Seminar on Concurrency Theory

Ori Lahav

March 9, 2020

Today

- What is this seminar about?

- Goals, requirements and logistics of the seminar

About me

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

Teaching this semester:
- Foundations of programming languages (0368-3241)
- Seminar on formal theory of concurrency (0368-3114)

Concurrency theory

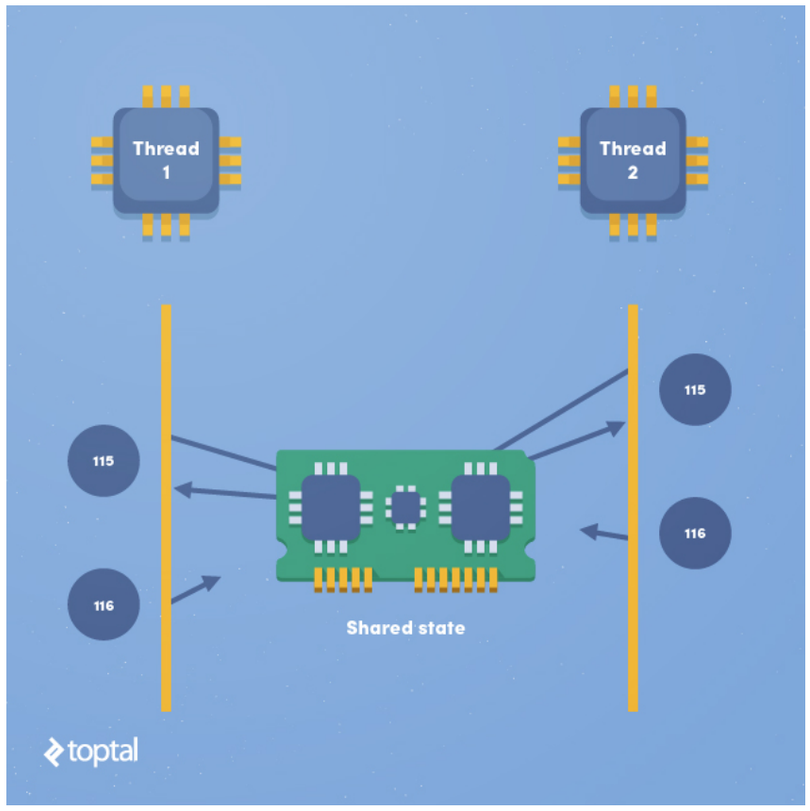
- Rigorous mathematical formalisms and techniques for **modeling** and **analyzing** concurrent systems.

- Concurrent systems:

    - Concurrent programs

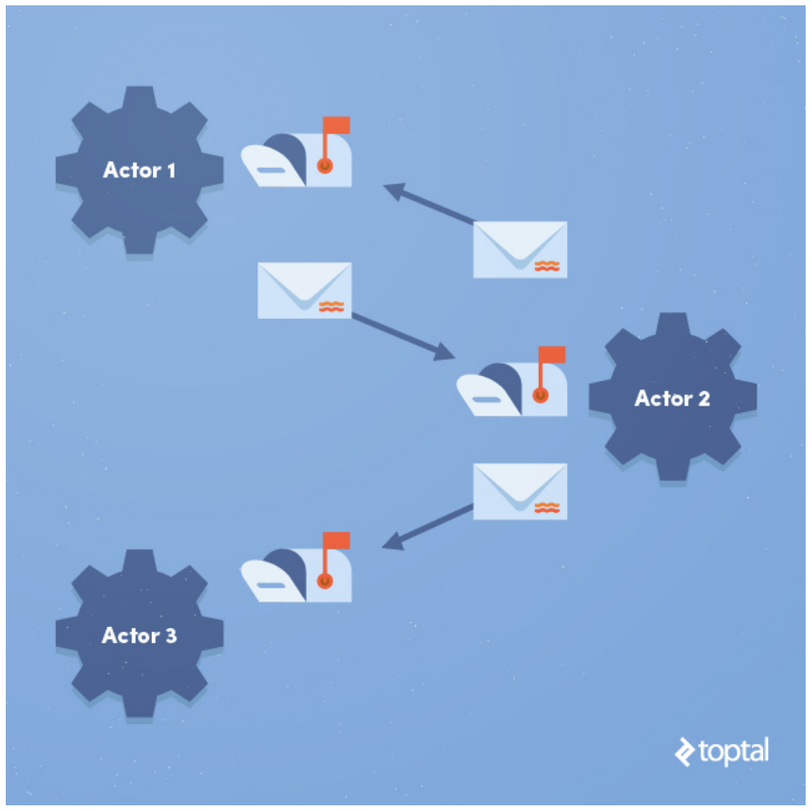
    - Reactive systems

Concurrent programming



shared memory

C / C++

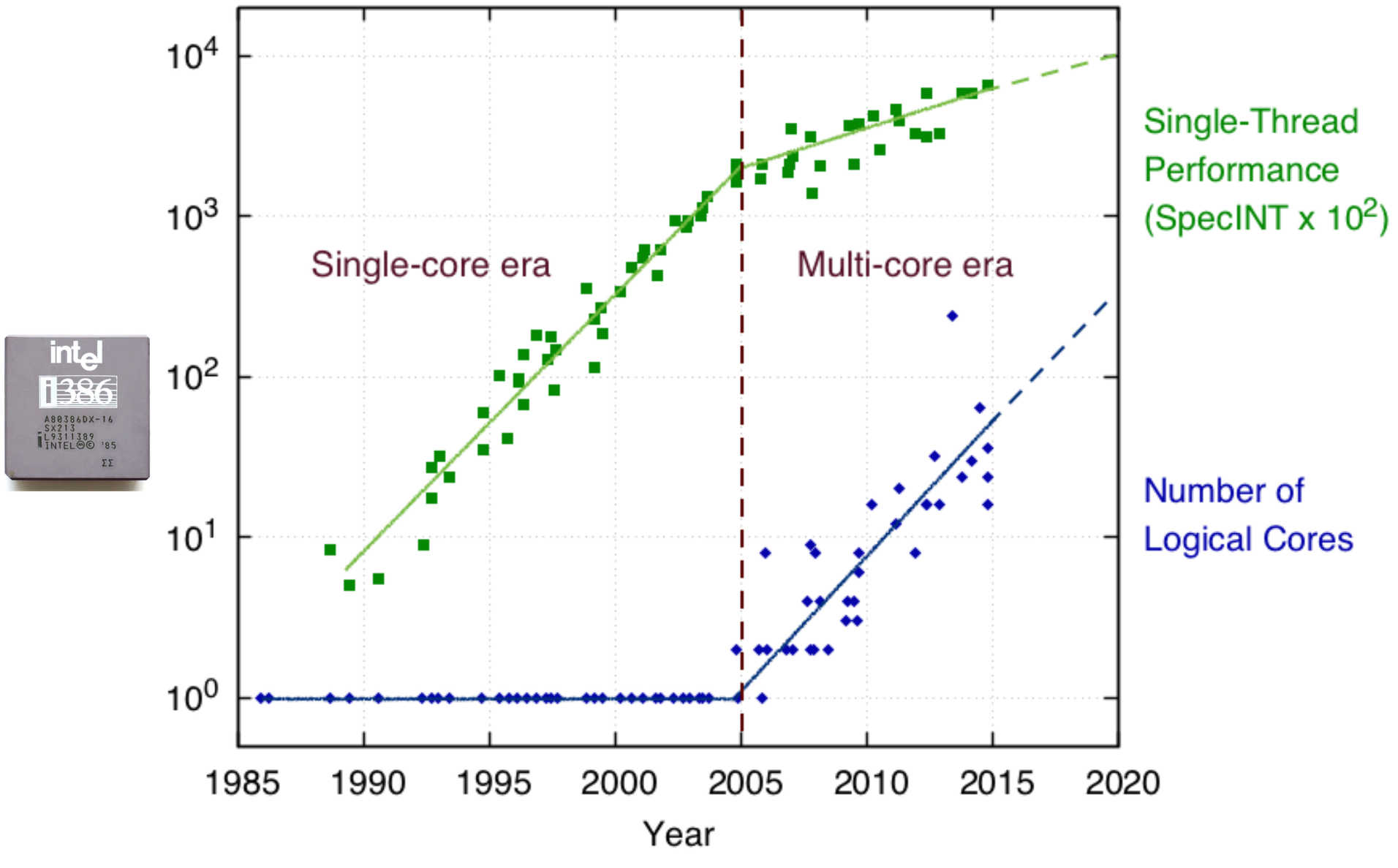


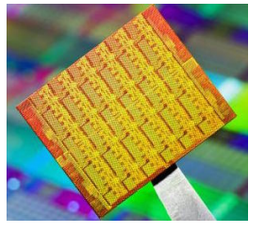
message passing

Erlang

Parallelism is here

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





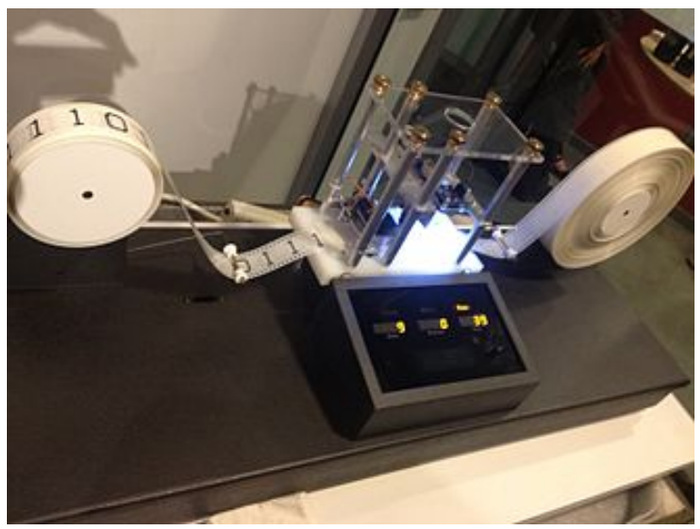
Reactive systems

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? communication protocol? control programs? database management 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.
- Related and similar to **parallelism**.

Problem

- Concurrency is widespread, but it is also **error prone**.
- **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

Example

Initially `X = 0`.

```
X := X+1;  ||  X := X+2;
```

- How many possible outcomes?

Verification

system ⊨ specification

Testing → **Formal verification**

Hard to apply for concurrent systems

Even short concurrent programs are hard to analyze

Randomization

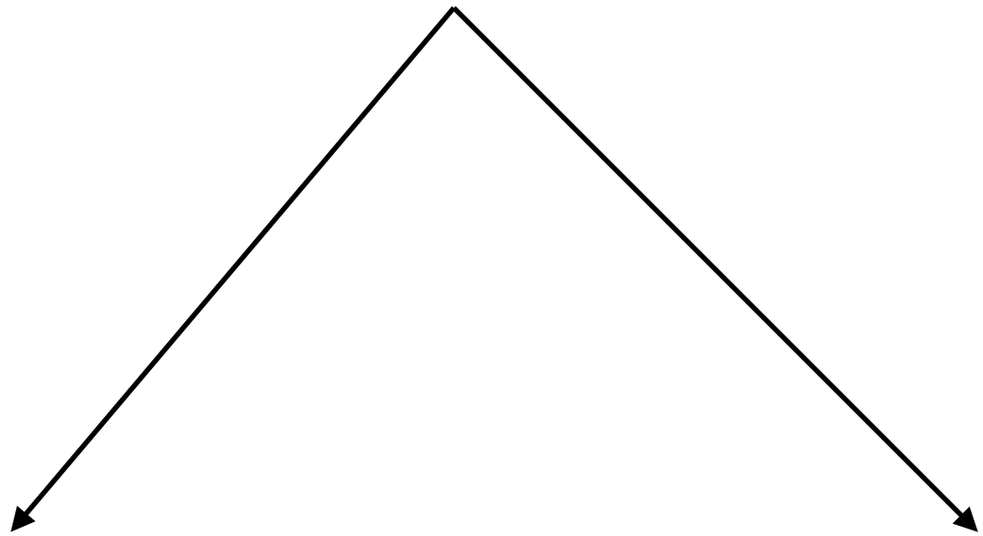
Reasoning principles

Automatic decision procedures

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”

Practice

- **Modeling** and simulating concurrent systems (different formalisms and tools)

- Concurrent **programming**: programming languages have different concurrency models in (e.g., C, Go, Rust, Erlang)

- **Verifying** correctness to avoid expensive bugs (many tools)

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

- Attend all meetings.
- Present one subject in a 70-90 minute talk, based on a research paper or a chapter from a book.
- Prepare slides and/or handouts (pdf, in English), and send them to me **two weeks before the lecture**.
- May work in pairs (present 2 subjects).
- Discuss presentations with me a week before the lecture.
- **Recommendation**: use other sources besides the one that will be suggested (with citations).
- **Grade**: meeting these requirements; understanding of the material; quality and clarity of presentation in class; quality of the slides/handouts.
- Optional: extra short lecture (e.g., introduce an alternative model of concurrency).

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 questions!).

Some tips

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

Logistics

- Website:

https://www.cs.tau.ac.il/~orilahav/seminar20/index.html

- By next week: topic assignments
  add your preferences in

https://docs.google.com/document/d/1x5UEIcWsh3l1ruzBD9OWOiI9i2U0msrl7hN3GXpHivU/edit?usp=sharing

- Speaker for next week?

Topics

- See website.
- You are welcome to suggest topics / papers!