Seminar on Concurrency Theory

Ori Lahav

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

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:
• 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**.

• 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


**Reactive systems** continuously reacts to the environment and influences 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

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 \( X = 0 \).

\[
X := X+1; \quad X := X+3;
\]

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

$\text{system} \models \text{specification}$

Testing

Hard to apply for concurrent systems

Formal verification

Even short concurrent programs are hard to analyze

Reasoning principles

Compositionality
Verification

\[ \text{system} \models \text{specification} \]

**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.
<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter(s)</th>
<th>Topic</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>March 3</td>
<td>Ori</td>
<td>Introduction and guidelines</td>
<td>[slides]</td>
</tr>
<tr>
<td>March 10</td>
<td>Dvir, Mor</td>
<td>Transition systems and behavioral equivalences</td>
<td>Chapter 2 in Introduction to Concurrency Theory by Gorrieri &amp; Versari</td>
</tr>
<tr>
<td>March 17</td>
<td>Dor, Topaz</td>
<td>Calculus of communicating systems (CCS)</td>
<td>Chapter 3 in Introduction to Concurrency Theory by Gorrieri &amp; Versari</td>
</tr>
<tr>
<td>April 7</td>
<td></td>
<td>A Very Gentle Introduction to Multiparty Session Types</td>
<td>Nobuko Yoshida, Lorenzo Gher</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distributed Computing and Internet Technology. ICDCIT 2020. Springer.</td>
<td></td>
</tr>
<tr>
<td>April 21</td>
<td></td>
<td>An axiomatic proof technique for parallel programs I</td>
<td>Susan S. Owicki, David Gries</td>
</tr>
<tr>
<td>April 28</td>
<td></td>
<td>The rely-guarantee method for verifying shared variable concurrent programs</td>
<td>Qiwen Xu, Willem-Paul de Roever, Jifeng He</td>
</tr>
<tr>
<td>May 5</td>
<td></td>
<td>Separation logic: a logic for shared mutable data structures</td>
<td>John C. Reynolds</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proceedings 17th Annual IEEE Symposium on Logic in Computer Science, Copenhagen, Denmark, 2002, pp. 55-74</td>
<td></td>
</tr>
<tr>
<td>May 12</td>
<td></td>
<td>Resources, concurrency and local reasoning</td>
<td>Peter W. O'Hearn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1] recent CACM article</td>
<td></td>
</tr>
<tr>
<td>May 19</td>
<td></td>
<td>Linearizability: a correctness condition for concurrent objects</td>
<td>Maurice P. Herlihy, Jeannette M. Wing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>May 26</td>
<td></td>
<td>Wait-free synchronization</td>
<td>Maurice Herlihy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>June 2</td>
<td></td>
<td>Laws of order: expensive synchronization in concurrent algorithms cannot be eliminated</td>
<td>Hagit Attiya, Rachid Guerraoui, Danny Hendler, Petr Kuznetsov, Maged M. Michael, Martin Vechev</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>June 9</td>
<td></td>
<td>FastTrack: efficient and precise dynamic race detection</td>
<td>Cormac Flanagan, Stephen N. Freund</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[1]</td>
<td></td>
</tr>
<tr>
<td>June 16</td>
<td></td>
<td>Conflict-free Replicated Data Types: An Overview</td>
<td>Nuno Preguiça</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Logistics

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

• By next week: topic assignments