Lecture 29: Linearizability

Mack Joyner
mjoyner@rice.edu

http://comp322.rice.edu
Linearizability: Correctness of Concurrent Objects

• A *concurrent object* is an *object* that can correctly handle *methods* invoked *concurrently* by different tasks or threads
  —e.g., AtomicInteger, ConcurrentHashMap, ConcurrentLinkedQueue, …

• For the discussion of linearizability, we will assume that the body of each method in a concurrent object is itself sequential
  —Assume that methods do not create threads or async tasks
Linearizability: Correctness of Concurrent Objects

- Consider a simple FIFO (First In, First Out) queue as a canonical example of a concurrent object
  - Method `q.enq(o)` inserts object `o` at the tail of the queue
  - Assume that there is unbounded space available for all `enq()` operations to succeed
  - Method `q.deq()` removes and returns the item at the head of the queue.
  - Throws `EmptyException` if the queue is empty.

- Without seeing the implementation of the FIFO queue, we can tell if an execution of calls to `enq()` and `deq()` is correct or not, in a sequential program

- How can we tell if the execution is correct for a parallel program?
Linearization: Identifying a sequential order of concurrent method calls

"Linearizability" -- identify order of enq() and deq() calls that is consistent with sequential execution

Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt
Informal Definition of Linearizability

- Assume that each method call takes effect “instantaneously” at some point in time between its invocation and return.

- An execution (schedule) is linearizable if we can choose one set of instantaneous points that is consistent with a sequential execution in which methods are executed at those points
  - It’s okay if some other set of instantaneous points is not linearizable

- A concurrent object is linearizable if all its executions are linearizable
  - Linearizability is a “black box” test based on the object’s behavior, not its internals
Example 1

Task T1

\[ q.\text{enq}(x) \]

Source: [http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt](http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt)
Example 1 cont.

Task T1
- q.enq(x)

Task T2
- q.enq(y)

Source: [http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt](http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt)
Example 1 cont.

Task T1
- q.enq(x)

Task T2
- q.enq(y)
- q.deq():x

Source: [http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt](http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt)
Example 1 cont.

Task T1

Task T2

Source: [http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt](http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt)
Example 1: is this execution linearizable?

Task T1

(1) q.enq(x)

Task T2

(2) q.enq(y)

(3) q.deq():x

(4) q.deq():y

Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt
Example 2: is this execution linearizable?

Task T1

Task T2

Source: http://www.elsevierdirect.com/companions/9780123705914/Lecture%20Slides/03~Chapter_03.ppt
Example 3

Is this execution linearizable? How many possible linearizations does it have?

\[\begin{align*}
q.\text{enq}(x) & \\
q.\text{enq}(y) & \\
q.\text{deq}():y & \\
q.\text{deq}():x & \\
\end{align*}\]
Example 4: execution of isolated implementation of FIFO queue q

Is this a linearizable execution?

<table>
<thead>
<tr>
<th>Time</th>
<th>Task A</th>
<th>Task B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Invoke q.enq(x)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Work on q.enq(x)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Work on q.enq(x)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Return from q.enq(x)</td>
<td>Invoke q.enq(y)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Work on q.enq(y)</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Work on q.enq(y)</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>Return from q.enq(y)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Invoke q.deq()</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Return x from q.deq()</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Linearizability of Concurrent Objects (Summary)

Concurrent object
• A concurrent object is an object that can correctly handle methods invoked in parallel by different tasks or threads
  —Examples: Concurrent Queue, AtomicInteger

Linearizability
• Assume that each method call takes effect “instantaneously” at some distinct point in time between its invocation and return.
• An execution is linearizable if we can choose instantaneous points that are consistent with a sequential execution in which methods are executed at those points
• An object is linearizable if all its possible executions are linearizable