Concurrency Theory

Winter 2025/26

Lecture 1: Introduction

Thomas Noll, Peter Thiemann
Programming Languages Group
University of Freiburg

https://proglang.github.io/teaching/25ws/ct.html

Thomas Noll, Peter Thiemann

Winter 2025/26

Outline of Lecture 1

Staff

- Lectures:
  - Peter Thiemann
- Exercises:
  - Marius Weidner
- Contact: `weidner@informatik.uni-freiburg.de`

Target Audience

- Master Computer Science
- Specialization Cyber-Physical Systems

Target Audience

- Master Computer Science
- Specialization Cyber-Physical Systems
- In general:
  - interest in formal models for concurrent (software) systems
  - application of mathematical modelling and reasoning methods
  - *not* (in the first place): concurrent programming
- Expected: basic knowledge in
  - essential concepts of operating systems and system software
  - formal languages and automata theory
  - mathematical logic

Course Objectives

Objectives

- Understand the foundations of concurrent systems
- Understand the main semantical underpinnings of concurrency
- Model, reason about, and compare concurrent systems in a rigorous manner

Course Objectives

Objectives

- Understand the foundations of concurrent systems
- Understand the main semantical underpinnings of concurrency
- Model, reason about, and compare concurrent systems in a rigorous manner

Motivation

- Supporting the design phase of systems
  - “Programming Concurrent Systems”
  - synchronisation, scheduling, semaphores, ...
- Verifying functional correctness properties
  - “Model Checking”
  - validation of mutual exclusion, fairness, absence of deadlocks, ...
- Comparing expressivity of models of concurrency

Organisation

- All material (slides, videos, exercise sheets, ...) made available via lecture website
- Schedule:
  - Lecture Tue 14:00–16:00, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 21)
  - Exercise Mon 10–12, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 27)

Organisation

- All material (slides, videos, exercise sheets, ...) made available via lecture website
- Schedule:
  - Lecture Tue 14:00–16:00, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 21)
  - Exercise Mon 10–12, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 27)
- Assignment sheets:
  - in weekly intervals, starting Tue Oct 21
  - no submission required, but consider it as exam preparation

Organisation

- All material (slides, videos, exercise sheets, ...) made available via lecture website
- Schedule:
  - Lecture Tue 14:00–16:00, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 21)
  - Exercise Mon 10–12, R 04 007 Videokonferenz G.-Köhler-Allee 106 (starting Oct 27)
- Assignment sheets:
  - in weekly intervals, starting Tue Oct 21
  - no submission required, but consider it as exam preparation
- Exam (6 CP):
  - written
  - TBA
  - no specific admission requirements

Outline of Lecture 1

Observation: concurrency introduces new phenomena

Example 1.1

$$\begin{gathered} x := 0; \\ (x := x + 1 \parallel x := x + 2) \end{gathered}$$

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- At first glance: x is assigned 3

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$1$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$1 \qquad\qquad\qquad 2$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(\underset{1}{x := x + 1} \parallel \underset{2}{x := x + 2}) \qquad \text{value of } x\text{: } 1$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel \underset{2}{x} := x + 2) \qquad \text{value of } x\text{: } 2$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2,

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2,

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$

$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

$$1$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2,

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(\underset{1}{x := x + 1} \parallel \underset{2}{x := x + 2}) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2,

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$

$$(\underset{1}{x := x + 1} \parallel \underset{2}{x := x + 2}) \qquad \text{value of } x\text{: } 2$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2,

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 1$$
$$1$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1,

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1,

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 0$$
$$2$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1,

Observation: concurrency introduces new phenomena

Example 1.1

$$
\begin{array}{c}
x := 0; \\
(x := x + 1 \parallel x := \underbrace{x + 2}_{2})
\end{array}
\qquad \text{value of } x\text{: } 2
$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1,

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$

$$(x := \underset{3}{x + 1} \parallel x := x + 2) \qquad \text{value of } x\text{: } 2$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1,

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2) \qquad \text{value of } x\text{: } 3$$
$$3$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1, or 3

Concurrency and Interaction by Example

Observation: concurrency introduces new phenomena

Example 1.1

$$x := 0;$$
$$(x := x + 1 \parallel x := x + 2)$$

- At first glance: x is assigned 3
- But: both parallel components might read x before it is written
- Thus: x is assigned 2, 1, or 3
- If exclusive access to shared memory and atomic execution of assignments guaranteed
  $\Longrightarrow$ only possible outcome: 3

Concurrency and Interaction

The problem arises due to the combination of

- concurrency and
- interaction (here: via shared memory)

The problem arises due to the combination of

- concurrency and
- interaction (here: via shared memory)

Conclusion

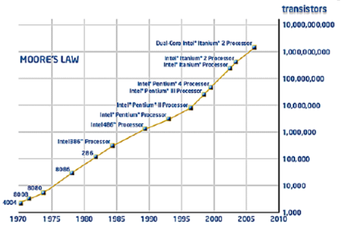
When modelling concurrent systems, the precise description of the mechanisms of both concurrency and interaction is crucially important.

Concurrency Everywhere

Herb Sutter: *The Free Lunch Is Over*, Dr. Dobb's Journal, 30(3), 2005

"The biggest sea change in software development since the OO revolution is knocking at the door, and its name is Concurrency."

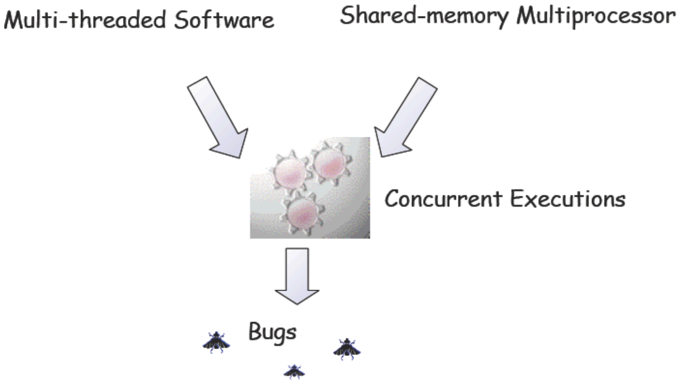
- Operating systems
- Embedded/reactive systems
  - parallelism (at least) between hardware, software, and environment
- High-end parallel hardware infrastructure:
  - high-performance computing
- Low-end parallel hardware infrastructure
  - increasing performance only achievable by parallelism
  - multi-core computers, GPGPUs, FPGAs



Moore's Law: Transistor density doubles every 2 years

Problems Everywhere

- Operating systems:
  - mutual exclusion
  - fairness (no starvation)
  - no deadlocks, ...
- Shared-memory systems:
  - memory models
  - data races
  - inconsistencies ("sequential consistency" vs. relaxed notions)
- Embedded systems:
  - safety
  - liveness, ...

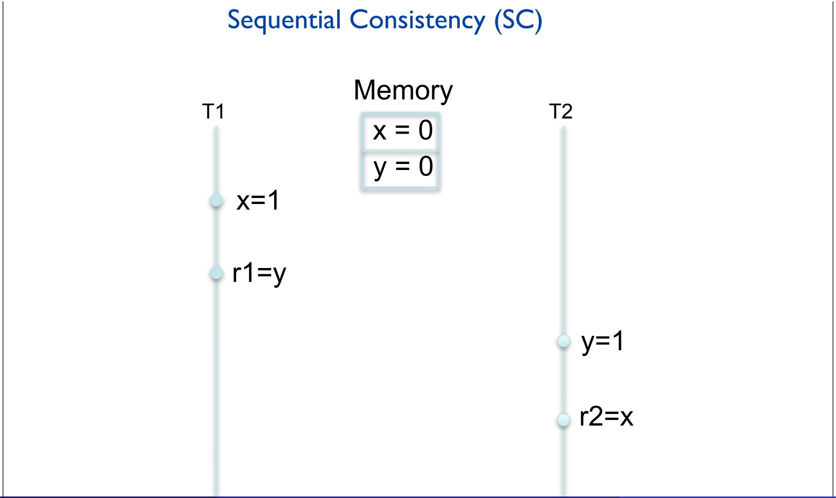


Outline of Lecture 1

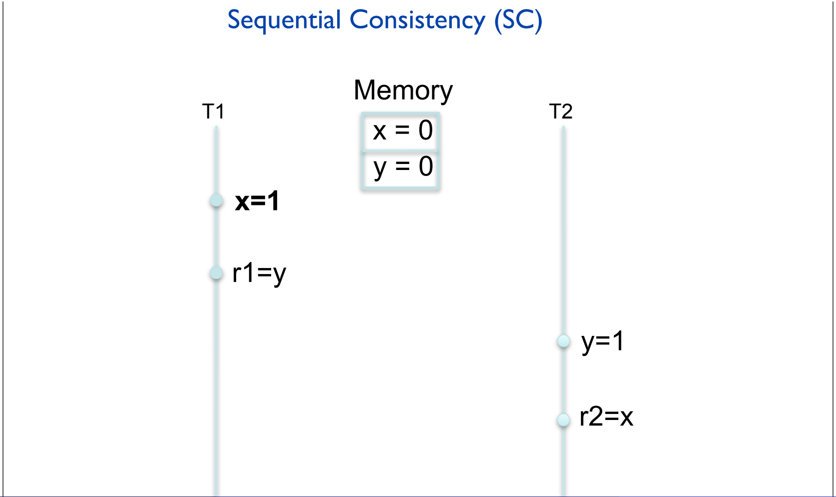
An illustrative example

Initially: x = y = 0

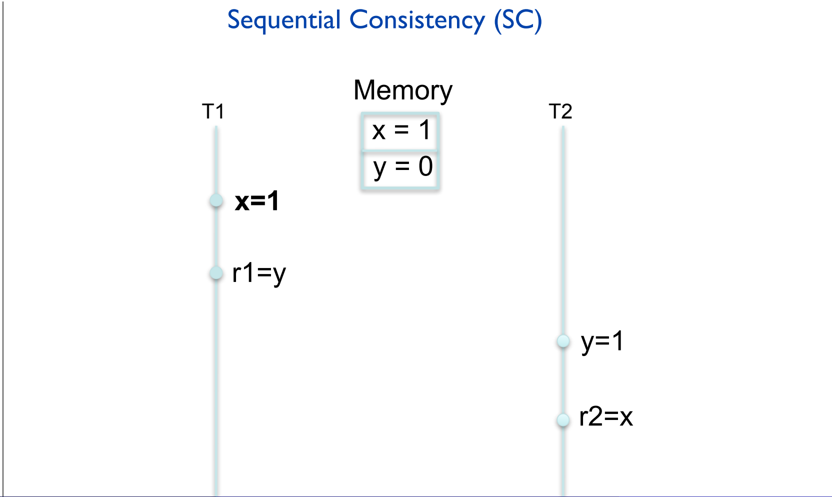
thread1:	thread2:
1: x = 1	3: y = 1
2: r1 = y	4: r2 = x


Sequential Consistency (SC)
Memory
T1
x = 0
y = 0
T2
x=1
r1=y
y=1
r2=x

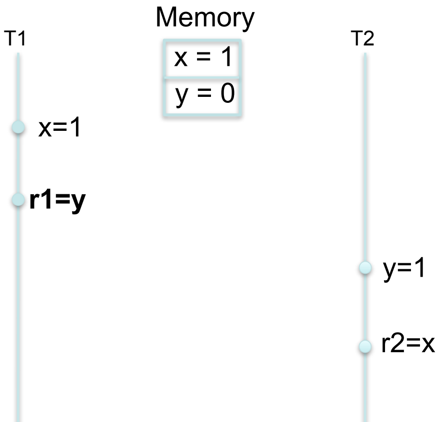
Memory Models

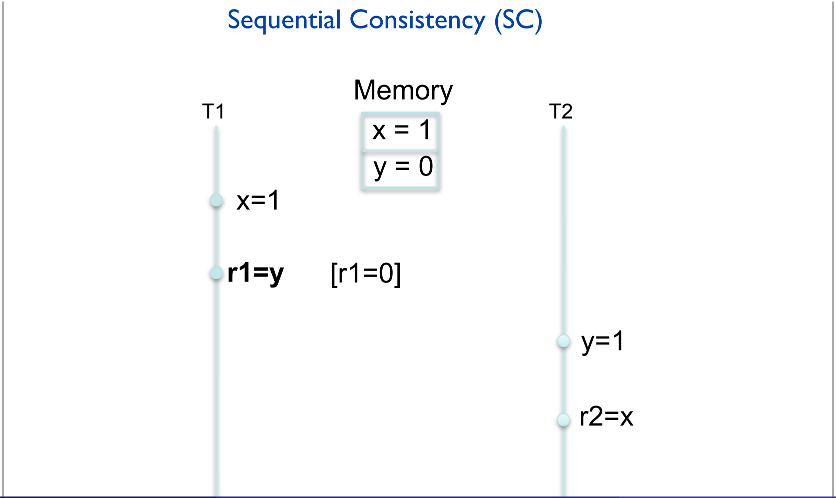


Memory Models

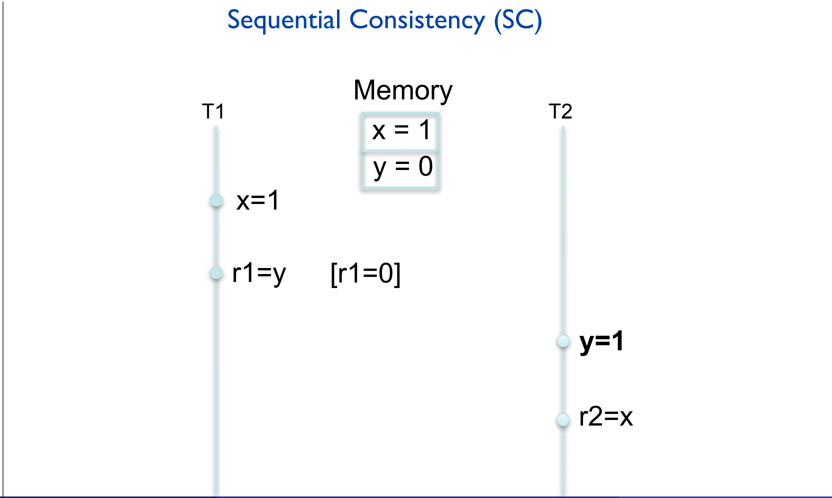


Sequential Consistency (SC)



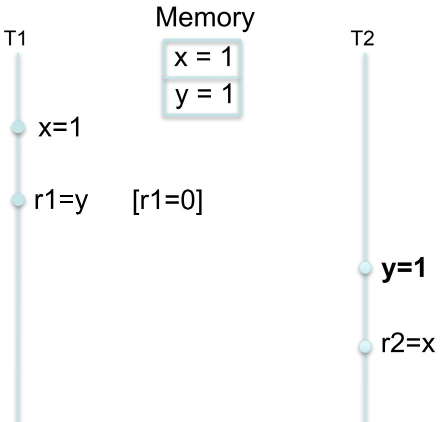


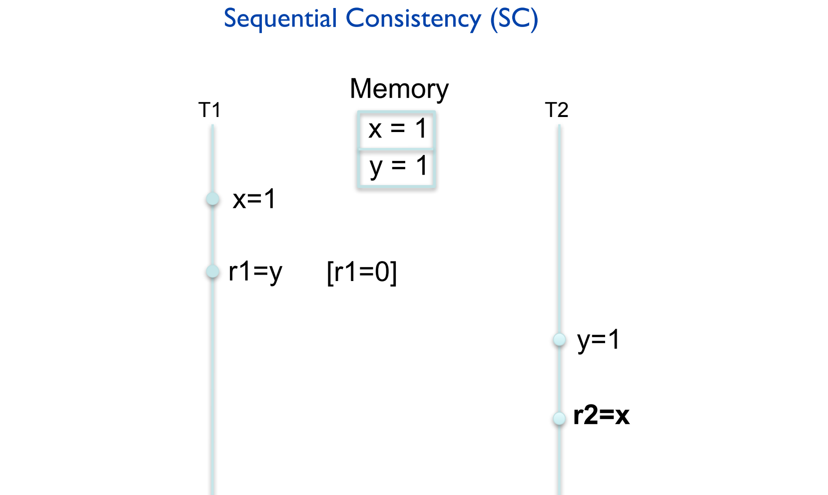
Sequential Consistency (SC)
Memory
T1
T2
x = 1
y = 0
x=1
r1=y
[r1=0]
y=1
r2=x

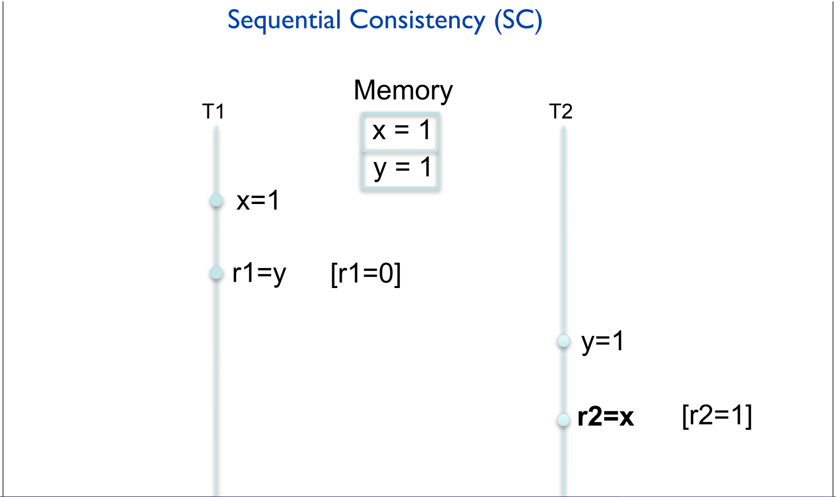

Sequential Consistency (SC)
Memory
T1
x = 1
y = 0
T2
x=1
r1=y
[r1=0]
y=1
r2=x

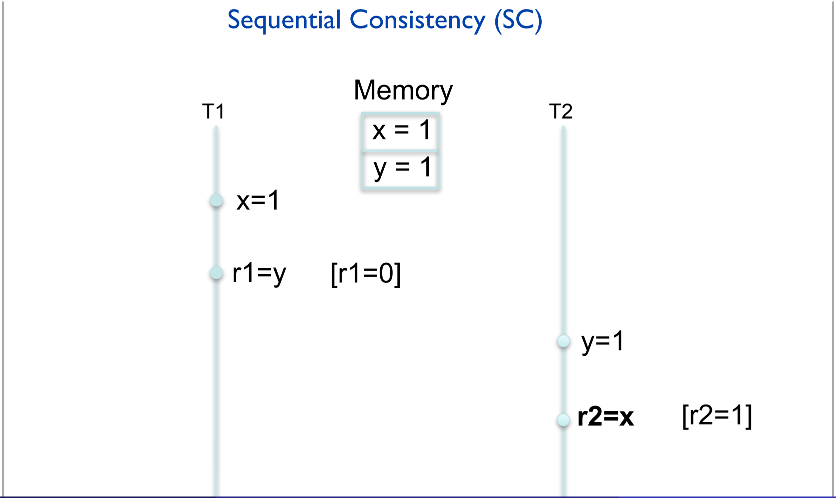
Memory Models

Sequential Consistency (SC)

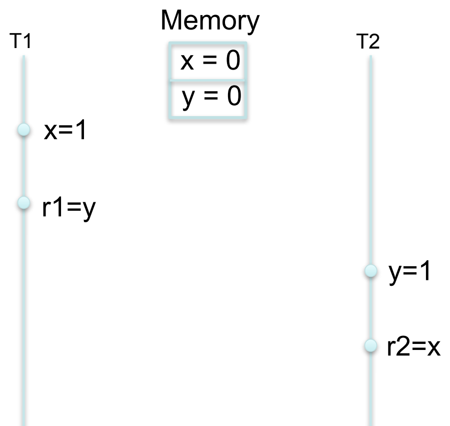


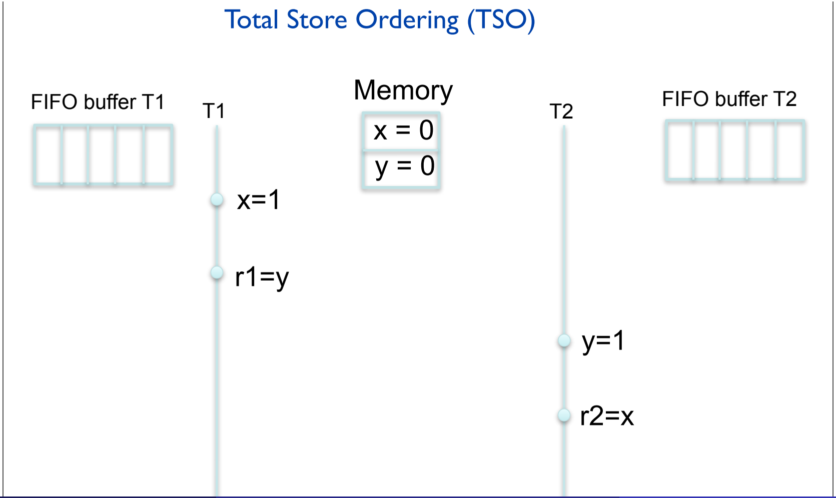

Sequential Consistency (SC)
Memory
T1
T2
x = 1
y = 1
x=1
r1=y
[r1=0]
y=1
r2=x


Sequential Consistency (SC)
Memory
T1
T2
x = 1
y = 1
x=1
r1=y
[r1=0]
y=1
r2=x
[r2=1]

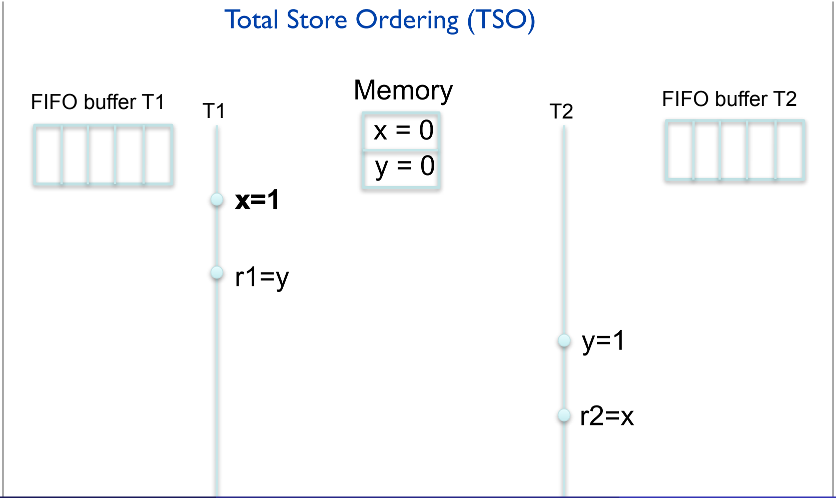

Sequential Consistency (SC)
Memory
T1
T2
x = 1
y = 1
x=1
r1=y
[r1=0]
y=1
r2=x
[r2=1]

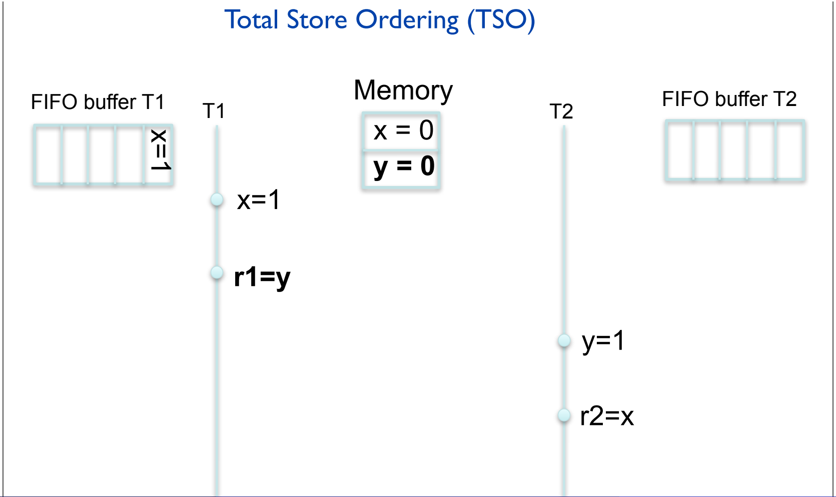
Total Store Ordering (TSO)



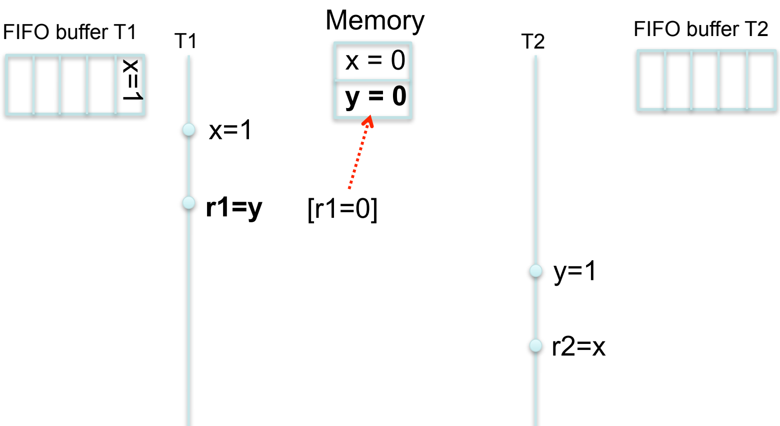

Total Store Ordering (TSO)
FIFO buffer T1
T1
Memory
x = 0
y = 0
T2
FIFO buffer T2
x=1
r1=y
y=1
r2=x

Memory Models

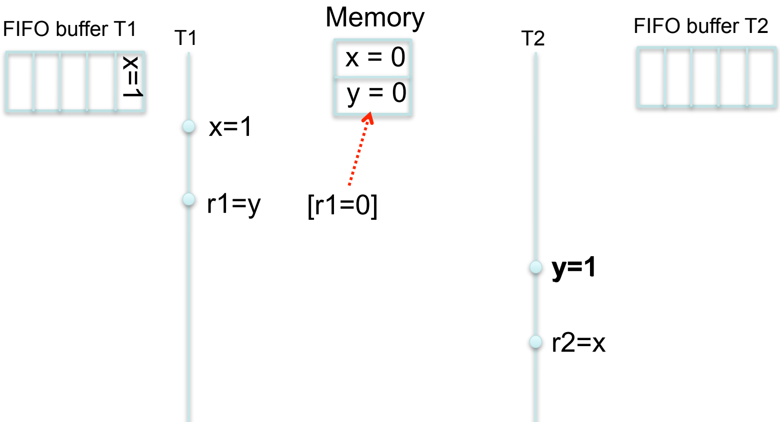



Total Store Ordering (TSO)
FIFO buffer T1
x=1
T1
Memory
x = 0
y = 0
T2
FIFO buffer T2
x=1
r1=y
y=1
r2=x

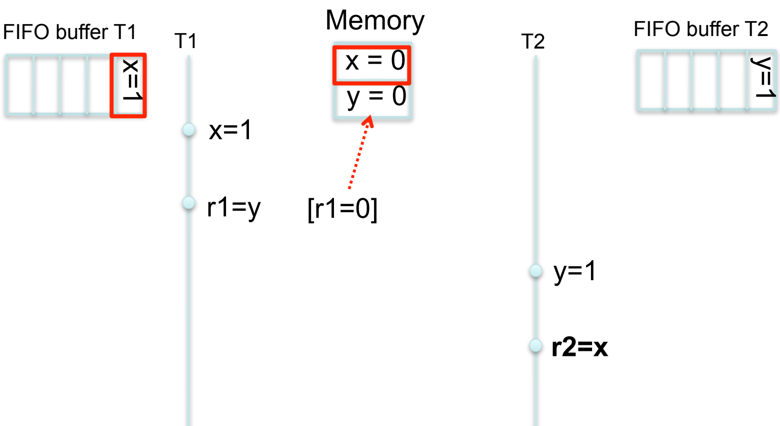
Total Store Ordering (TSO)

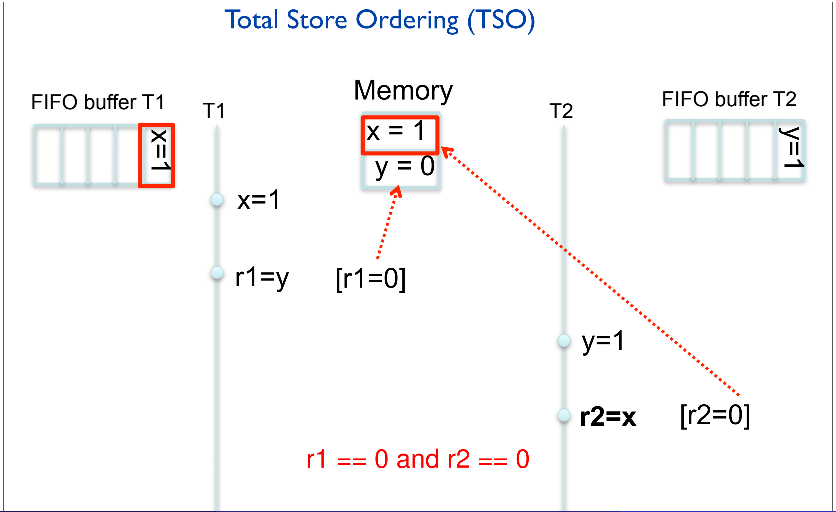


Total Store Ordering (TSO)



Total Store Ordering (TSO)




Total Store Ordering (TSO)
FIFO buffer T1
x=1
T1
Memory
x = 1
y = 0
T2
FIFO buffer T2
y=1
x=1
r1=y
[r1=0]
y=1
r2=x
[r2=0]
r1 == 0 and r2 == 0

Outline of Lecture 1

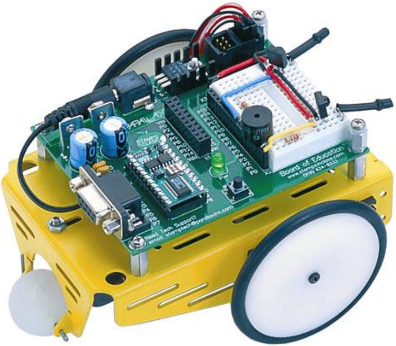
Reactive Systems I

- “Classical” model for sequential systems

$$\mathit{System} : \mathit{Input} \rightarrow \mathit{Output}$$

  (transformational systems) is not adequate

- Missing: aspect of interaction

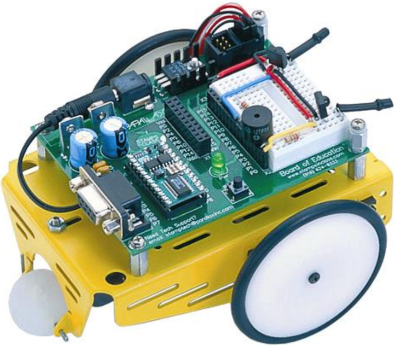


Reactive Systems I

- "Classical" model for sequential systems

$$System : Input \rightarrow Output$$

  (transformational systems) is not adequate
- Missing: aspect of interaction
- Rather: reactive systems which interact with environment and among themselves

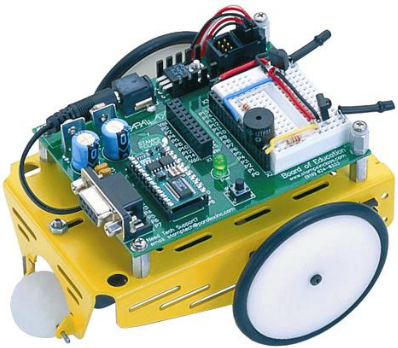


Reactive Systems I

- "Classical" model for sequential systems

$$\textit{System} : \textit{Input} \rightarrow \textit{Output}$$

  (transformational systems) is not adequate
- Missing: aspect of interaction
- Rather: reactive systems which interact with environment and among themselves
- Main interest: not terminating computations but infinite behaviour (system maintains ongoing interaction with environment)
- Examples:
  - operating systems
  - embedded systems controlling mechanical or electrical devices (planes, cars, home appliances, ...)



Reactive Systems II

Observation

Reactive systems are often safety critical, thus trustworthiness has to be ensured.

- Safety properties: “Nothing bad is ever going to happen.”
  - e.g., “at most one process in the critical section”
- Liveness properties: “Eventually something good will happen.”
  - e.g., “every request will finally be answered by the server”
- Fairness properties: “No component will starve to death.”
  - e.g., “any process requiring entry to the critical section will eventually be admitted”
- Reliability, performance, survivability, ...

Outline of Lecture 1

Overview of the Course

(1) Introduction and Motivation

(2) The “Interleaving” Approach

- Syntax and semantics of CCS
- Hennessy-Milner Logic
- Case study: mutual exclusion
- Extensions, alternative approaches (value passing, mobility, CSP, ACP, ...)

(3) Equivalence, Refinement and Compositionality

- Behavioural equivalences ((bi-)simulation)
- Case study: mutual exclusion
- (Pre-)congruences and compositional abstraction
- HML and bisimilarity

(4) The “True Concurrency” Approach

- Petri nets: basic concepts
- Case study: mutual exclusion
- Branching processes and net unfoldings
- Analysing Petri nets

Literature

- Fundamental:
  - Luca Aceto, Anna Ingólfsdóttir, Kim Guldstrand Larsen and Jiří Srba: *Reactive Systems: Modelling, Specification and Verification*, Cambridge University Press, 2007
  - Wolfgang Reisig: *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*, Springer Verlag, 2012
- Supplementary:
  - Jan Bergstra, Alban Ponse and Scott Smolka (Eds.): *Handbook of Process Algebra*, Elsevier, 2001
  - Maurice Herlihy and Nir Shavit: *The Art of Multiprocessor Programming*, Elsevier, 2008
  - Davide Sangiorgi and David Walker: *The Pi-Calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001