

Parallel Programming

Recitation Session 12

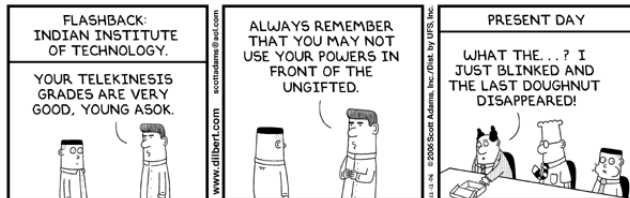
Thomas Weibel <weibelt@ethz.ch>

Laboratory for Software Technology,
Swiss Federal Institute of Technology Zürich

June 3, 2010

Executive Summary

- Linearizability
- Assignment 11
 - Proving program properties
 - Possible executions



© Scott Adams, Inc./Dist. by UFS, Inc.

Outline

- 1 Linearizability**
- 2 Proving Program Properties
- 3 Possible Executions

Definition

- 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**

Is it really about the object?

- Each method should
 - “take effect”
 - Instantaneously
 - Between invocation and response events
- Observation: methods must appear to execute in a one-at-a-time sequential order
- Sounds like a property of an execution
- A linearizable object: one all of whose possible executions are linearizable

Linearizability in Practice



Herlihy and Shavit, *The Art of Multiprocessor Programming*,
Chapter 3

www.elsevierdirect.com/companions/9780123705914



Hendler, et al., *A Dynamic-sized Nonblocking Work Stealing Deque*,

www.springerlink.com/index/Y7HQ174L92170355.pdf



Michael and Scott, *Simple, Fast, and Practical Non-blocking and Blocking Concurrent Queue Algorithms*,

portal.acm.org/citation.cfm?id=248052.248106

Outline

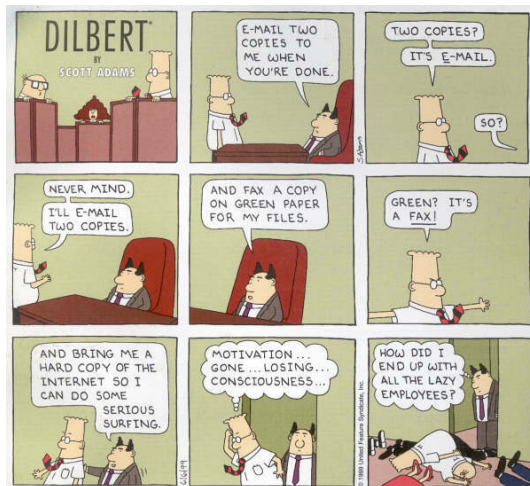
1 Linearizability

2 Proving Program Properties

3 Possible Executions

Variant of Peterson's Solution

- Proving that this variant of Peterson's solution works
- Equivalent to question 2 of the example exam
- See lecture of June 1st, 2010



Outline

1 Linearizability

2 Proving Program Properties

3 Possible Executions

0 1 2

1) T0:
 read f, eval
 print f

2) T1:
 read f
 f++
 store f

3) T0:
 read f, eval
 print f
 read f, eval

4) T1:
 read f
 f++
 store f

5) T0:
 print f

0 0 2

1) T0:

```
read f, eval
print f
read f, eval
print f
```

2) T1:

```
read f
f++
store f
```

3) T0:

```
read f, eval
```

4) T1:

```
read f
f++
store f
```

5) T0:

```
print f
```

0 1

1) T0:
 read f, eval
 print f

2) T1:
 read f
 f++
 store f

3) T0:
 read f, eval
 print f

4) T1:
 read f
 f++
 store f

5) T0:
 read f, eval

The value 2 will not always
appear.

The End

Enjoy your “vacations” and best of luck for the exam!

