Seminar on Concurrency Theory

Ori Lahav

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

March 3, 2021

Today

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

About me

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

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Postdoctoral researcher
Program verification
Host: M. Sagiv

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

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

MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

Since 2017 - Faculty member
Tel Aviv University

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

Main areas of research:

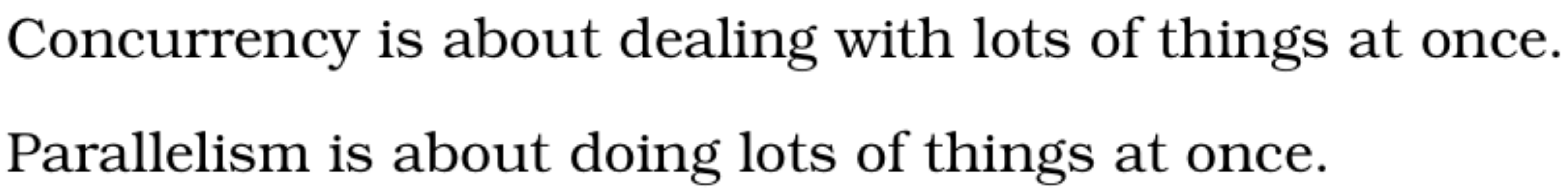
- Programming languages theory
- Verification
- Concurrency
- Relaxed memory models

Teaching this semester:

- Shared memory concurrency semantics (0368-4217)
- Seminar in concurrency theory (0368-3114)

Concurrency theory

- Rigorous mathematical formalisms and techniques for **modeling** and **analyzing** concurrent systems.
- Concurrent systems include concurrent programs & reactive systems.
- Concurrent doesn't necessarily mean parallel.
  סמינר בתיאוריה של בו-זמניות (?)
- Particular focus on communication and synchronization
  (rather than simple parallelism).



Concurrency is about dealing with lots of things at once.

Parallelism is about doing lots of things at once.

Rob Pike - 'Concurrency Is Not Parallelism'

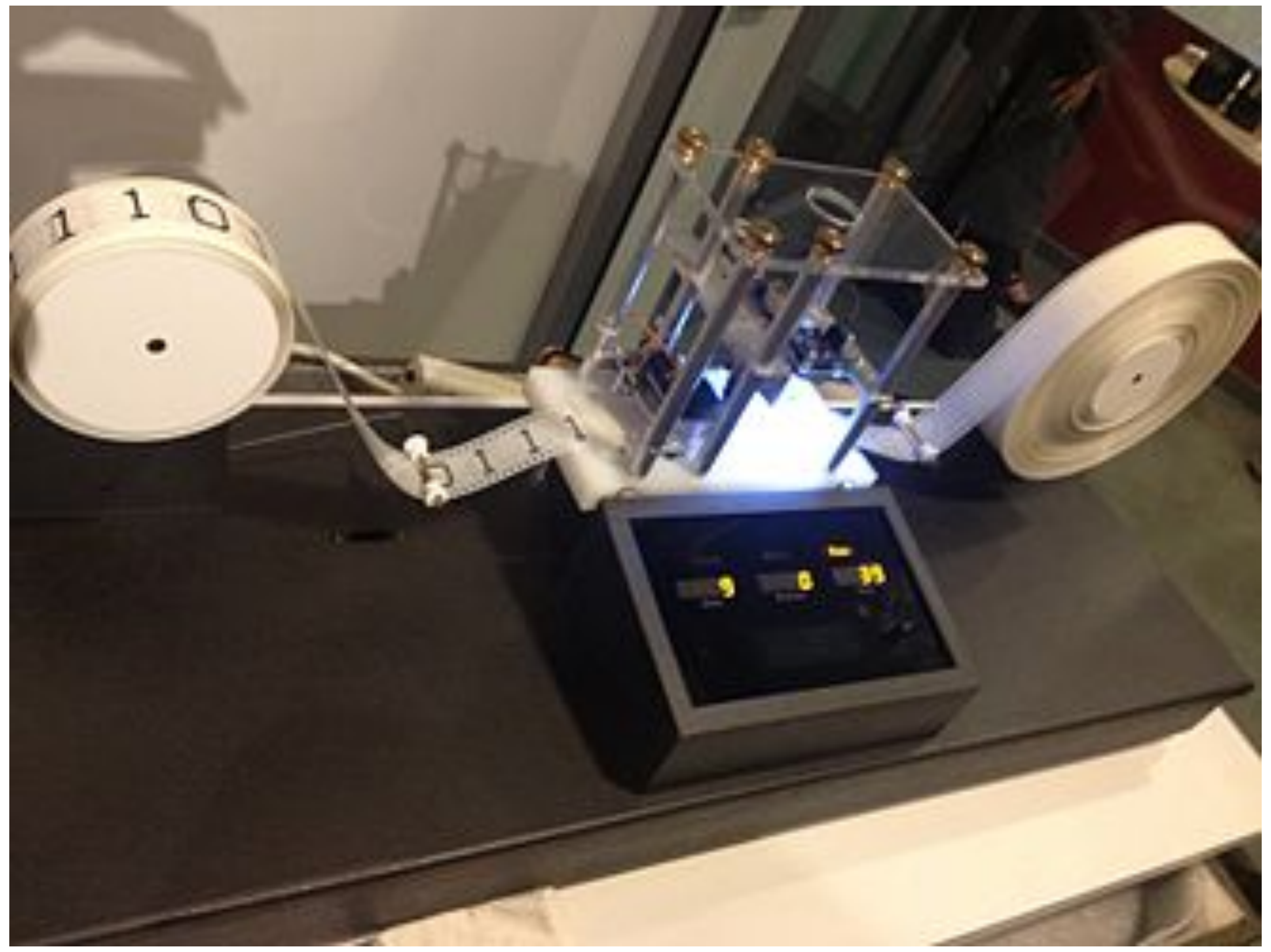
Reactive systems

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?*

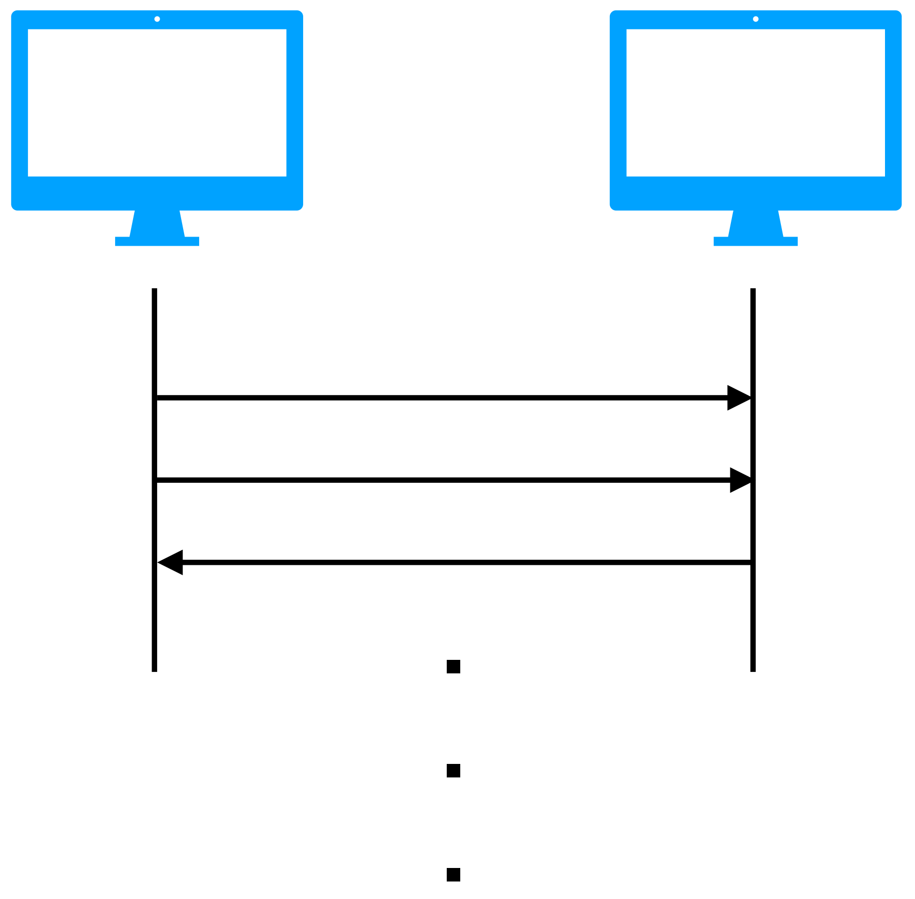


Reactive systems

- What about: operating systems? websites? database systems? power plants? vending machines?

Reactive systems continuously reacts to the environment and influence the environment

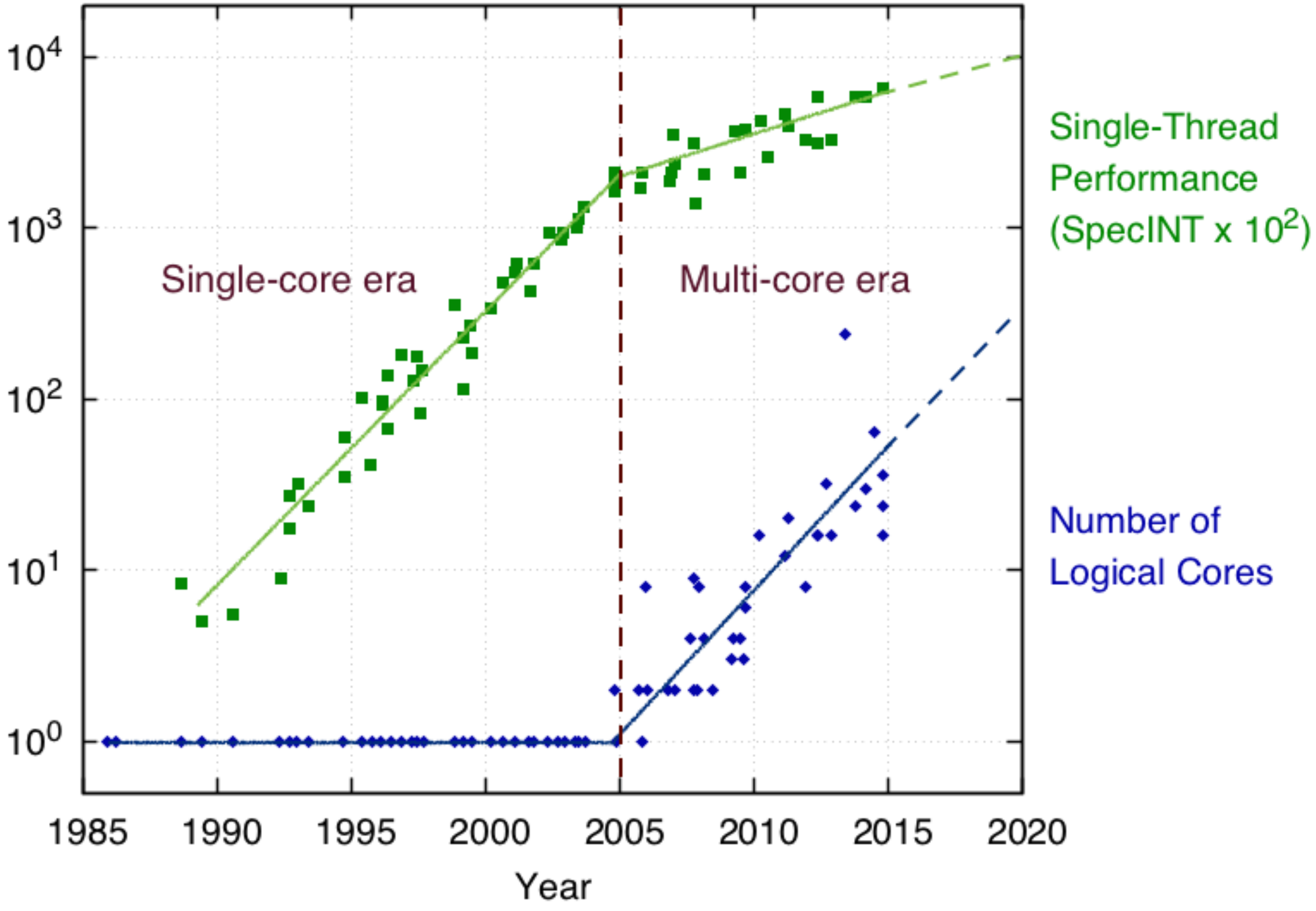
- Key issue: communication and interaction.
- Non-determinism is often inevitable.
- What is correctness?
  - Often halting is actually a problem.
  - Not crashing (e.g., “dividing by 0”).
  - Serving requests on time.
  - Adhering to certain communication protocols.
- What is equivalence? refinement?

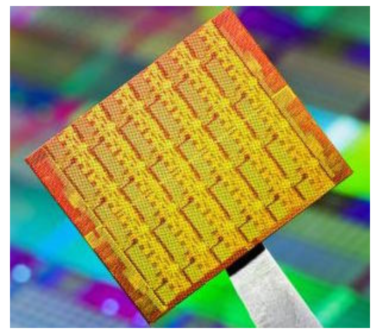


Concurrent programming

Parallelism is here

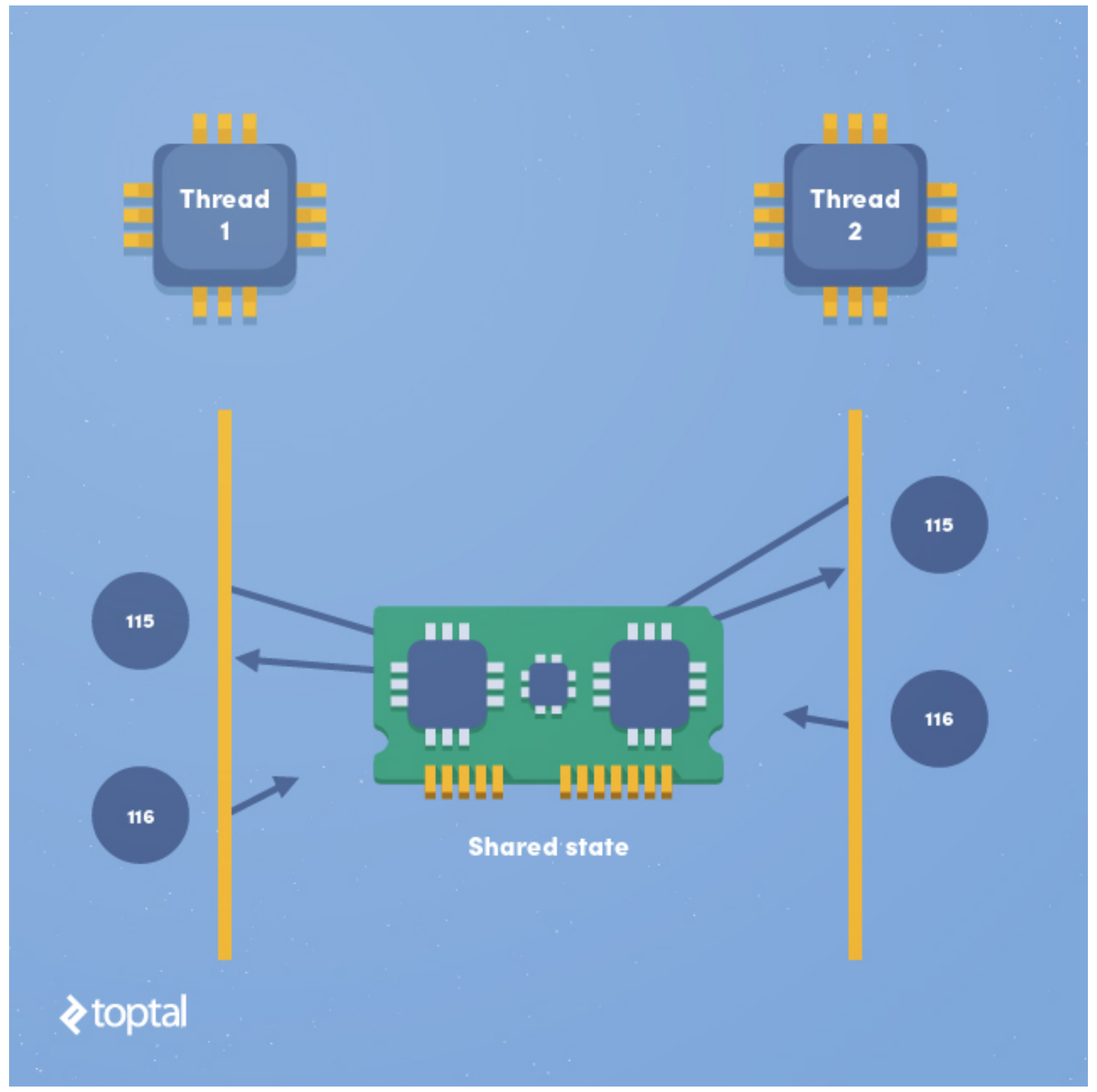
“The Free Lunch Is Over: A Fundamental Turn Toward Concurrency in Software”.
By Herb Sutter (2005)





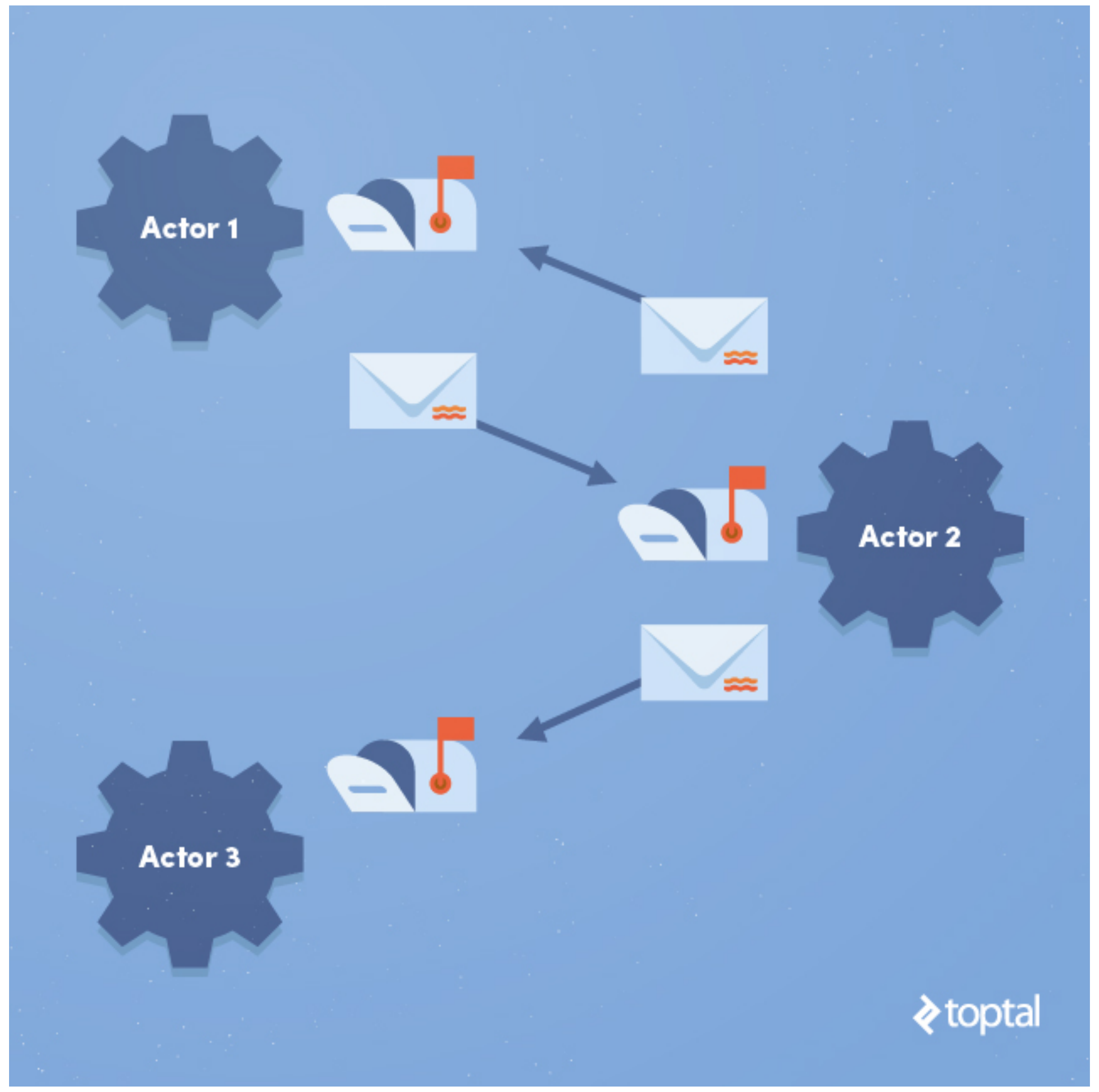
Two fundamental models of concurrent programming

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

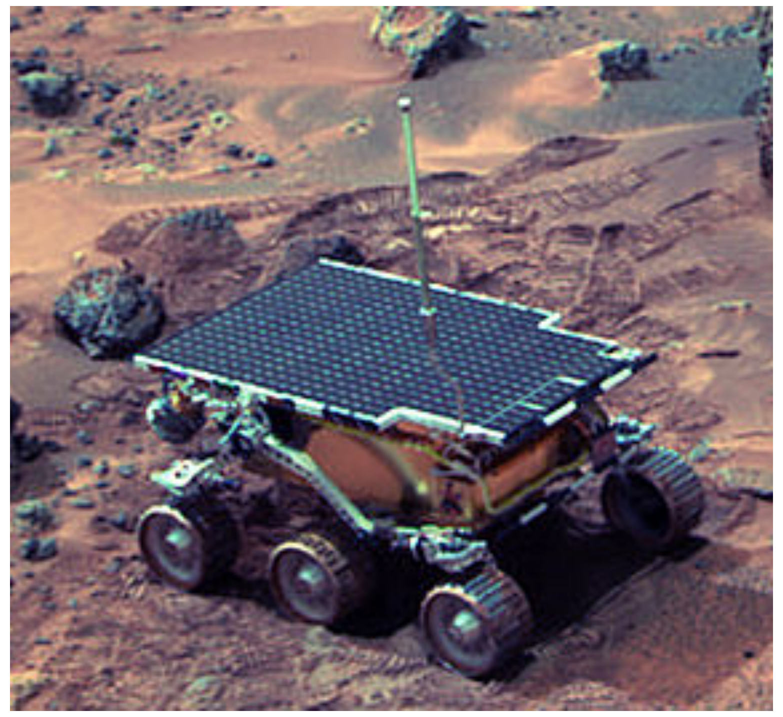
Hard to get right!

- 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!

Verification

system $\models$ specification

Testing → **Formal verification**

Hard to apply for concurrent systems

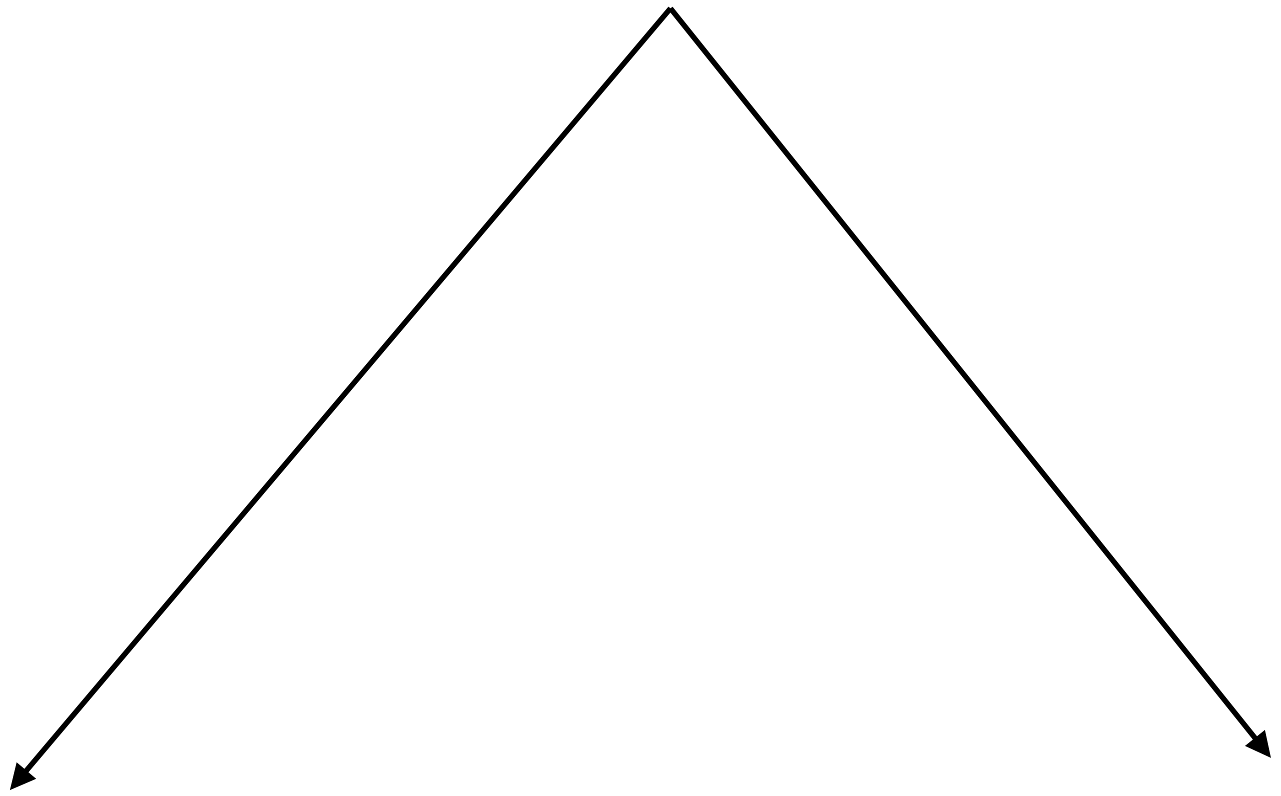
Even short concurrent programs are hard to analyze

Reasoning principles

Compositionality

Verification

system $\models$ 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 (by zoom with enabled video) and actively participate.
- Present one subject in a 70-90 minute talk, based on a research paper or a chapter from a book.
- Should work in pairs (*interleaved not parallel…*).
- Prepare slides (pdf, in English), and send them to me **two weeks before the lecture**.
- Discuss presentations with me a week before the lecture.

Requirements 2/2

- Each lecture should include three “closed questions” (using zoom polls) to verify understanding of the material. At least one of them in the very end.
- Answers to there polls will be used for **attendance check**.
- **Grade**:
  95%: meeting these requirements (including sending presentation on time); understanding of the material; quality and clarity of presentation in class; quality of the slides/handouts.
  5%: best 80% answers in polls during the semester.

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.
- Demonstrate with *clear and effective examples*.
- Be precise.
- May (and often should) skip proof details.
- 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

March 3	Ori	Introduction and guidelines [slides]
March 10	Dvir, Mor	Transition systems and behavioral equivalences [Chapter 2 in Introduction to Concurrency Theory by Gorrieri&Versari]
March 17	Dor, Topaz	Calculus of communicating systems (CCS) [Chapter 3 in Introduction to Concurrency Theory by Gorrieri&Versari]
April 7		A Very Gentle Introduction to Multiparty Session Types Nobuko Yoshida, Lorenzo Gher Distributed Computing and Internet Technology. ICDCIT 2020. Springer. [1]
April 21		An axiomatic proof technique for parallel programs I Susan S. Owicki, David Gries Acta Informatica 6: 319-340, 1976 [1]
April 28		The rely-guarantee method for verifying shared variable concurrent programs Qiwen Xu, Willem-Paul de Roever, Jifeng He Formal Aspects of Computing 9: 149-174, 1997 [1]
May 5		Separation logic: a logic for shared mutable data structures John C. Reynolds Proceedings 17th Annual IEEE Symposium on Logic in Computer Science, Copenhagen, Denmark, 2002, pp. 55-74 [1]
May 12		Resources, concurrency and local reasoning Peter W. O'Hearn Theoretical Computer Science 375, 1-3: 271-307, 2007 [1] [recent CACM article]
May 19		Linearizability: a correctness condition for concurrent objects Maurice P. Herlihy, Jeannette M. Wing ACM Trans. Program. Lang. Syst. 12, 3: 463-492, 1990 [1]
May 26		Wait-free synchronization Maurice Herlihy. ACM Trans. Program. Lang. Syst. 13, 1: 124-149, 1991 [1]
June 2		Laws of order: expensive synchronization in concurrent algorithms cannot be eliminated Hagit Attiya, Rachid Guerraoui, Danny Hendler, Petr Kuznetsov, Maged M. Michael, Martin Vechev In Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on principles of programming languages (POPL '11). ACM, New York, NY, USA, 487-498 [1]
June 9		FastTrack: efficient and precise dynamic race detection Cormac Flanagan, Stephen N. Freund Proceedings of the 30th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '09). ACM, New York, NY, USA, 121–133 [1]
June 16		Conflict-free Replicated Data Types: An Overview Nuno Preguiça [1]

Logistics

- Website:

https://www.cs.tau.ac.il/~orilahav/seminar21/index.html

- By next week: topic assignments