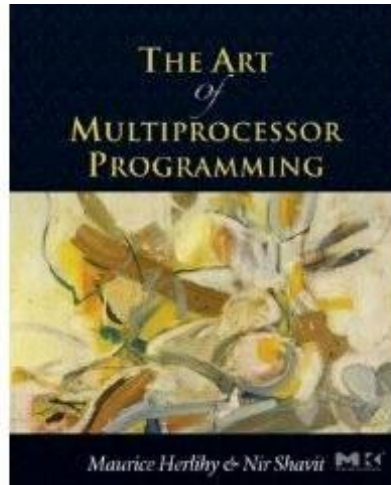
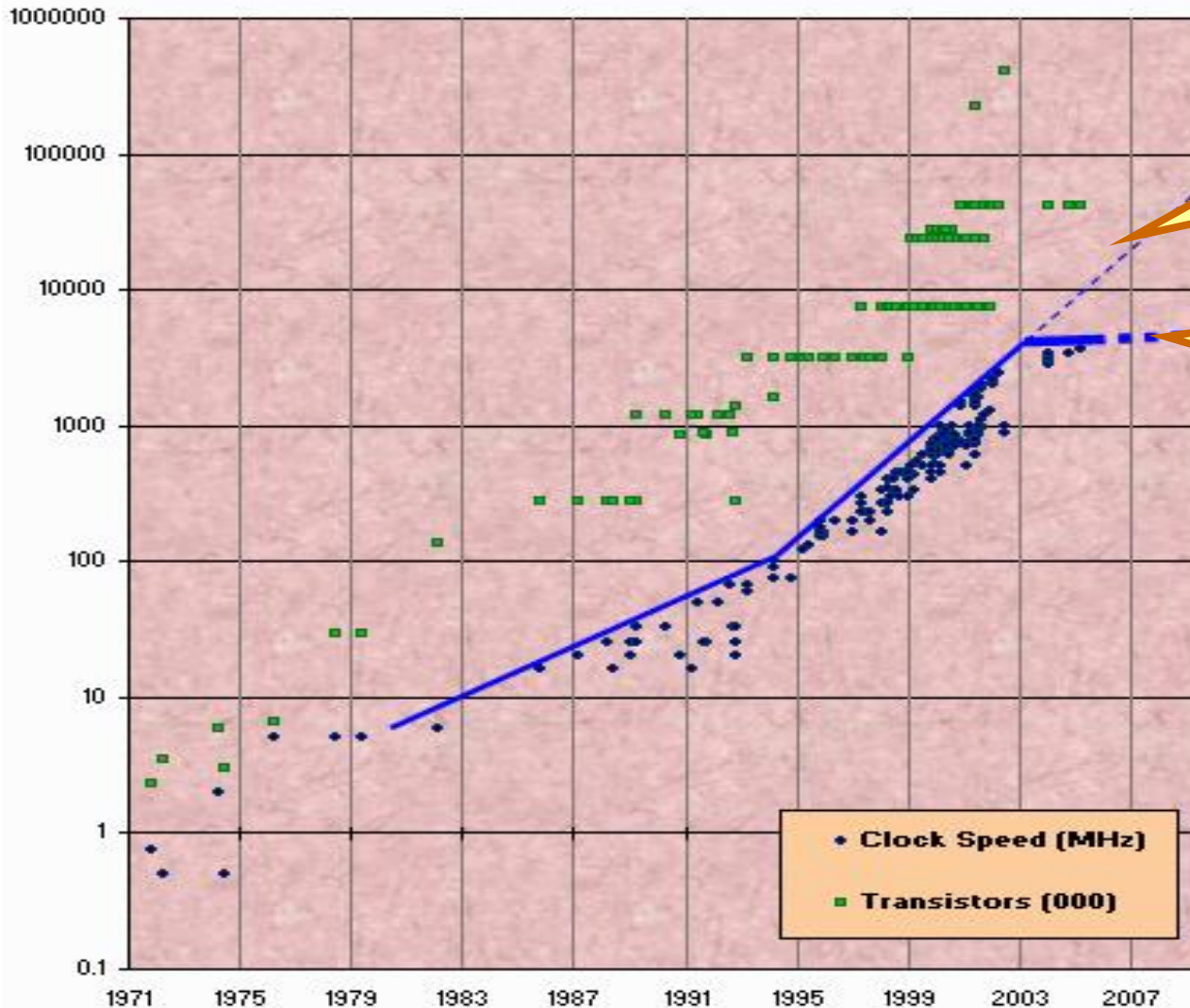


Introduction



Companion slides for
The Art of Multiprocessor Programming
by Maurice Herlihy & Nir Shavit

Moore's Law

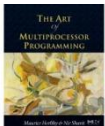


Transistor count still rising

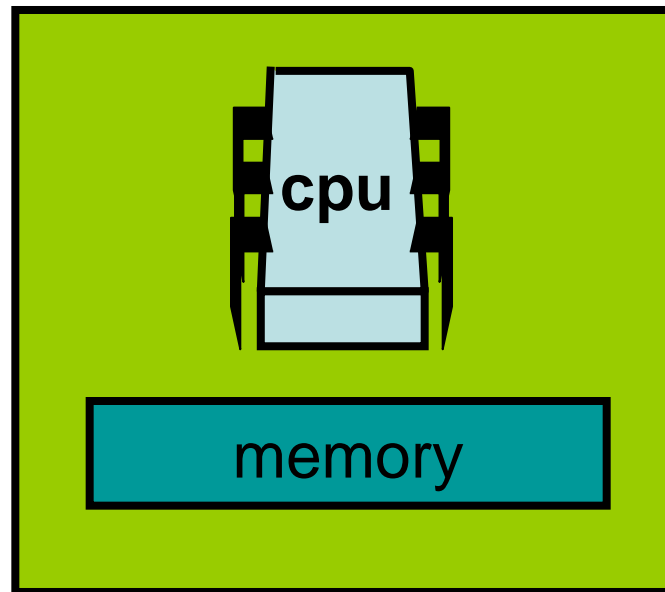
Clock speed flattening sharply



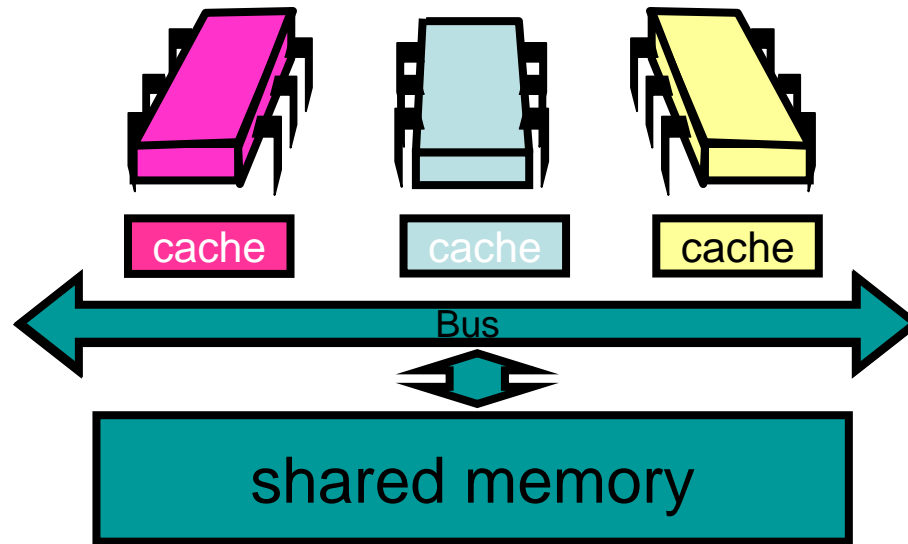
Moore's Law (in practice)



Nearly Extinct: the Uniprocessor

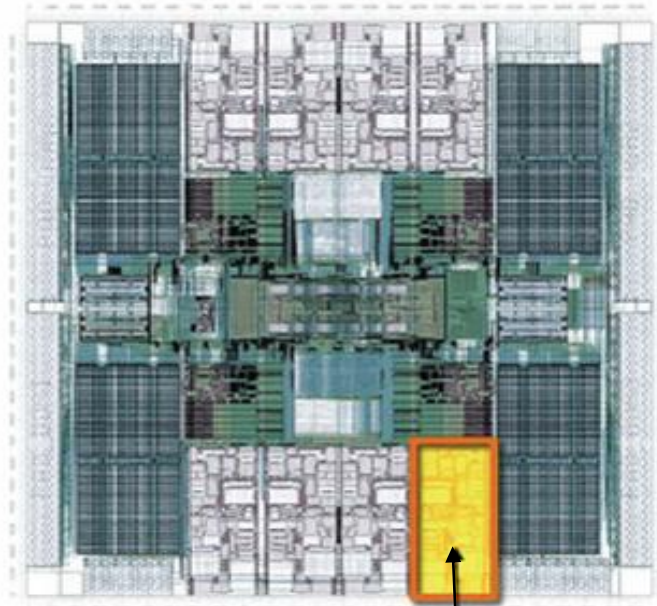


Endangered: The Shared Memory Multiprocessor (SMP)

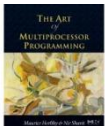
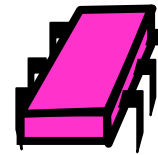


The New Boss: The Multicore Processor (CMP)

All on the
same chip

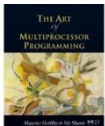


Sun
T2000
Niagara



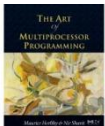
Why do we care?

- We want as much as possible to execute concurrently (in parallel)
- A larger sequential part implies reduced performance
- **Amdahl's law:** this relation is not linear...



Amdahl's Law

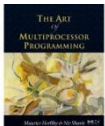
$$\text{Speedup} = \frac{\text{1-thread execution time}}{\text{\textit{n}-thread execution time}}$$



Amdahl's Law

Speedup=

$$\frac{1}{1 - i + \frac{p}{n}}$$

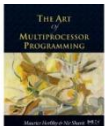


Amdahl's Law

Speedup=

$$\frac{1}{1 + \frac{p}{n}}$$

Parallel fraction



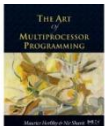
Amdahl's Law

Speedup=

**Number of
threads**

$$\frac{1}{1 + \frac{p}{n}}$$

Parallel fraction



Amdahl's Law

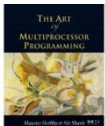
Sequential fraction

Speedup =

Parallel fraction

Number of threads

$$1 + \frac{p}{n}$$



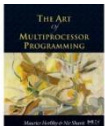
Amdahl's Law in Practice



Bad synchronization ruins everything

Example

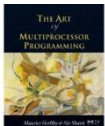
- Ten processors
- 60% concurrent, 40% sequential
- How close to 10-fold speedup?



Example

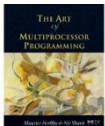
- Ten processors
- 60% concurrent, 40% sequential
- How close to 10-fold speedup?

$$\text{Speedup} = 2.17 = \frac{1}{1 - 0.6 + \frac{0.6}{10}}$$



Example

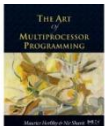
- Ten processors
- 80% concurrent, 20% sequential
- How close to 10-fold speedup?



Example

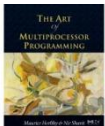
- Ten processors
- 80% concurrent, 20% sequential
- How close to 10-fold speedup?

$$\text{Speedup} = 3.57 = \frac{1}{1 - 0.8 + \frac{0.8}{10}}$$



Example

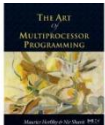
- Ten processors
- 90% concurrent, 10% sequential
- How close to 10-fold speedup?



Example

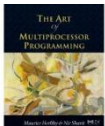
- Ten processors
- 90% concurrent, 10% sequential
- How close to 10-fold speedup?

$$\text{Speedup} = 5.26 = \frac{1}{1 - 0.9 + \frac{0.9}{10}}$$



Example

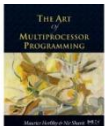
- Ten processors
- 99% concurrent, 01% sequential
- How close to 10-fold speedup?



Example

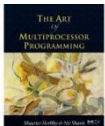
- Ten processors
- 99% concurrent, 01% sequential
- How close to 10-fold speedup?

$$\text{Speedup} = 9.17 = \frac{1}{1 - 0.99 + \frac{0.99}{10}}$$



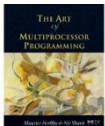
Concurrent Objects

- What is a concurrent object?
 - How do we **describe** one?
 - How do we **implement** one?
 - How do we **tell if we're right**?



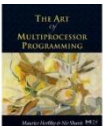
Sequential Objects

- Each object has a ***state***
 - Usually given by a set of ***fields***
 - Queue example: sequence of items
- Each object has a set of ***methods***
 - Only way to manipulate state
 - Queue example: **enq** and **deq** methods



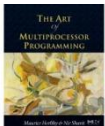
Sequential Specifications

- If (precondition)
 - the object is in such-and-such a state
 - before you call the method,
- Then (postcondition)
 - the method will return a particular value
 - or throw a particular exception.
- and (postcondition, con't)
 - the object will be in some other state
 - when the method returns,



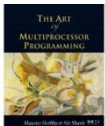
Pre and PostConditions for Dequeue

- **Precondition:**
 - Queue is non-empty
- **Postcondition:**
 - Returns first item in queue
- **Postcondition:**
 - Removes first item in queue



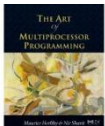
Pre and PostConditions for Dequeue

- **Precondition:**
 - Queue is empty
- **Postcondition:**
 - Throws Empty exception
- **Postcondition:**
 - Queue state unchanged



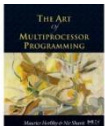
Sequential Specifications

- Interactions among methods captured by side-effects on object state
 - State meaningful between method calls
- Documentation size linear in number of methods
 - Each method described in isolation
- Can add new methods
 - Without changing descriptions of old methods

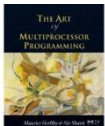
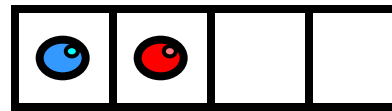


What About Concurrent Specifications ?

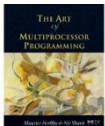
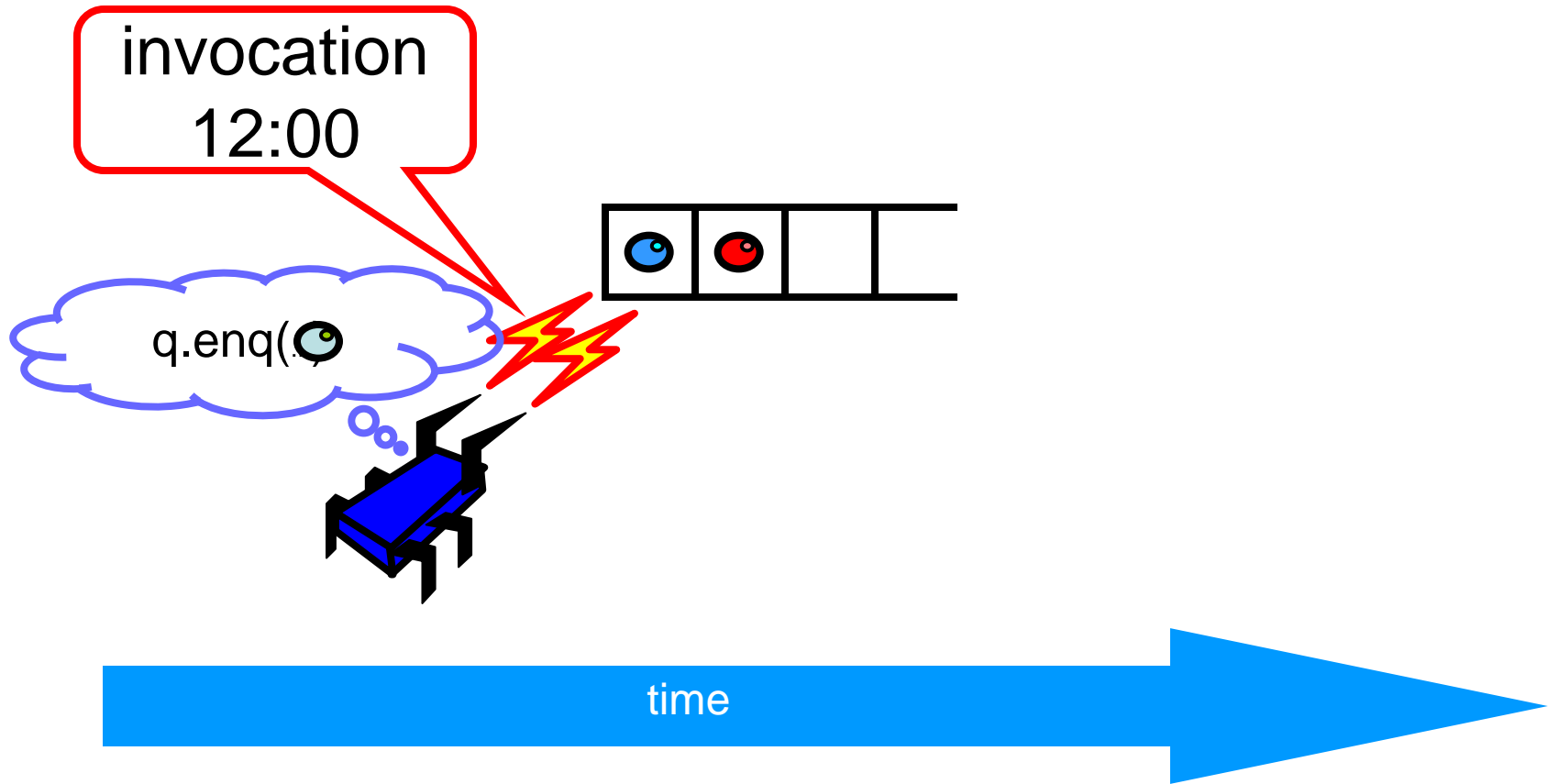
- Methods?
- Documentation?
- Adding new methods?



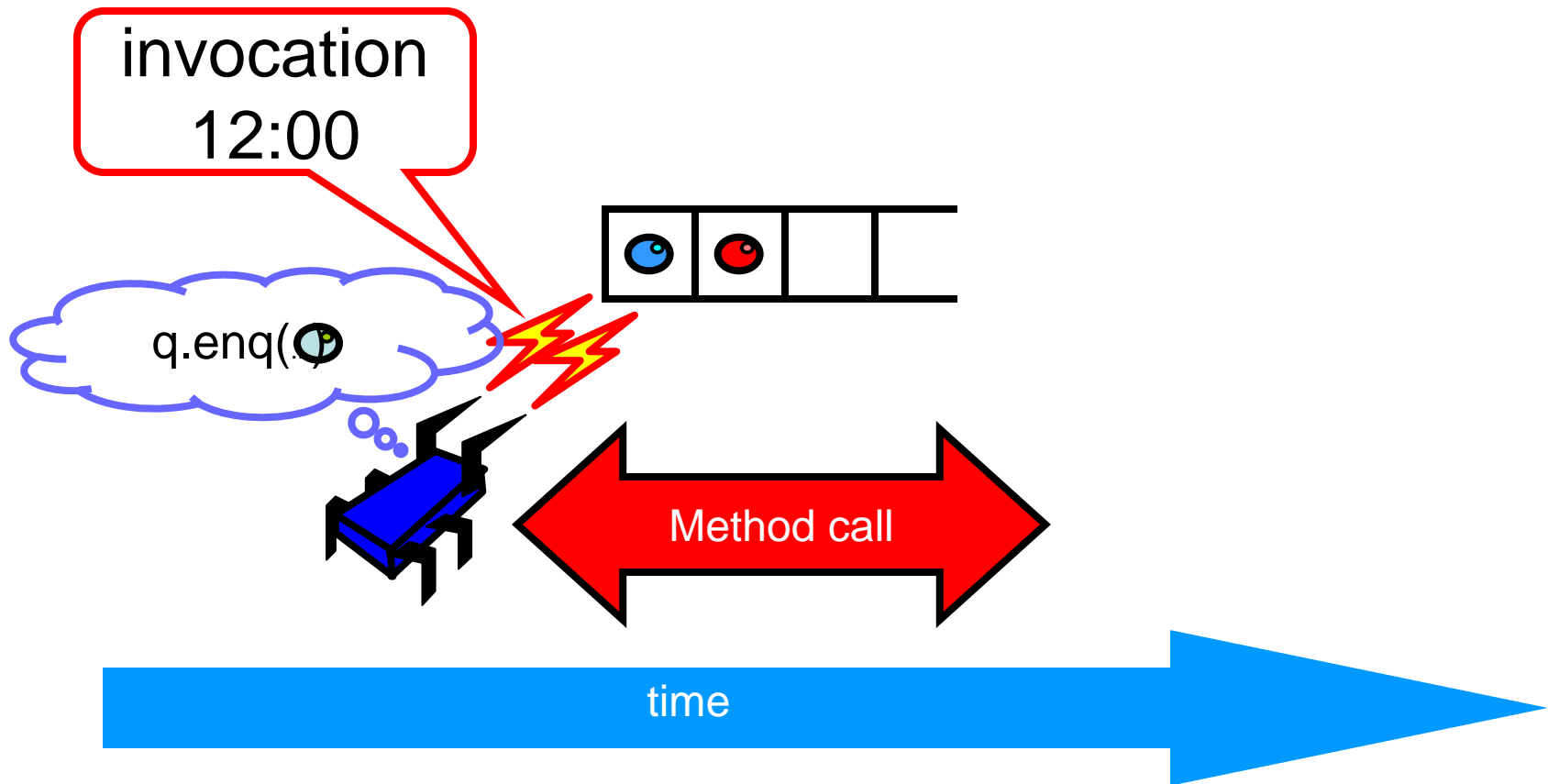
Methods Take Time



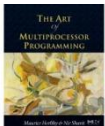
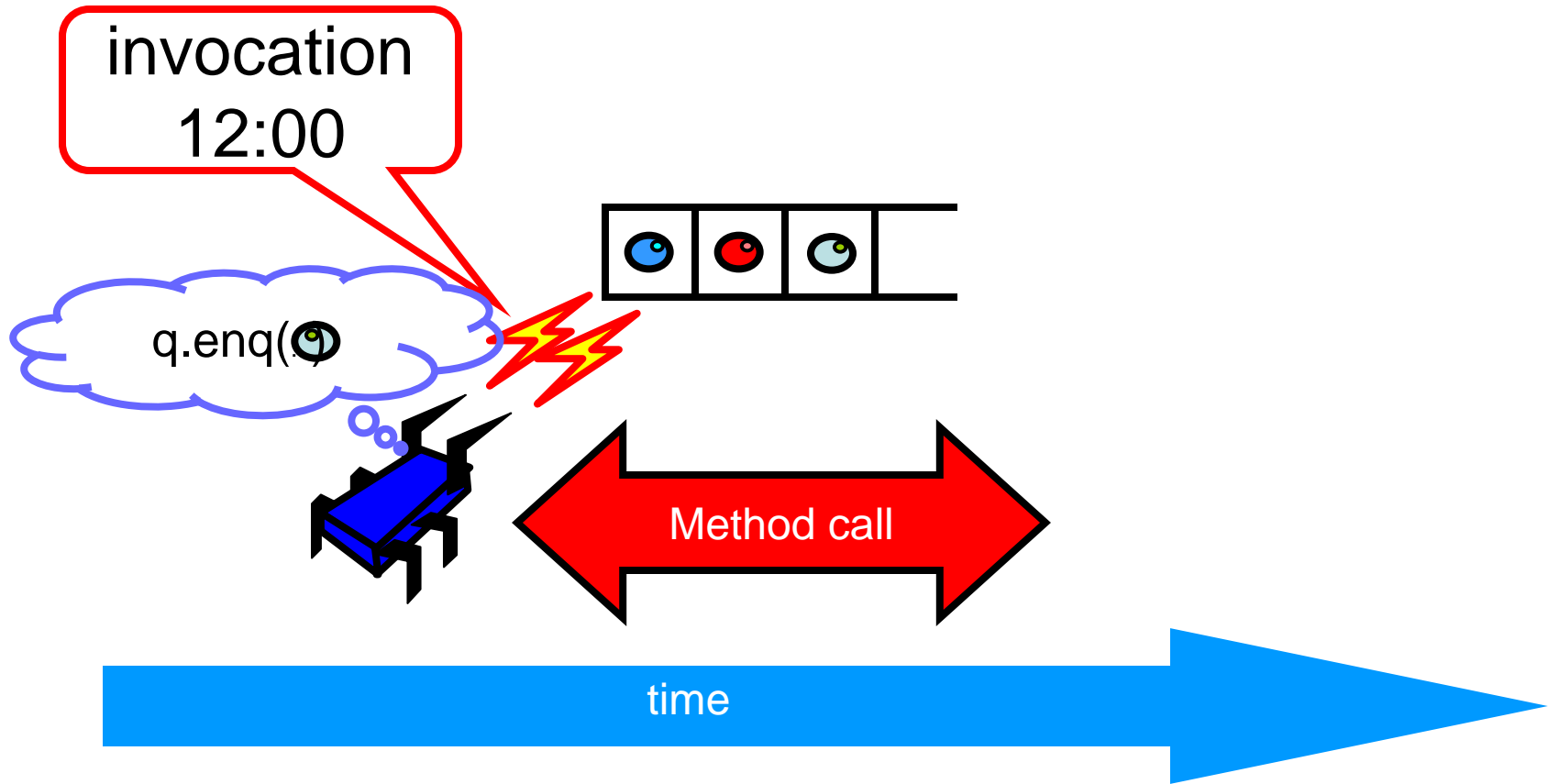
Methods Take Time



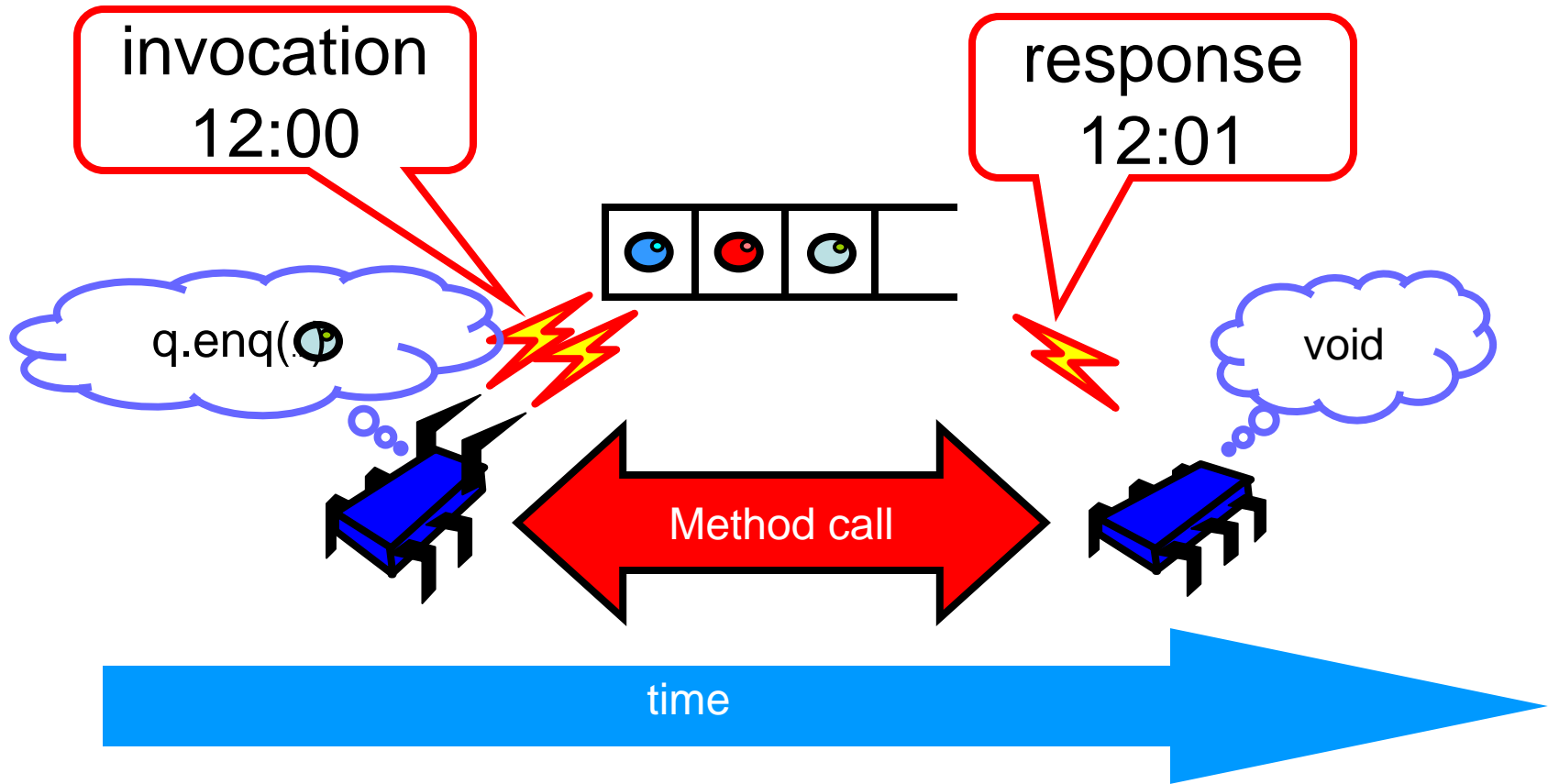
Methods Take Time



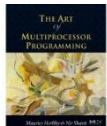
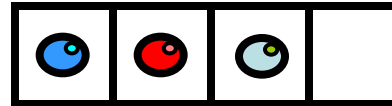
Methods Take Time



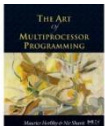
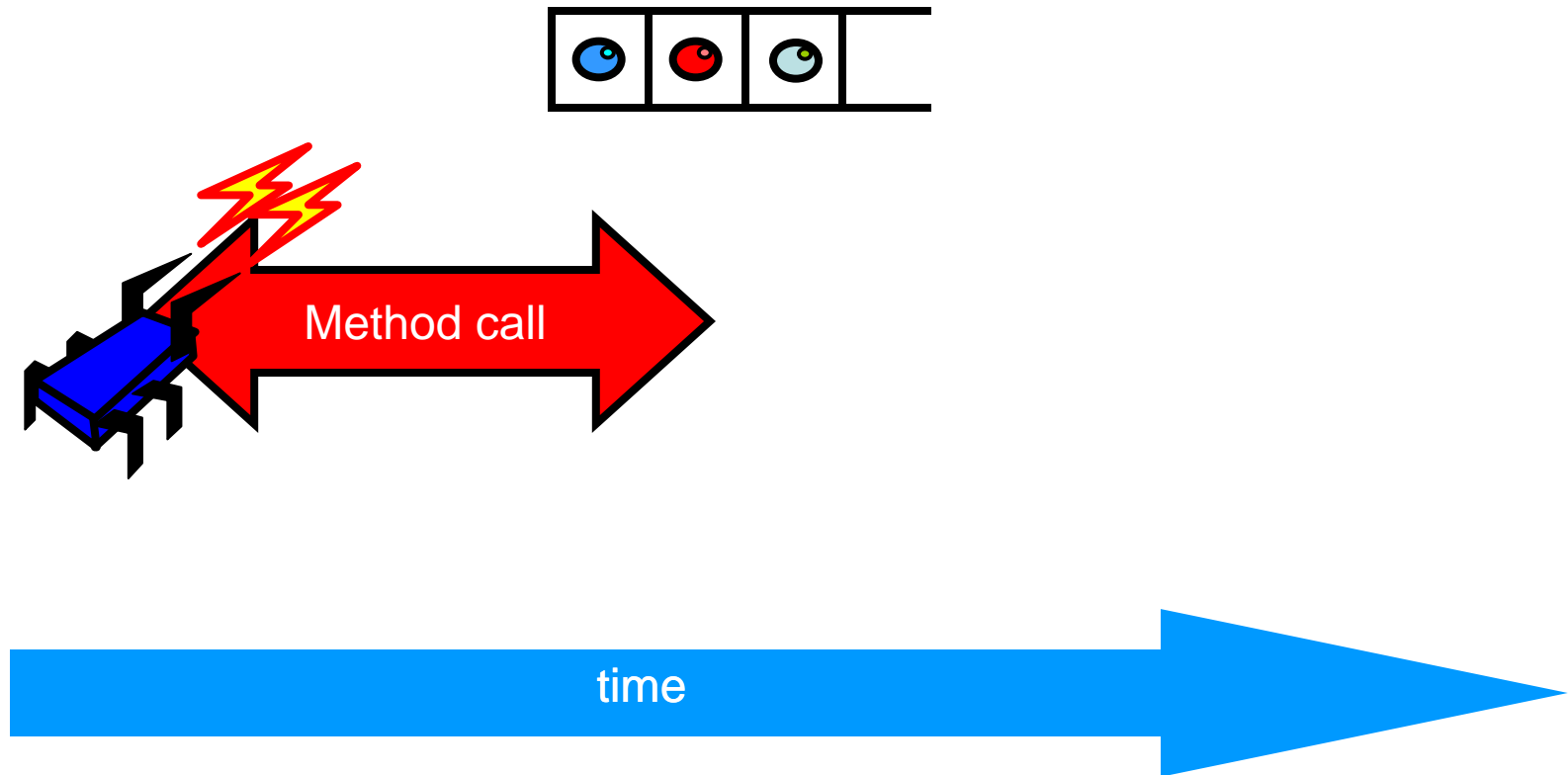
Methods Take Time



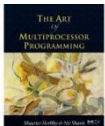
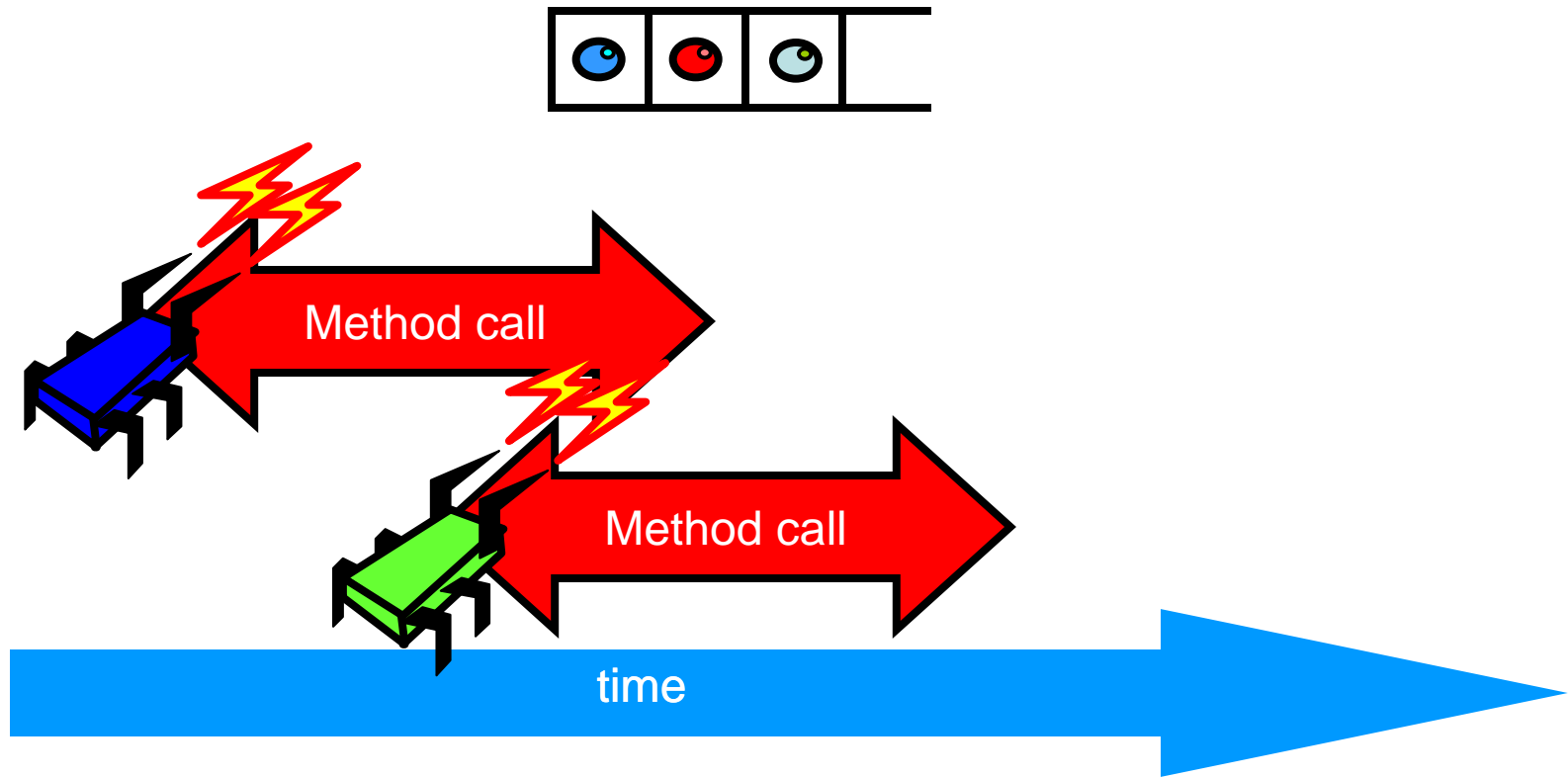
Concurrent Methods Take Overlapping Time



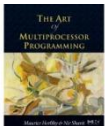
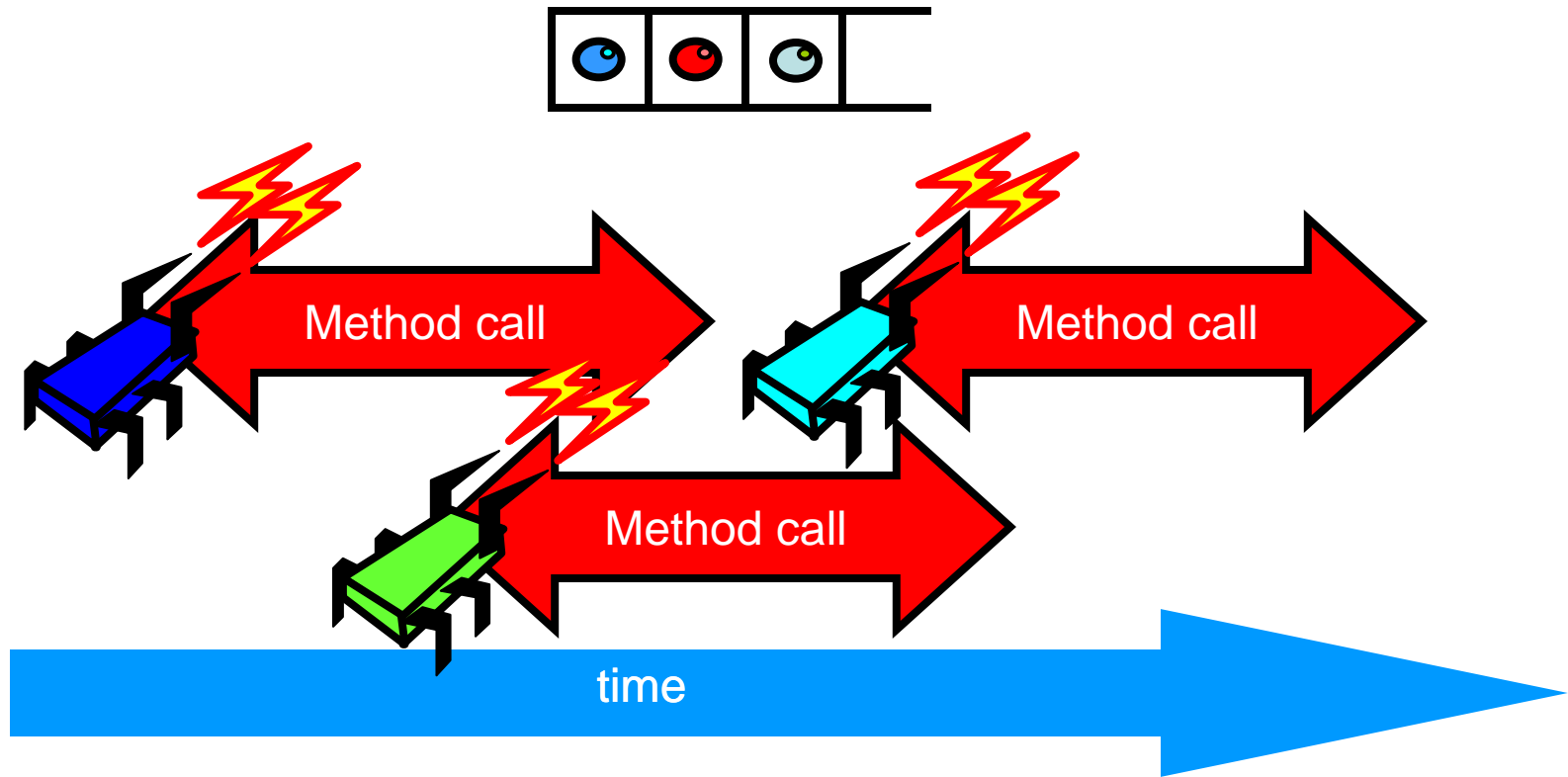
Concurrent Methods Take Overlapping Time



Concurrent Methods Take Overlapping Time

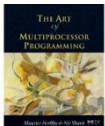


Concurrent Methods Take Overlapping Time

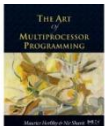
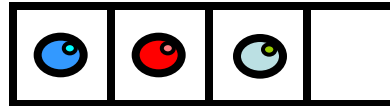


Linearizability

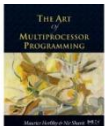
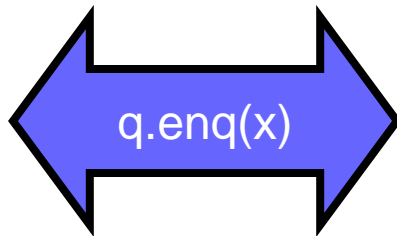
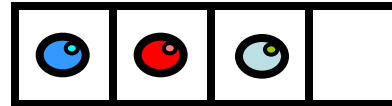
- Each method should
 - “take effect”
 - Instantaneously
 - Between invocation and response events
- Object is correct if this “sequential” behavior is correct
- Any such concurrent object is
 - **Linearizable**TM



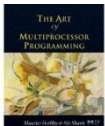
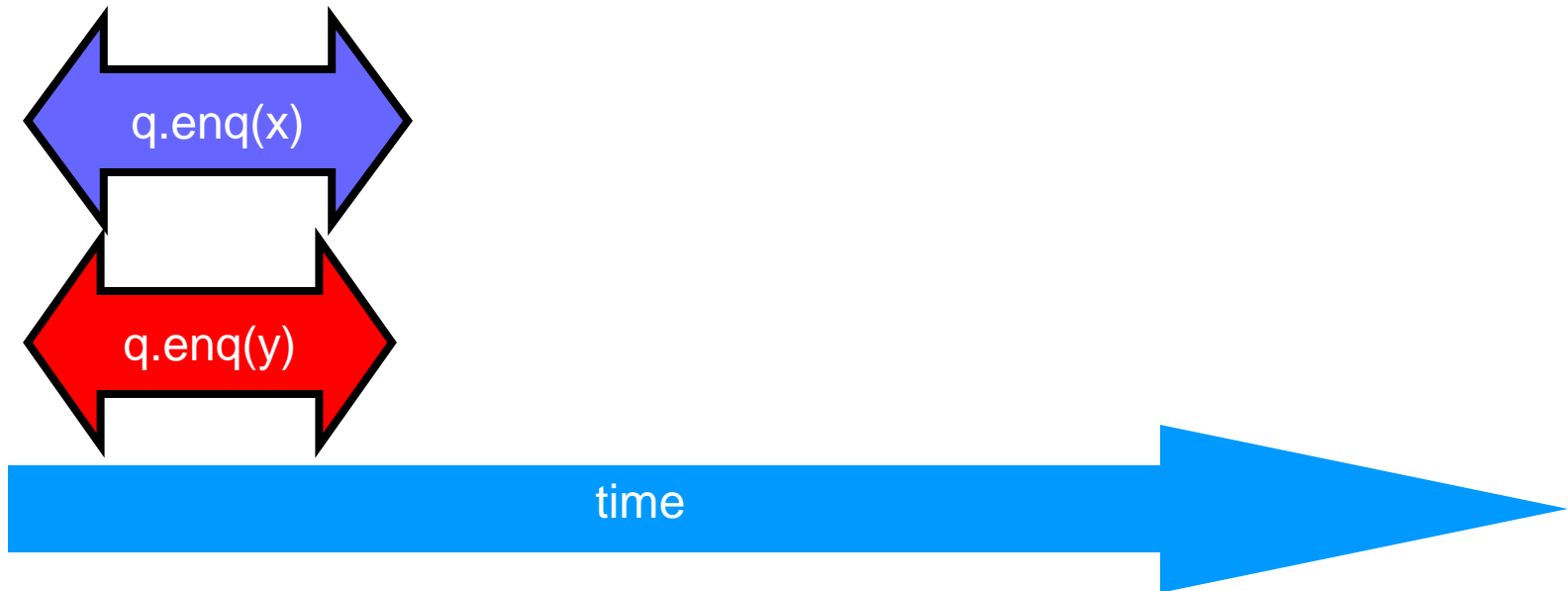
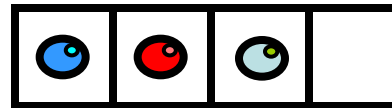
Example



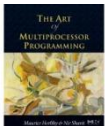
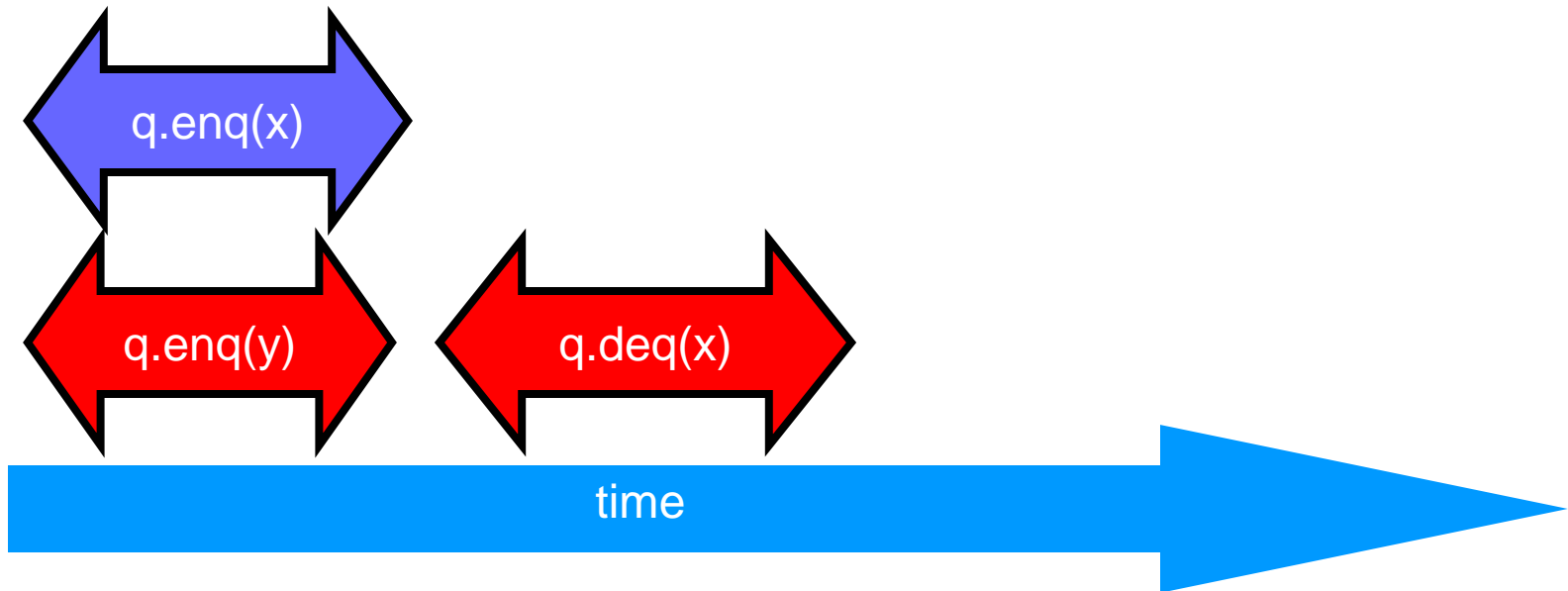
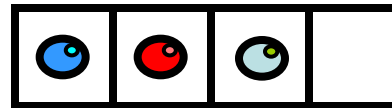
Example



Example

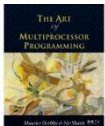
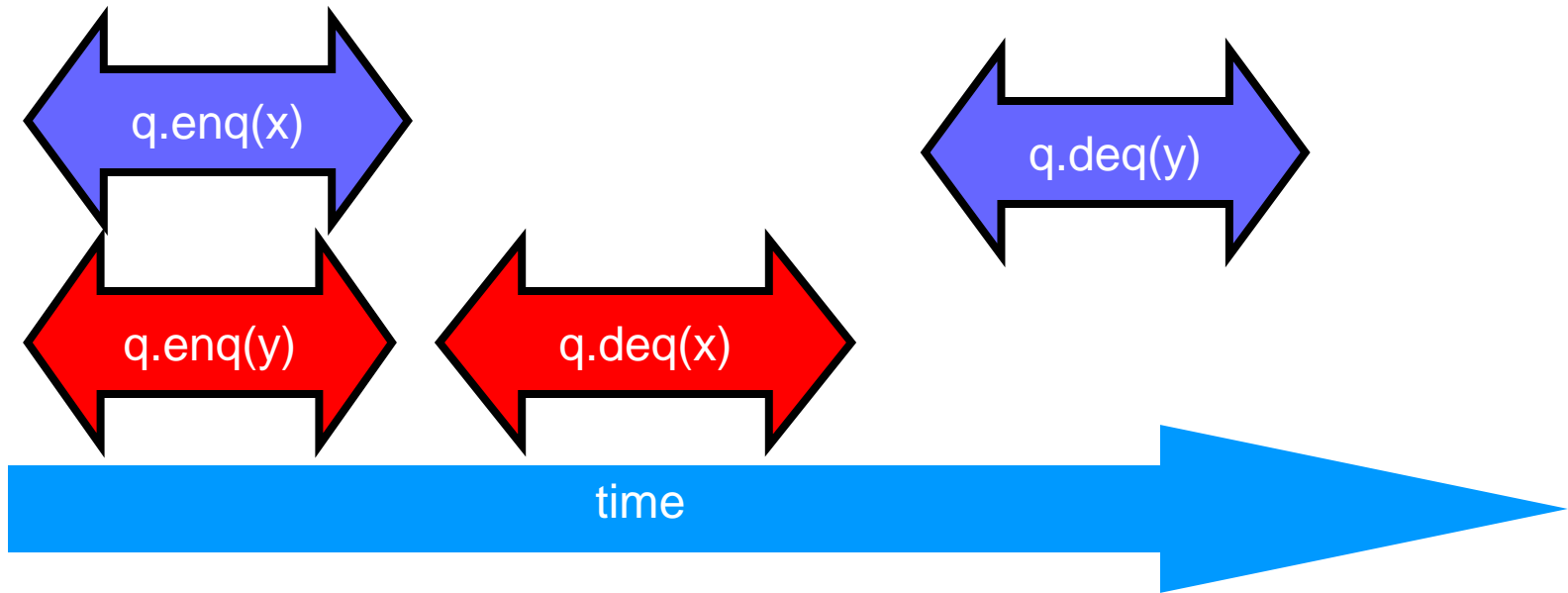
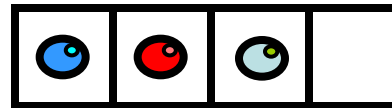


Example

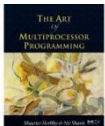
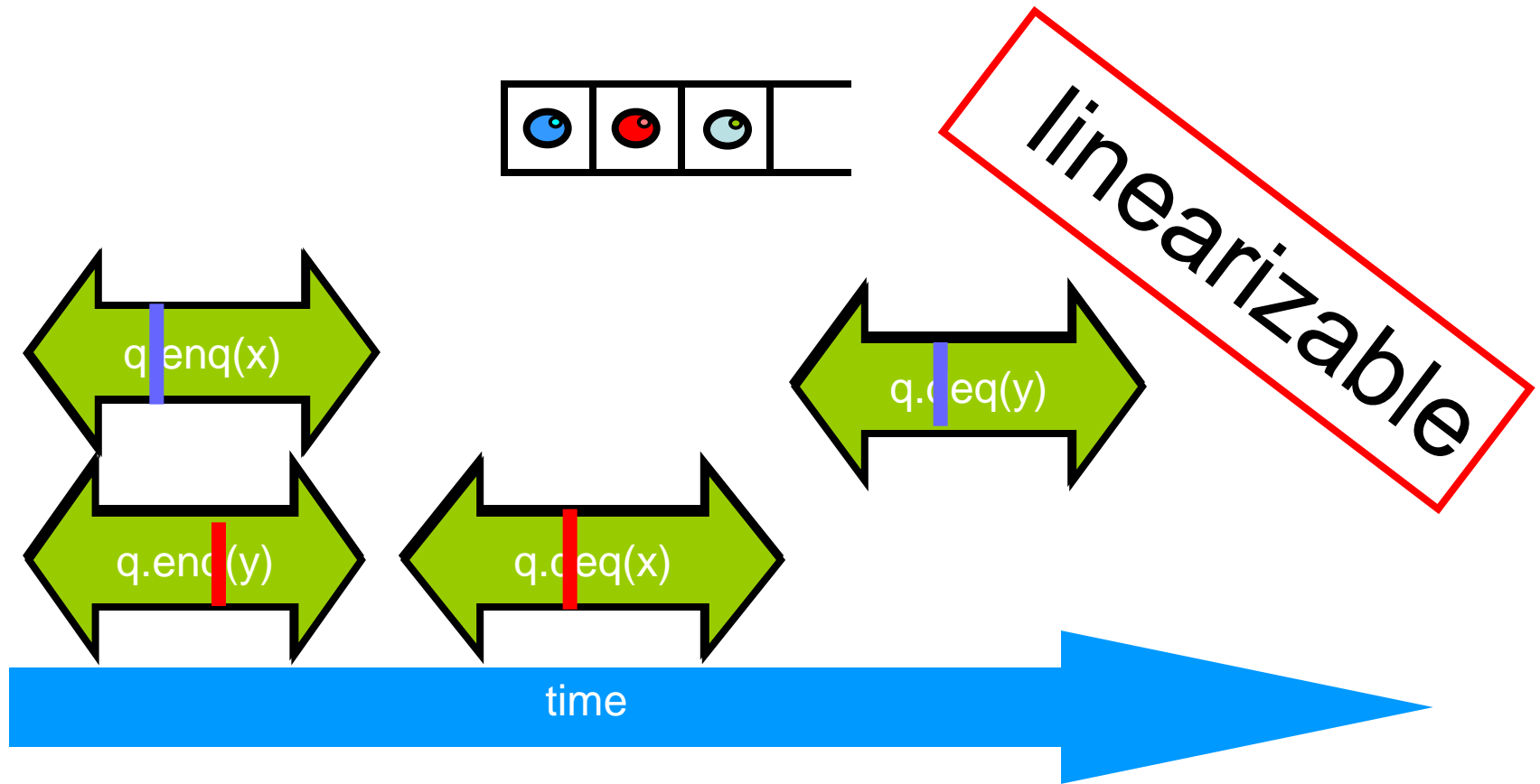




Example



Example



This work is licensed under a [Creative Commons Attribution-ShareAlike 2.5 License](https://creativecommons.org/licenses/by-sa/3.0/).

- **You are free:**
 - **to Share** – to copy, distribute and transmit the work
 - **to Remix** – to adapt the work
- **Under the following conditions:**
 - **Attribution.** You must attribute the work to “The Art of Multiprocessor Programming” (but not in any way that suggests that the authors endorse you or your use of the work).
 - **Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under the same, similar or a compatible license.
- For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to
 - <http://creativecommons.org/licenses/by-sa/3.0/>.
- Any of the above conditions can be waived if you get permission from the copyright holder.
- Nothing in this license impairs or restricts the author's moral rights.

