Seminar on Concurrency Theory

Ori Lahav

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

March 22, 2023

Go to

www.menti.com

Enter the code

28 86 16 6

Or use QR code

Today

- What is this seminar about?
- Goals, requirements and logistics of the seminar
- List of student presentations

About me: Ori Lahav

Ph.D.
Logic in computer science
Advisor: A. Avron

Postdoctoral researcher
Program verification
Host: M. Sagiv

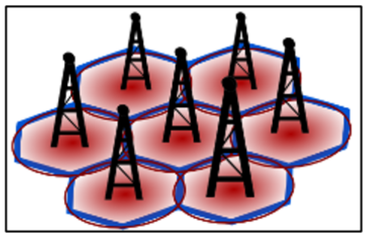
Postdoctoral researcher
Weak memory models
Hosts: V. Vafeiadis, D. Dreyer

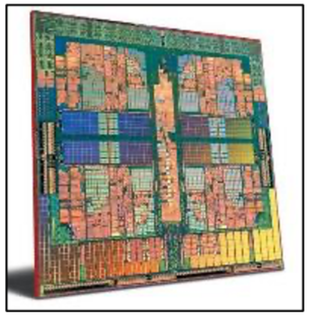
Since 2017 - Faculty member
Tel Aviv University

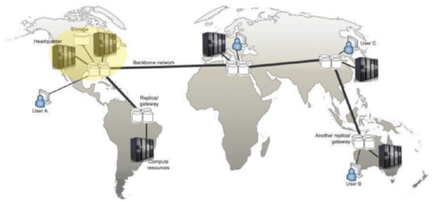
Main areas of research:
- Programming languages theory
- Verification
- Concurrency
- Relaxed memory models

Concurrency theory

- Rigorous mathematical formalisms and techniques for **modeling** and **analyzing** concurrent systems.
- Concurrent systems include concurrent programs & reactive systems.
- Particular focus on communication and synchronization (rather than simple parallelism).

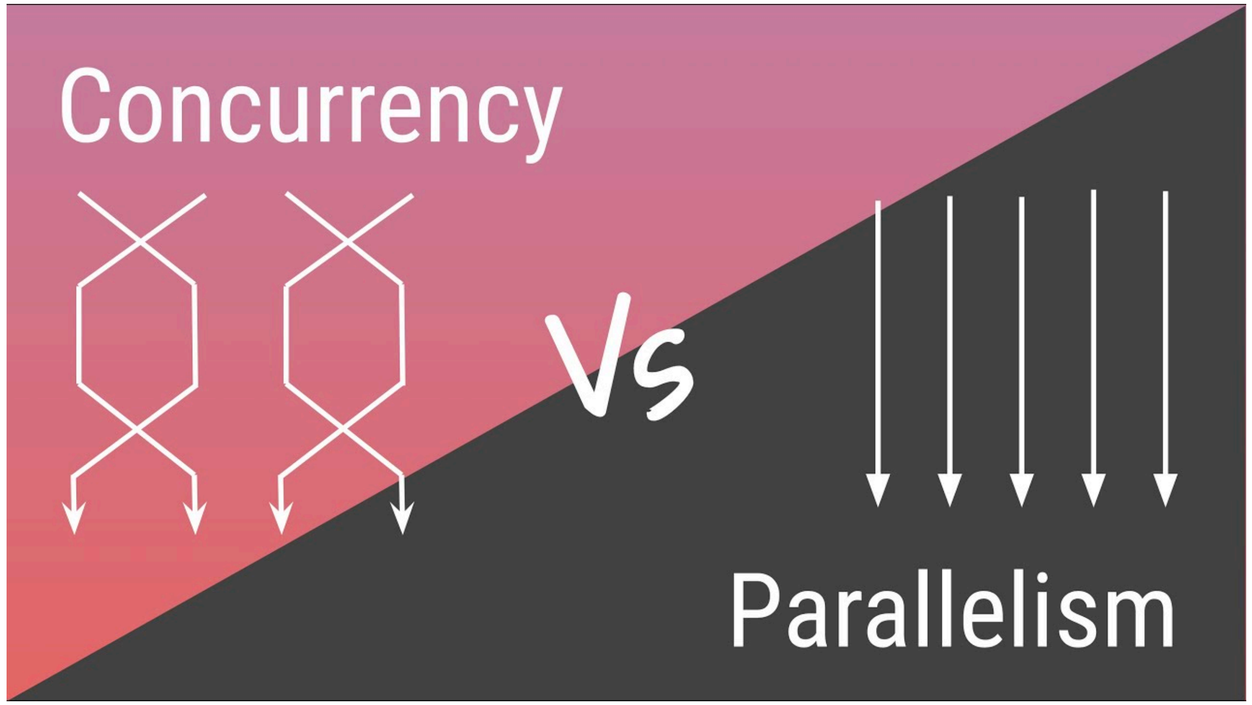






Concurrency vs. Parallelism

- סמינר בתיאוריה של בו-זמניות (?)



Parallelism

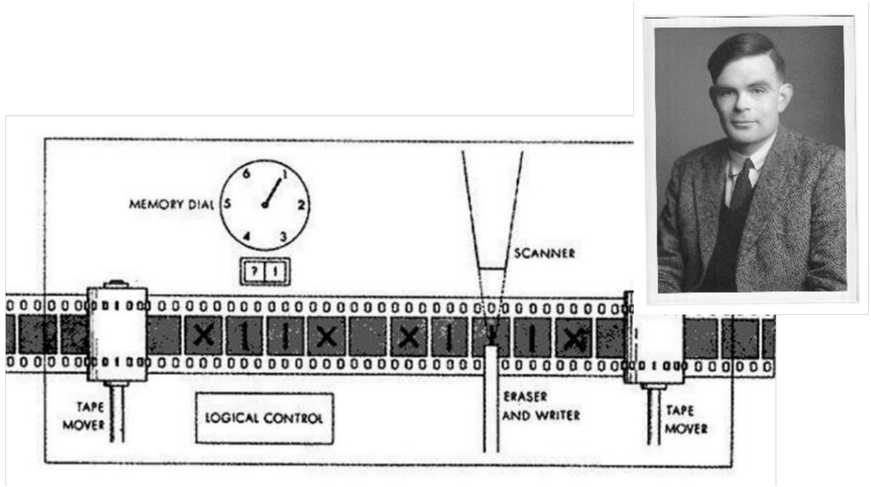
Concurrency

Parallelism is about doing lot of thing at once

Concurrency is about dealing with lot of thing at once - Rob Pike

Reactive systems

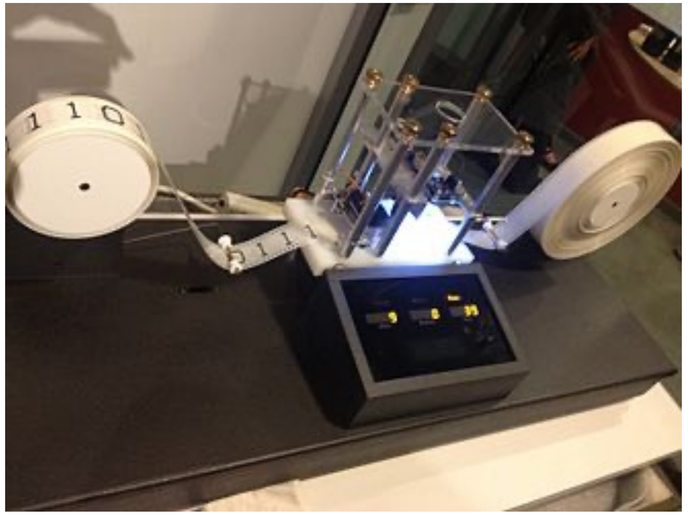
The classical view



- A program transforms an input into an output.
- Denotational semantics:
  the meaning of a program is a partial function:

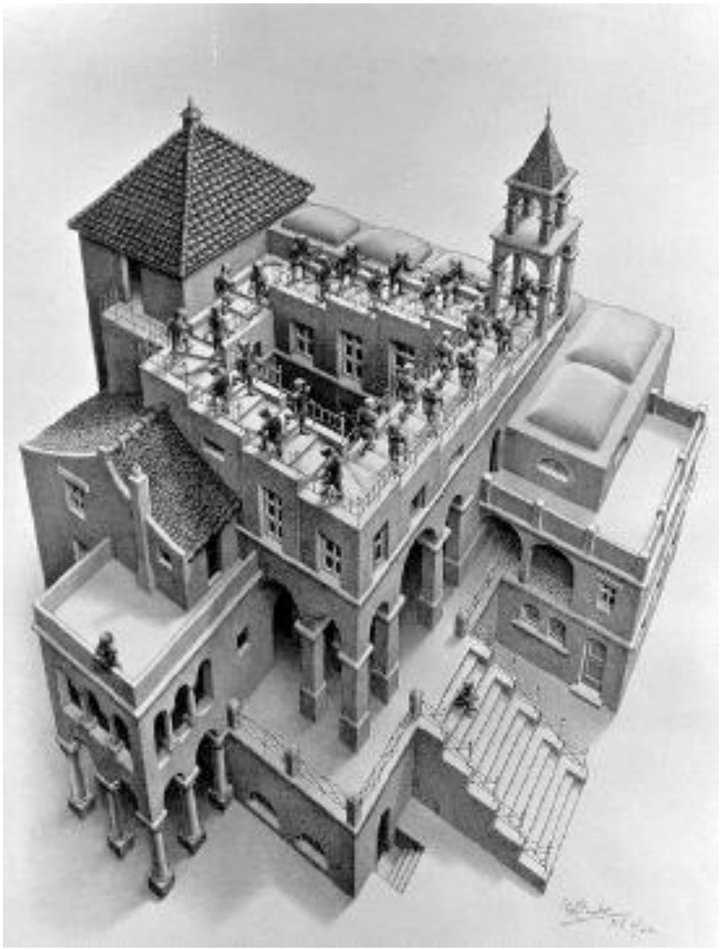
$$States \rightarrow States$$

- Non-termination is bad.
- *Is that what we need?*



What about:

- operating systems?
- websites?
- database systems?
- power plants?
- vending machines? …

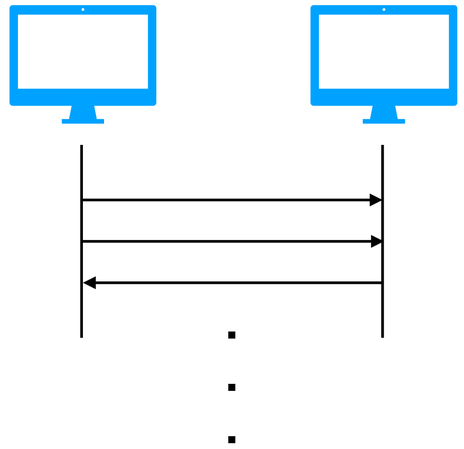


Ascending and Descending by M. C. Escher

Reactive systems

Reactive systems continuously reacts to the environment and influence the environment

- Key issue: communication and interaction.
- Non-determinism is often inevitable.
- What is correctness?
  - Often halting is actually a problem.
  - Not crashing (e.g., “dividing by 0”).
  - Serving requests on time.
  - Adhering to certain communication protocols.
- What is equivalence? refinement?



Example: Equivalence

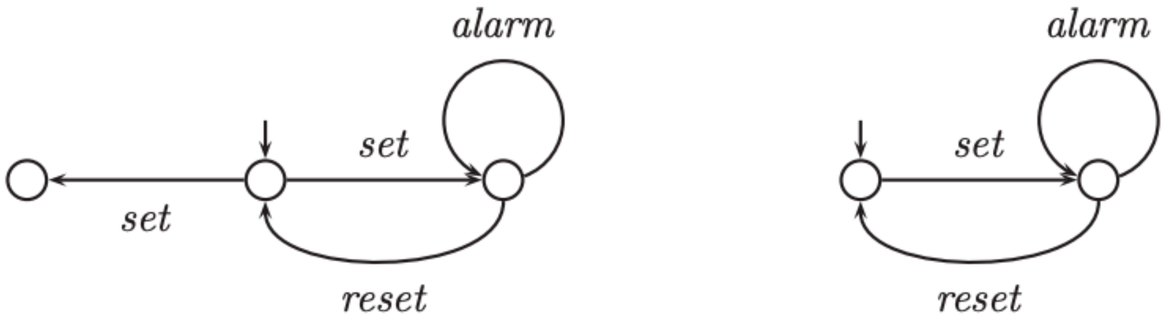


Figure 2.5: Two trace-equivalent alarm clocks

Example taken from Chapter 2 of:
Modeling and Analysis of Communicating Systems
By Jan Friso Groote, Mohammad Reza Mousavi

Example: Session Types

- In plain English, we can describe a simple protocol for ATM:
  - The client communicates his/her ID to the ATM
  - The ATM then answers either ok or err
    - In the first case, the client then proceeds to request either a deposit or withdraw
      - For a deposit the client first sends an amount, then the ATM responds with the updated balance
      - For a withdraw the client sends the amount to withdraw, and the ATM responds with either ok or err to indicate whether or not the transaction was successful
  - If the ATM answers err, then the session terminates.

Sources:

http://munksgaard.me/papers/laumann-munksgaard-larsen.pdf

https://stanford-cs242.github.io/f18/lectures/07-2-session-types.html

Example: Session Types

$$\begin{aligned}\mathsf{ATM} &= \mathsf{recv\ id};\ \mathsf{choose}\ \{\mathsf{ok}\colon (\mathsf{ATM}_{\mathsf{auth}}) \mid \mathsf{err}\colon (\varepsilon)\} \\ \mathsf{ATM}_{\mathsf{auth}} &= \mathsf{offer}\ \{\mathsf{deposit}\colon (\mathsf{recv\ u64};\ \mathsf{send\ u64};\ \varepsilon) \mid \mathsf{withdraw}\colon (\mathsf{recv\ u64};\ \mathsf{choose}\ \{\mathsf{ok}\colon (\varepsilon) \mid \mathsf{err}\colon (\varepsilon)\})\}\end{aligned}$$

Dual type

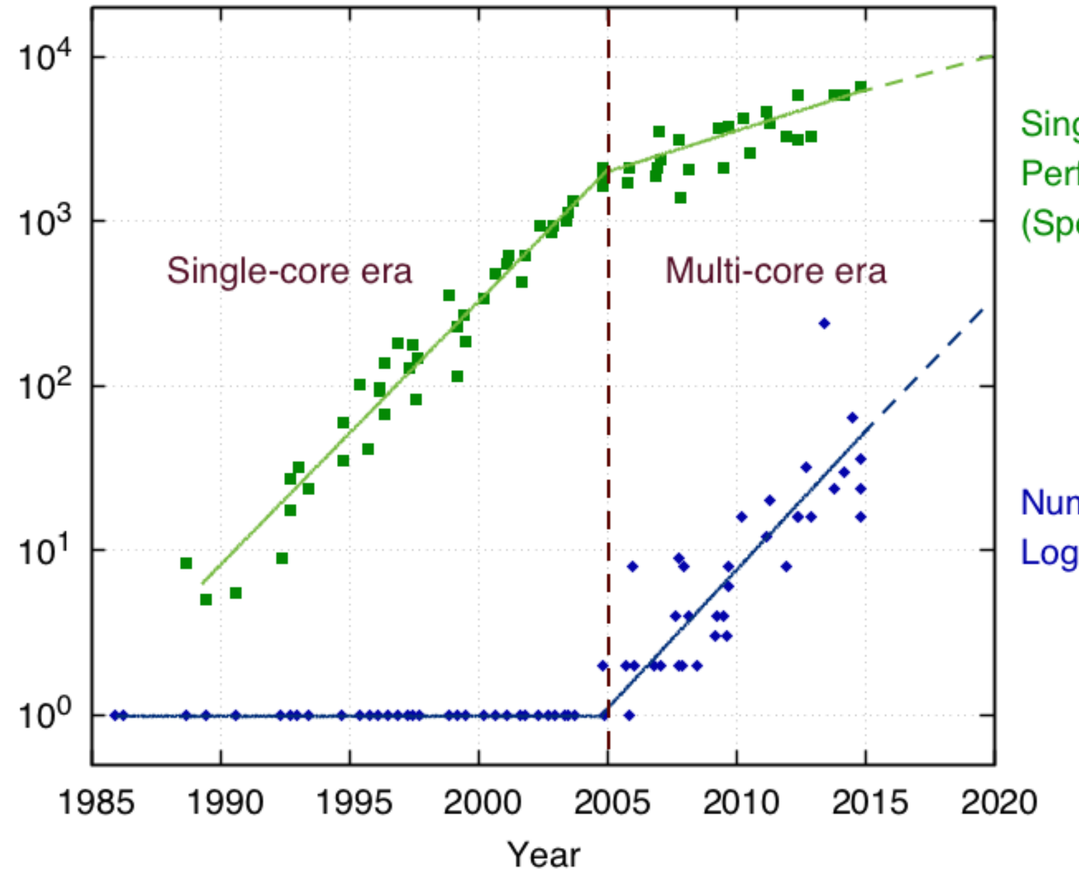
$$\begin{aligned}\mathsf{Client} &= \mathsf{send\ id};\ \mathsf{offer}\ \{\mathsf{ok}\colon (\mathsf{Client}_{\mathsf{auth}}) \mid \mathsf{err}\colon (\varepsilon)\} \\ \mathsf{Client}_{\mathsf{auth}} &= \mathsf{choose}\ \{\mathsf{deposit}\colon (\mathsf{send\ u64};\ \mathsf{recv\ u64};\ \varepsilon) \mid \mathsf{withdraw}\colon (\mathsf{send\ u64};\ \mathsf{offer}\ \{\mathsf{ok}\colon (\varepsilon) \mid \mathsf{err}\colon (\varepsilon)\})\}\end{aligned}$$

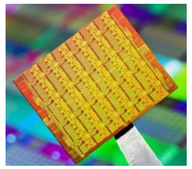
$$\begin{aligned}\overline{\mathsf{send}\ \tau;\ \sigma} &= \mathsf{recv}\ \tau;\ \overline{\sigma} \\ \overline{\mathsf{recv}\ \tau;\ \sigma} &= \mathsf{send}\ \tau;\ \overline{\sigma} \\ \overline{\mathsf{choose}\ \{L\colon (\sigma_L) \mid R\colon (\sigma_R)\}} &= \mathsf{offer}\ \{L\colon (\overline{\sigma_L}) \mid R\colon (\overline{\sigma_R})\} \\ \overline{\mathsf{offer}\ \{L\colon (\sigma_L) \mid R\colon (\sigma_R)\}} &= \mathsf{choose}\ \{L\colon (\overline{\sigma_L}) \mid R\colon (\overline{\sigma_R})\} \\ \overline{\varepsilon} &= \varepsilon\end{aligned}$$

Concurrent programming

Parallelism is here

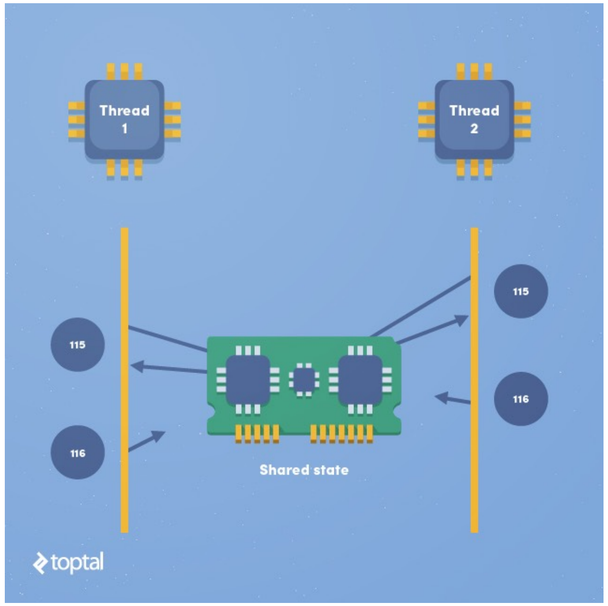
“The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”. By Herb Sutter (2005)





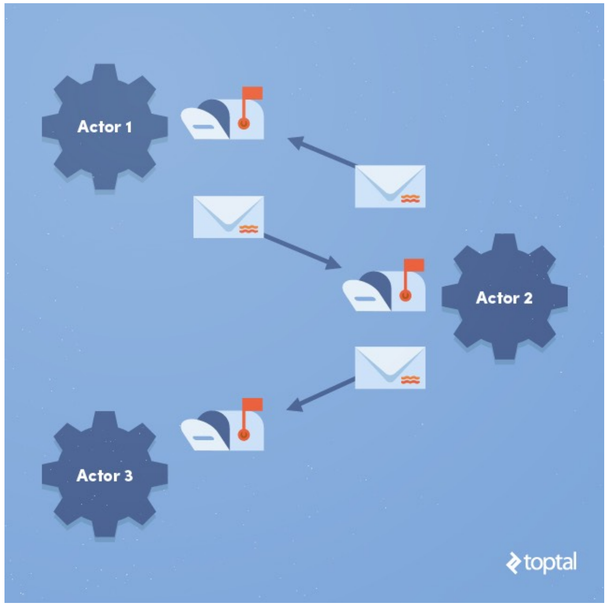
Two fundamental models of concurrent programming

shared memory



concurrent modules interact by reading and writing shared objects in memory

message passing

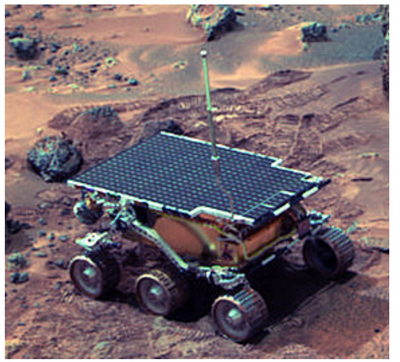


concurrent modules interact by sending messages to each other through a communication channel

PL examples: C / C++ Scala Erlang, Go

Hard to get right!

- Concurrency is widespread, but it is also **error prone**, and hard to debug and reproduce.
- **Non-determinism** is inherent.
- Unlike sequential programs, programmers need to take care of synchronization, race conditions, deadlocks, etc.
- Therac-25: Concurrent programming errors (in particular, race conditions) → accidents causing death and serious injury



- Mars Rover: Problems with interaction between concurrent tasks caused periodic software resets reducing availability for exploration

Simple example

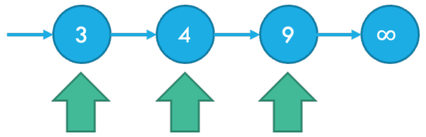
Initially X = 0.

X := X+1; X := X+3;

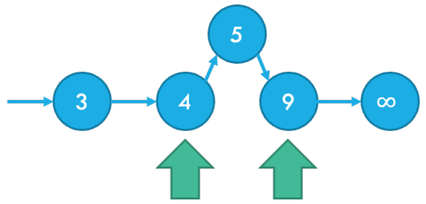
- How many possible outcomes?
- Such “bugs” may even disappear when you try to print it or even debug!

Concurrent Data Structures

Insert(5)

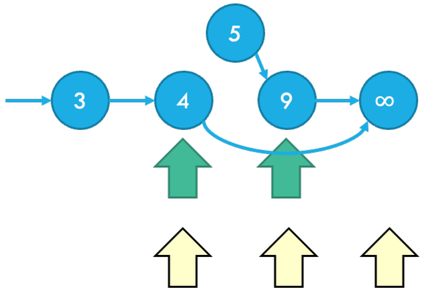


Insert(5)

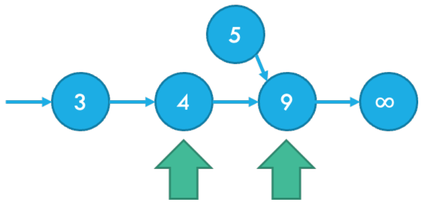


Insert(5)

Delete(9)

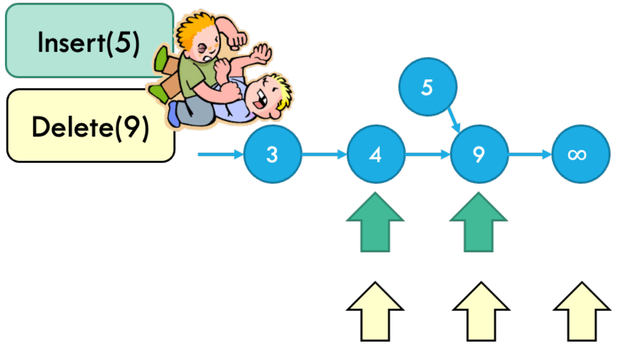


Insert(5)



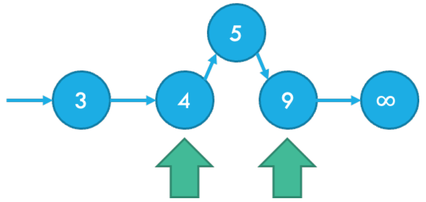
Insert(5)

Delete(9)



Insert(5)

Delete(9)



What does correctness mean?

Verification

system ⊨ specification

Testing

Hard to apply for concurrent systems

Formal verification

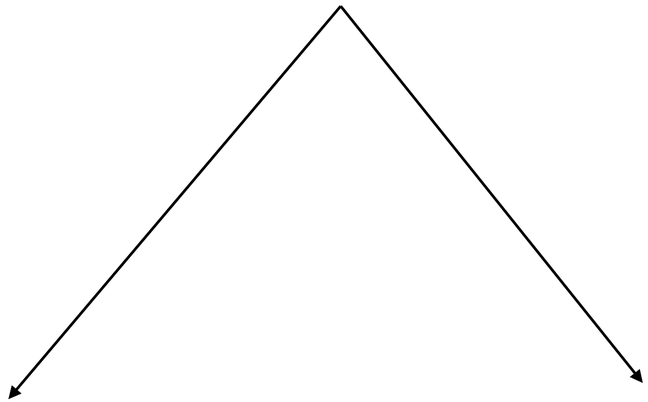
Even short concurrent programs are hard to analyze

Reasoning principles

Compositionality

Verification

system ⊨ specification



Safety:
nothing **bad** will happen

E.g., “at most one process in the critical section”

Liveness:
something **good** will happen (eventually)

E.g., “every request will finally be answered by the server”

This seminar

Goals

- Introduction different fundamental topics in concurrency *(basis for advanced studies)*
- Independent understanding of a scientific topic
- Understanding scientific literature
- Technical presentation skills

Requirements 1/2

- **Attend all meetings** (and actively participate).
- Present one subject in a **90 minute talk**, based on research papers or a chapter from a book.
- Should work in pairs.
- Prepare slides (pdf, in English), and send them to me **two weeks before the lecture**.
- Recommended: discuss presentations with me before the lecture.

Requirements 2/2

- Each lecture should include **four** “closed questions” (using mentimeter) to verify understanding of the material. At least one of them in the very end.

- **Grade**:
  90%: meeting these requirements (including sending presentation on time); understanding of the material; quality and clarity of presentation in class; quality of the slides/handouts.
  10%: best 80% answers in polls during the semester.

Question

- How many questions should each presentation include?

  A. 1

  B. 2

  C. 3

  D. 4

  E. 5

  F. 6

Your presentations

- This is an advanced seminar: the material is sometimes not easy and not self-contained.
- Identify and present the crux, rather than all details.
- May (and should!) skip details.
- Demonstrate with *clear and effective examples*.
- Be precise.
- Initiate participation and discussion (e.g., **ask thought provoking questions!**).

Your presentations

- Use a **blank** background
- May (and often should) use material available online (related papers and surveys, lecture notes, slides, videos).
- List the sources you use and give credits in the second slide of your presentation
- Do **not** copy-paste as is

Some tips

- Take your ***time*** to understand the material → start soon!
- Discuss the content with me and other students.
- Practice your talk out loud.

Topics and Schedule

See Google Doc

Logistics

- Website:

https://www.cs.tau.ac.il/~orilahav/seminar23/index.html

- By next week: topic assignments