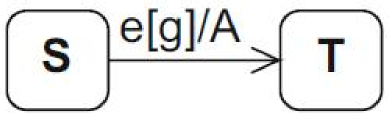


State Machine

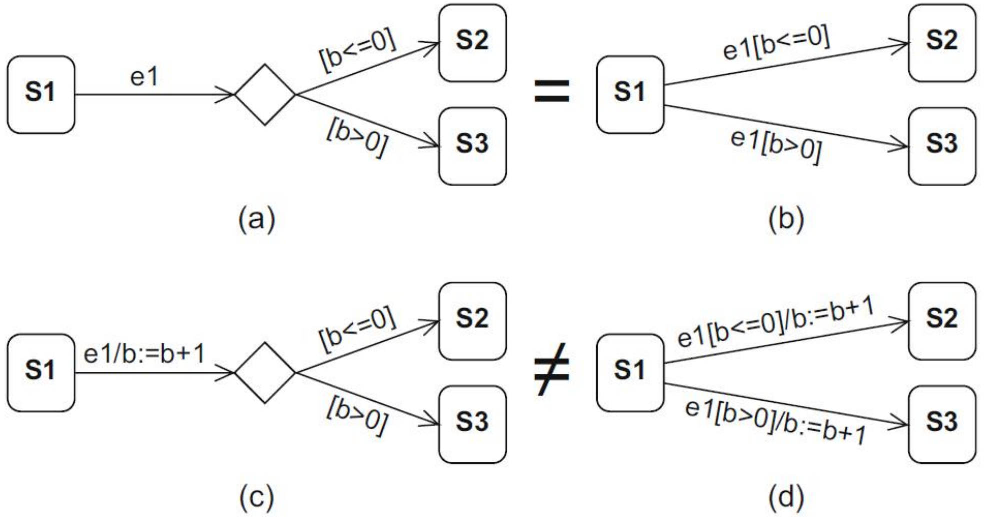
- State
  - entry: upon entering the state
  - do: during the state
  - exit: upon exiting the state
- Transition
  - e: event/trigger for the transition
  - g: guard/condition for the transition
  - A: activity when executing the transition

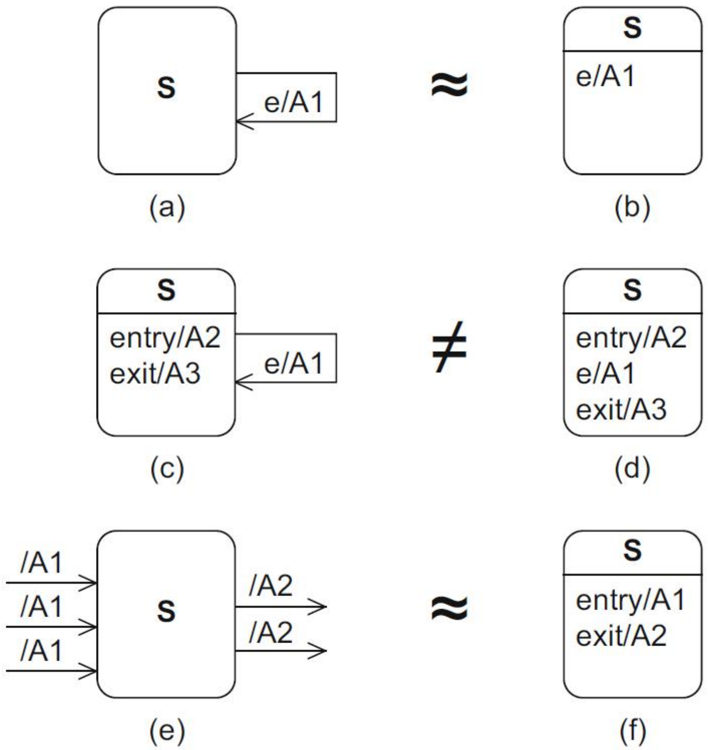


Transition



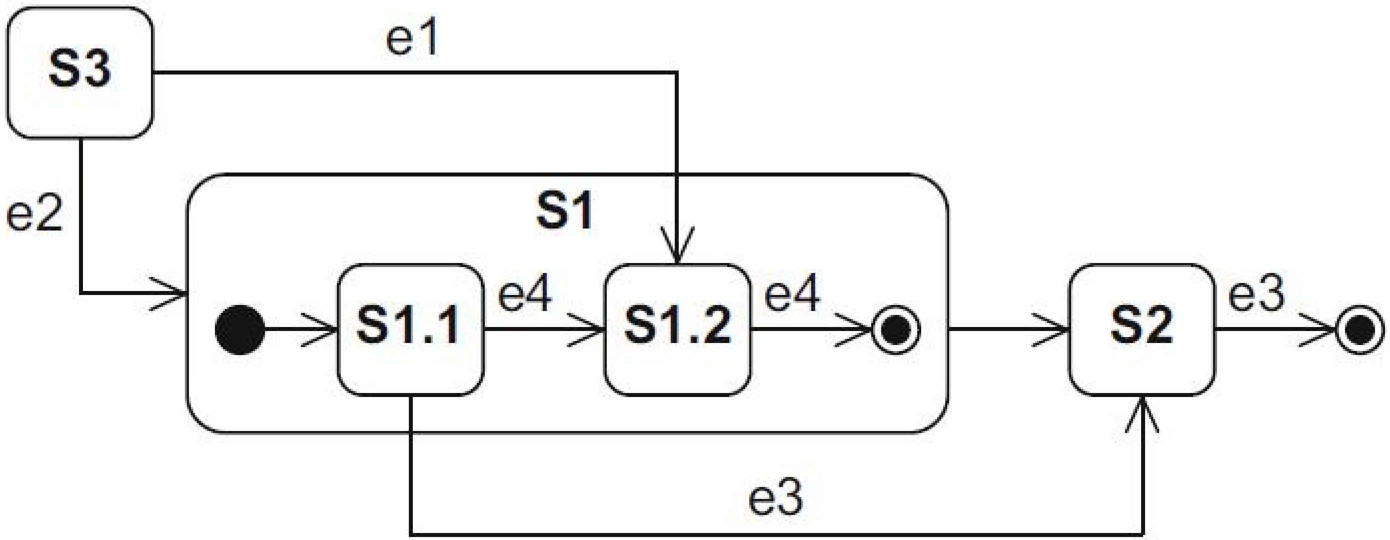
State Machine Diagram: Semantics



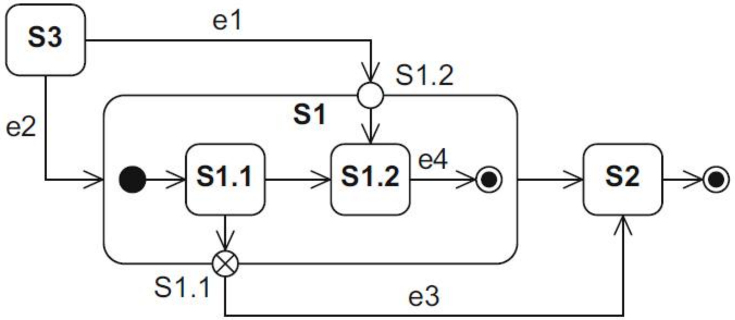


Composite States

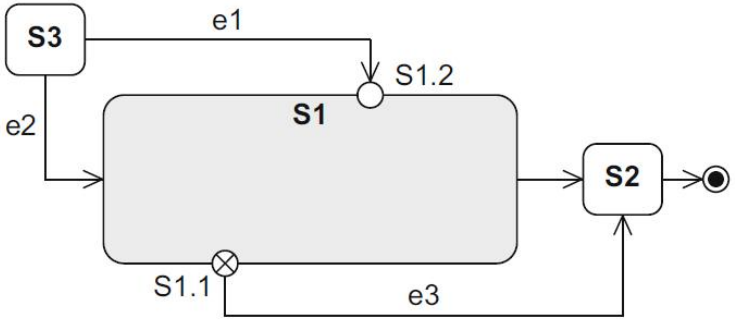
- Nested States



Entry and Exit Points



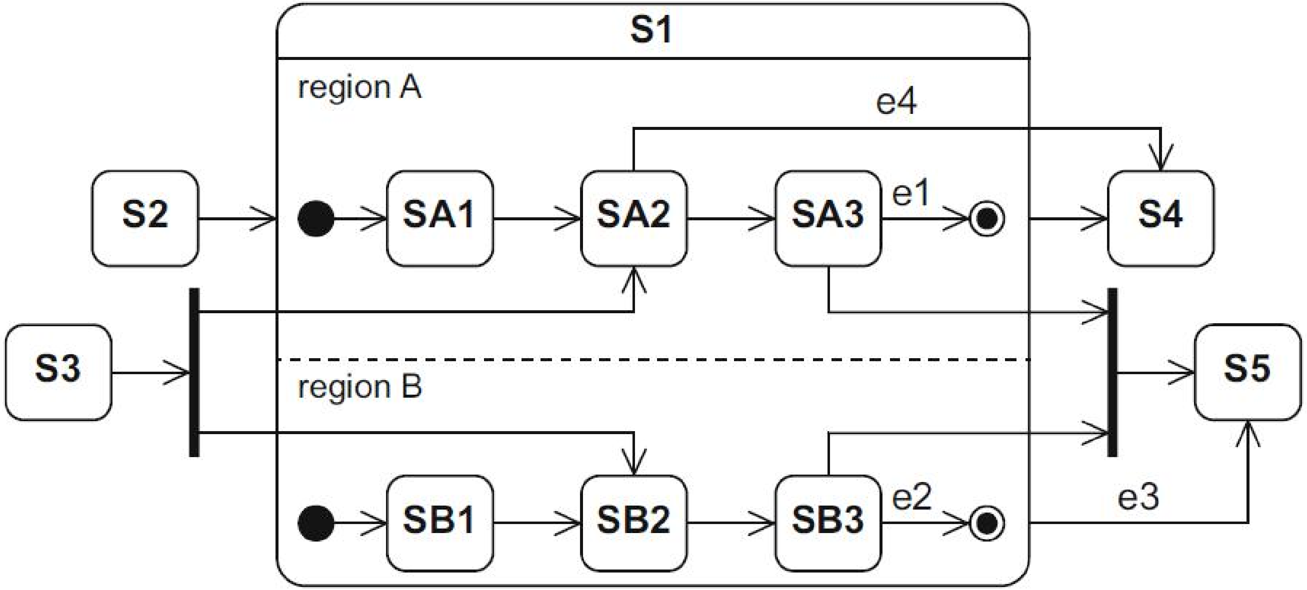
(a)



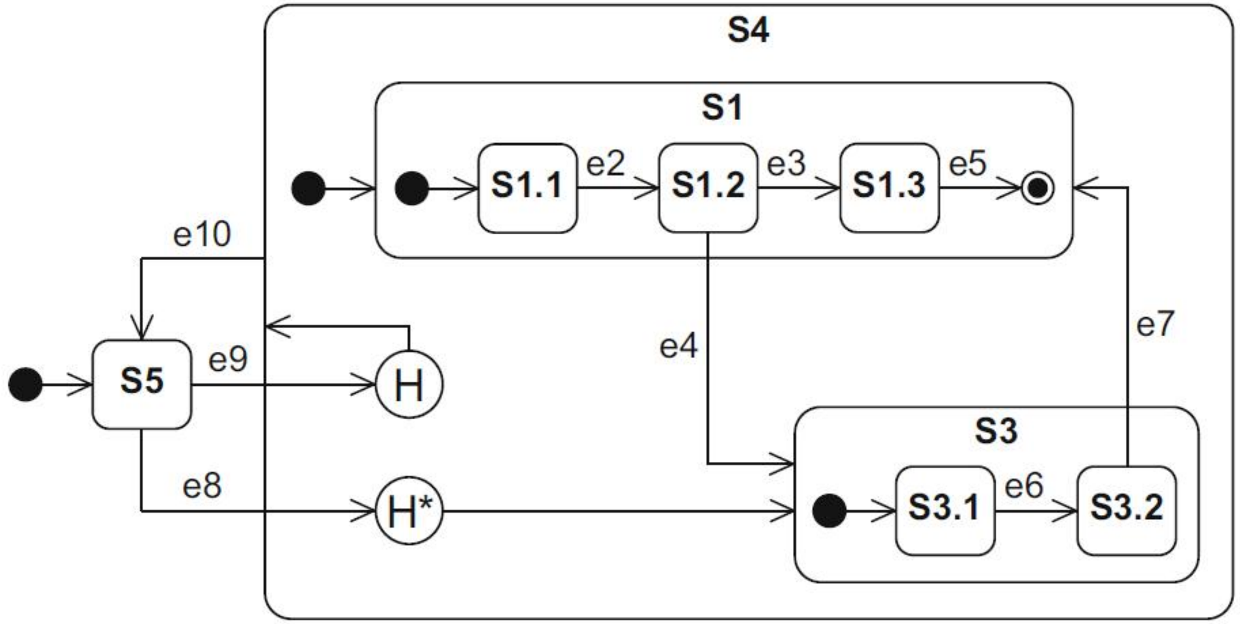
(b)

Orthogonal State

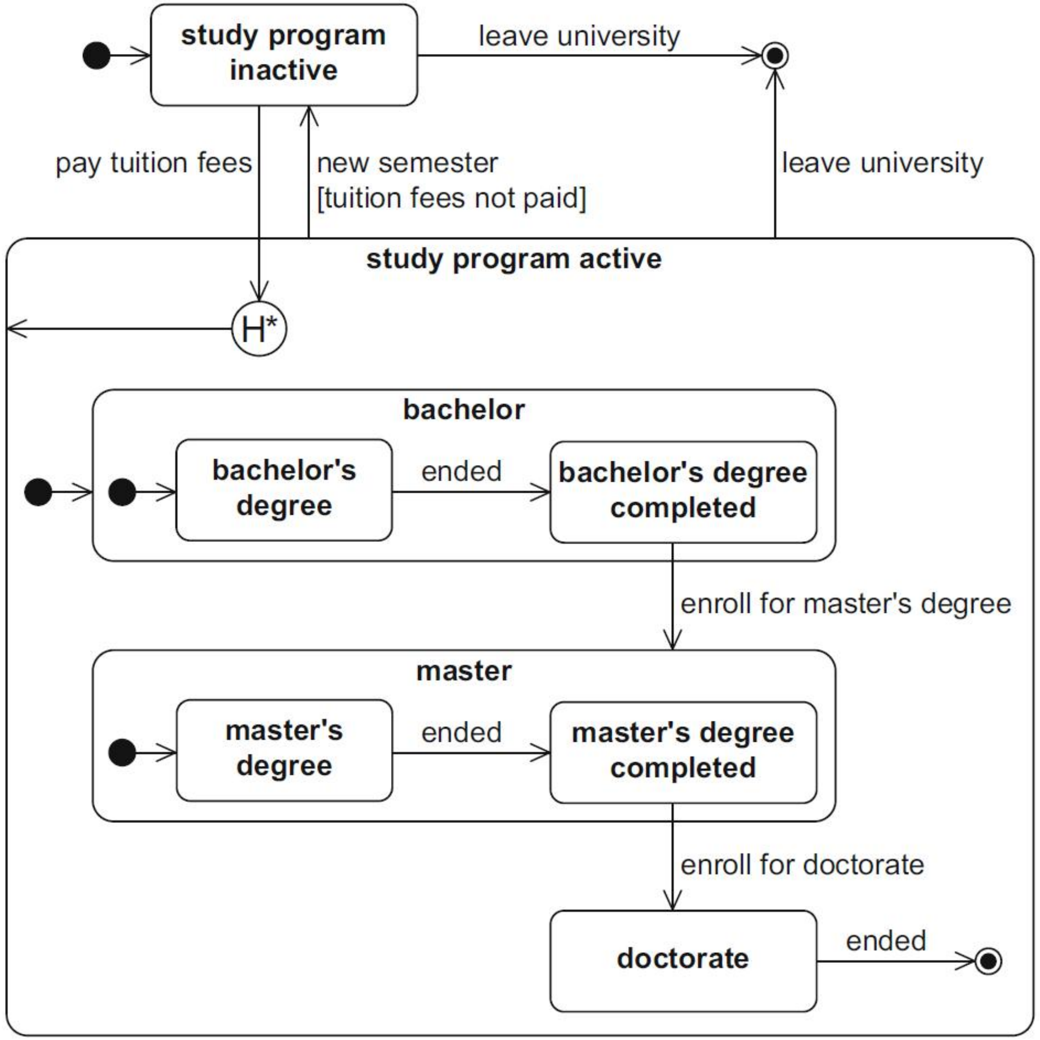
- Parallel State



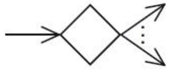
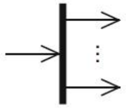
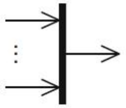
History State



Example


study program inactive
leave university
pay tuition fees
new semester [tuition fees not paid]
leave university
study program active
H*
bachelor
bachelor's degree
ended
bachelor's degree completed
enroll for master's degree
master
master's degree
ended
master's degree completed
enroll for doctorate
doctorate
ended

Summary

Name	*Notation*	*Description*
State	S entry/Activity(...) do/Activity(...) exit/Activity(...)	Description of a specific "time span" in which an object finds itself during its "life cycle". Within a state, activities can be executed on the object.
Transition	S $\xrightarrow{e}$ T	State transition e from a source state S to a target state T
Initial state	●	Start of a state machine diagram
Final state	◉	End of a state machine diagram
Terminate node	×	Termination of an object's state machine diagram
Decision node		Node from which multiple alternative transitions can proceed
Parallelization node		Splitting of a transition into multiple parallel transitions
Synchronization node		Merging of multiple parallel transitions into one transition
Shallow and deep history state	(H) / (H*)	"Return address" to a substate or a nested substate of a composite state