How We Test Concurrent Algorithms in Kotlin Coroutines

Nikita Koval

KotlinConf’23
Amsterdam

Speaker: Nikita Koval

- Redesigned synchronization and communication in Kotlin Coroutines
- Regularly publish at top-tier academic conferences
- Teach concurrent programming for over 5 years

@nkoval_

How to develop concurrent algorithms?

BRIAN GOETZ
WITH TIM PEIERLS, JOSHUA BLOCH, JOSEPH BOWBEER, DAVID HOLMES, AND DOUG LEA
JAVA CONCURRENCY IN PRACTICE

REVISED FIRST EDITION
THE ART *of* MULTIPROCESSOR PROGRAMMING
Maurice Herlihy & Nir Shavit
Copyrighted Material

Copyrighted Material
Doug Lea
Concurrent
ogramming in Java™
Second Edition
Design Principles and Patterns
The Java™ Series
... from the Sourc
Copyrighted Material

The Little Book of Semaphores
2nd Edition
The Ins and Outs of Concurrency Control and Common Mistakes
UNDERSTANDING SEMAPHORES AND LEARNING HOW TO APPLY THEM
Allen B. Downey

How to develop concurrent algorithms?

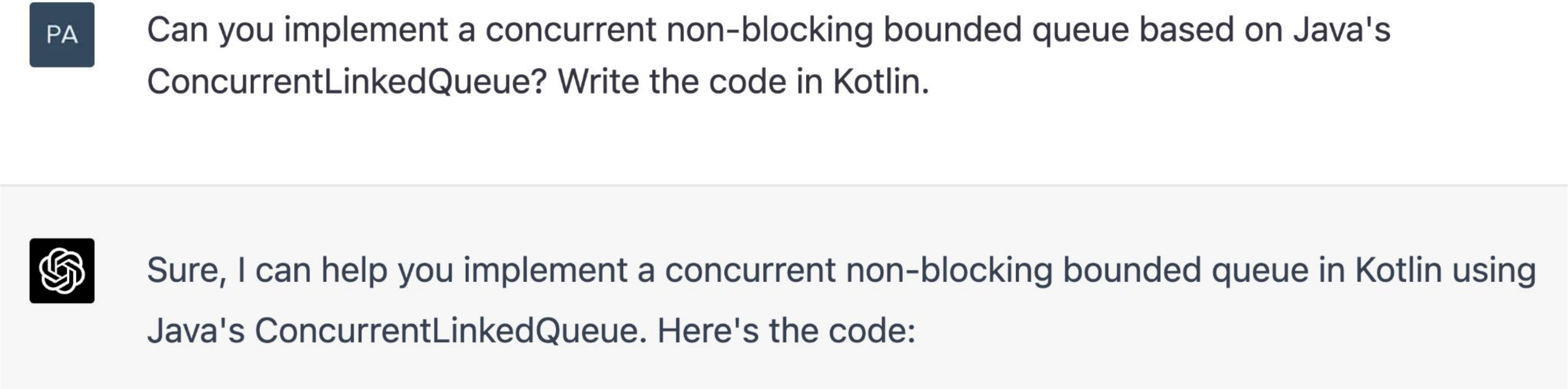
OpenAI

ChatGPT **4.0**

Bounded Queue

```
class BoundedQueue<T>(capacity: Int) {
   // Returns `true` on success and
   // `false` if the queue is full.
   fun add(item: T): Boolean

   // Returns the first element or
   // `null` if the queue is empty.
   fun poll(): T?
}
```

Bounded Queue by Chat-GPT



Can you implement a concurrent non-blocking bounded queue based on Java's ConcurrentLinkedQueue? Write the code in Kotlin.

Sure, I can help you implement a concurrent non-blocking bounded queue in Kotlin using Java's ConcurrentLinkedQueue. Here's the code:

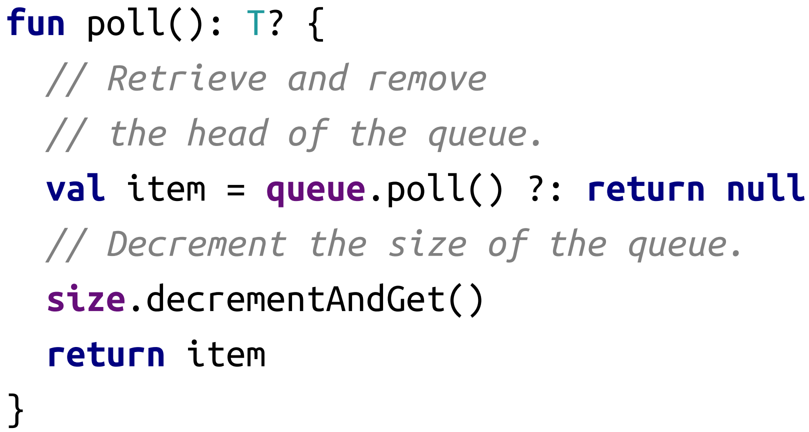
```
class BoundedQueueGPT<T>(private val capacity: Int) {
   private val queue = ConcurrentLinkedQueue<T>()
   private val size = AtomicInteger()
```

```
class BoundedQueueGPT<T>(private val capacity: Int) {
   private val queue = ConcurrentLinkedQueue<T>()
   private val size = AtomicInteger()
```

```
fun add(item: T): Boolean {
  // Check if there is space
  // available in the queue.
  if (size.get() == capacity)
      return false
  // Add the element.
  queue.offer(item)
  // After the element was added
  // to the queue, increment the size.
  size.incrementAndGet()
  return true
}
```

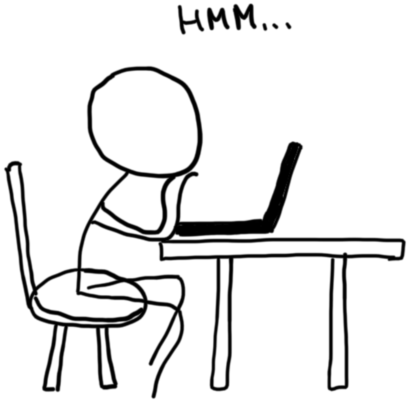
```
class BoundedQueueGPT<T>(private val capacity: Int) {
   private val queue = ConcurrentLinkedQueue<T>()
   private val size = AtomicInteger()
```

```
fun add(item: T): Boolean {
  // Check if there is space
  // available in the queue.
  if (size.get() == capacity)
      return false
  // Add the element.
  queue.offer(item)
  // After the element was added
  // to the queue, increment the size.
  size.incrementAndGet()
  return true
}
```



```
fun poll(): T? {
  // Retrieve and remove
  // the head of the queue.
  val item = queue.poll() ?: return null
  // Decrement the size of the queue.
  size.decrementAndGet()
  return item
}
```

Is `BoundedQueueGPT` correct?



Typical Concurrent Tests

😬 Cover only easy-to-verify scenarios

😤 Contains hundreds of lines of boilerplate code

🤯 Hard to analyze errors

Typical Concurrent Tests

😬 Cover only easy-to-verify scenarios

😤 Contains hundreds of lines of boilerplate code

🤯 Hard to analyze errors

We need a reliable easy-to-use framework
for writing concurrent tests!

An Ideal Concurrent Test

```
class BoundedQueueGPTTest {







}
```

An Ideal Concurrent Test

```
class BoundedQueueGPTTest {
    val q = BoundedQueueGPT<Int>(capacity = 2)



}
```

Initial state

An Ideal Concurrent Test

```
class BoundedQueueGPTTest {
   val q = BoundedQueueGPT<Int>(capacity = 2)

   @Operation fun add(item: Int) = q.add(item)
   @Operation fun poll() = q.poll()


}
```

Operations

An Ideal Concurrent Test

```
class BoundedQueueGPTTest {
   val q = BoundedQueueGPT<Int>(capacity = 2)

   @Operation fun add(item: Int) = q.add(item)
   @Operation fun poll() = q.poll()

   @Test fun test() = runConcurrentTest(this)
}
```

Lincheck

https://github.com/Kotlin/kotlinx-lincheck

Lincheck test for BoundedQueueGPT

```
class BoundedQueueGPTTest {
    val q = BoundedQueueGPT<Int>(capacity = 2)

    @Operation fun add(item: Int) = q.add(item)
    @Operation fun poll() = q.poll()

    @Test fun test() = ModelCheckingOptions()
                       .check(this::class)
}
```

Let’s run the test!

```
= Invalid execution results =
add(2):  true  || add(6): true
add(-8): true  ||

= The following interleaving leads to the error =
                  || add(6)
                  ||   size.get(): 0 at BoundedQueueGPT.kt:10
                  ||   queue.offer(6): true at BoundedQueueGPT.kt:12
                  ||   switch
add(2): true      ||
add(-8): true     ||
                  ||   size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                  ||   result: true
```

What is correctness
in the world of concurrency?

Correctness in Sequential World

Is this execution correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): false
q.poll(): 1
```

Correctness in Sequential World

Is this execution correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): false
q.poll(): 1
```

CORRECT

Correctness in Sequential World

Is this execution correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): false  // capacity is exceeded
q.poll(): 1
```

CORRECT

Correctness in Sequential World

Is this execution correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): false  // capacity is exceeded
q.poll(): 1  // extracts the first element
```

CORRECT

Correctness in Sequential World

Is this one also correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): true
```

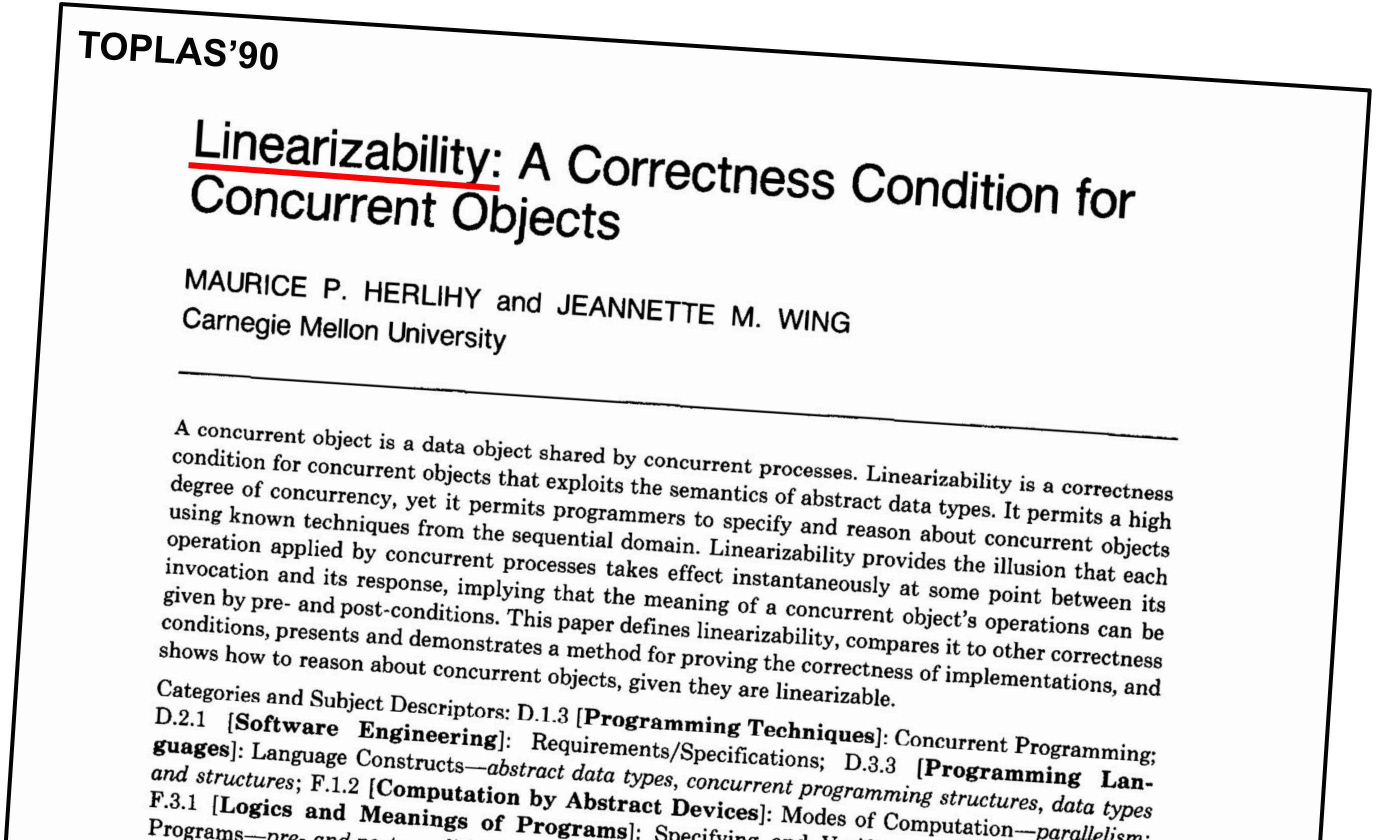
Correctness in Sequential World

Is this one also correct?

```
val q = BoundedQueue<Int>(2)
q.add(1): true
q.add(2): true
q.add(3): true  // capacity is exceeded
```

FORBIDDEN

Correctness in Concurrent World



TOPLAS'90

Linearizability: A Correctness Condition for Concurrent Objects

MAURICE P. HERLIHY and JEANNETTE M. WING
Carnegie Mellon University

A concurrent object is a data object shared by concurrent processes. Linearizability is a correctness condition for concurrent objects that exploits the semantics of abstract data types. It permits a high degree of concurrency, yet it permits programmers to specify and reason about concurrent objects using known techniques from the sequential domain. Linearizability provides the illusion that each operation applied by concurrent processes takes effect instantaneously at some point between its invocation and its response, implying that the meaning of a concurrent object's operations can be given by pre- and post-conditions. This paper defines linearizability, compares it to other correctness conditions, presents and demonstrates a method for proving the correctness of implementations, and shows how to reason about concurrent objects, given they are linearizable.

Categories and Subject Descriptors: D.1.3 [**Programming Techniques**]: Concurrent Programming; D.2.1 [**Software Engineering**]: Requirements/Specifications; D.3.3 [**Programming Languages**]: Language Constructs—*abstract data types, concurrent programming structures, data types and structures*; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*parallelism*; F.3.1 [**Logics and Meanings of Programs**]: Specifying and
Programs—*pre- and*

Correctness in Concurrent World

An execution is *linearizable* if it can be expressed with a sequential execution that:

1. does not violate the original happens-before order
   (program order + synchronization)

2. shows the same results

* the definition is simplified

Correctness in Concurrent World

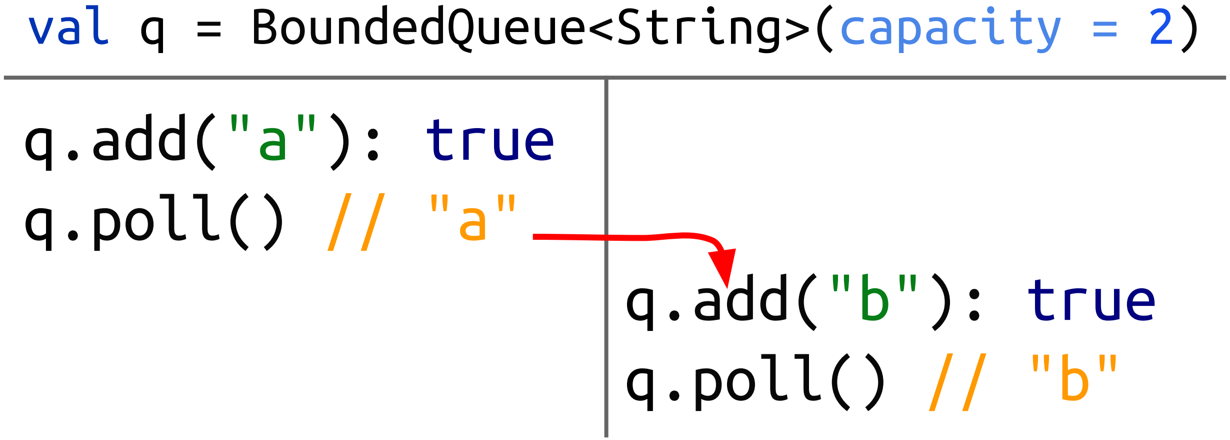
Is this execution correct?

```
val q = BoundedQueue<String>(capacity = 2)
```

`q.add("a"): true`
`q.poll() // "a"`
	`q.add("b"): true`
	`q.poll() // "b"`

Correctness in Concurrent World

Is this execution correct?



```
val q = BoundedQueue<String>(capacity = 2)
q.add("a"): true
q.poll() // "a"
                          q.add("b"): true
                          q.poll() // "b"
```

CORRECT

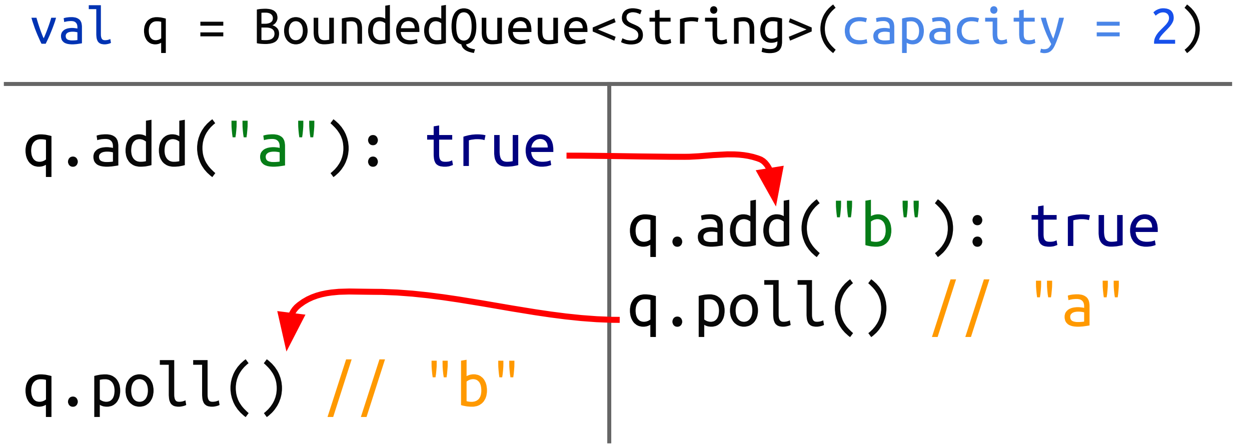
Correctness in Concurrent World

Is this one also correct?

```
val q = BoundedQueue<String>(capacity = 2)

q.add("a"): true     q.add("b"): true
q.poll() // "b"      q.poll() // "a"
```

Correctness in Concurrent World

Is this one also correct?



```
val q = BoundedQueue<String>(capacity = 2)
-------------------------------------------------------
q.add("a"): true      |
                      | q.add("b"): true
                      | q.poll() // "a"
q.poll() // "b"       |
```

CORRECT

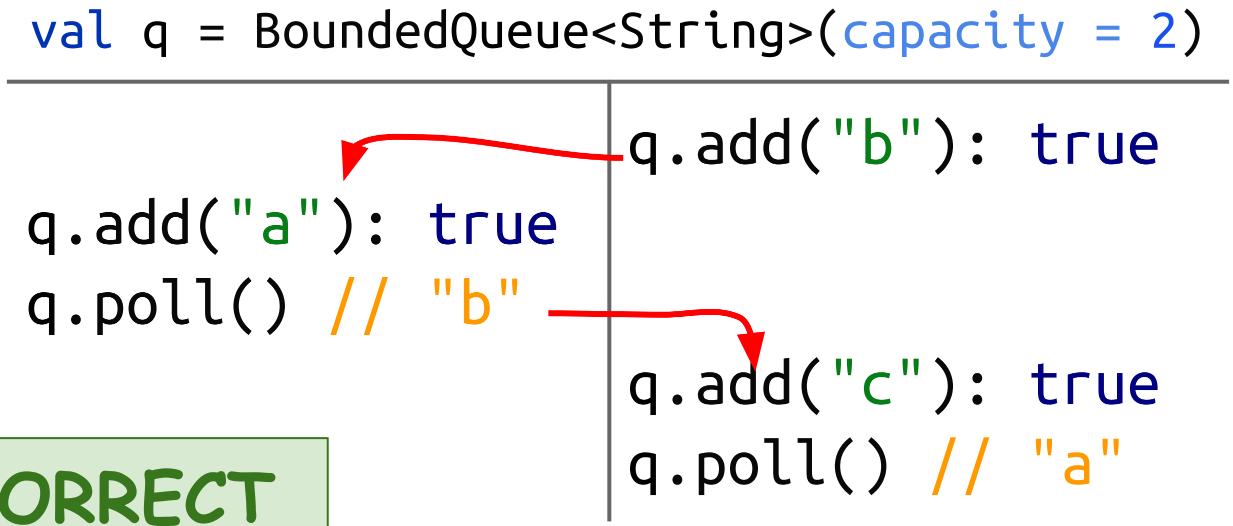
Correctness in Concurrent World

What about this one?

```
val q = BoundedQueue<String>(capacity = 2)
---------------------------------------------
q.add("a"): true     | q.add("b"): true
q.poll() // "b"      | q.add("c"): true
                     | q.poll() // "a"
```

Correctness in Concurrent World

What about this one?



```
val q = BoundedQueue<String>(capacity = 2)
---------------------------------------------------------
                              | q.add("b"): true
q.add("a"): true              |
q.poll() // "b"               |
                              | q.add("c"): true
                              | q.poll() // "a"
```

CORRECT

Correctness in Concurrent World

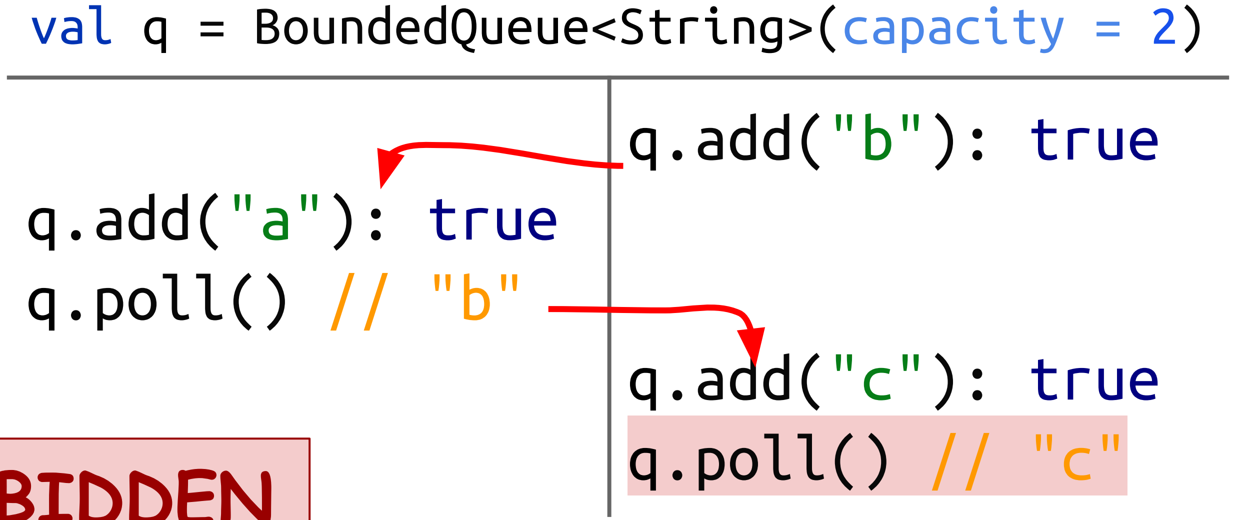
This is also correct, isn’t it?

```
val q = BoundedQueue<String>(capacity = 2)
```

`q.add("a"): true`	`q.add("b"): true`
`q.poll() // "b"`	`q.add("c"): true`
	`q.poll() // "c"`

Correctness in Concurrent World

This is also correct, isn’t it?



```
val q = BoundedQueue<String>(capacity = 2)
```

	`q.add("b"): true`
`q.add("a"): true`	
`q.poll() // "b"`	
	`q.add("c"): true`
	`q.poll() // "c"`

FORBIDDEN

Back to BoundedQueueGPT

```
= Invalid execution results =
add(2):  true  || add(6): true
add(-8): true  ||
```

```
= The following interleaving leads to the error =
                   || add(6)
                   ||   size.get(): 0 at BoundedQueueGPT.kt:10
                   ||   queue.offer(6): true at BoundedQueueGPT.kt:12
                   ||   switch
add(2): true       ||
add(-8): true      ||
                   ||
                   ||   size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                   ||   result: true
```

Exceeds capacity

Back to BoundedQueueGPT

```
= Invalid execution results =
add(2):  true || add(6): true
              ||
add(-8): true ||
```

```
= The following interleaving leads to the error =
                 || add(6)
                 ||   size.get(): 0 at BoundedQueueGPT.kt:10
                 ||   queue.offer(6): true at BoundedQueueGPT.kt:12
                 ||   switch
add(2): true     ||
add(-8): true    ||
                 ||
                 ||   size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                 ||   result: true
```

Back to BoundedQueueGPT

queue =

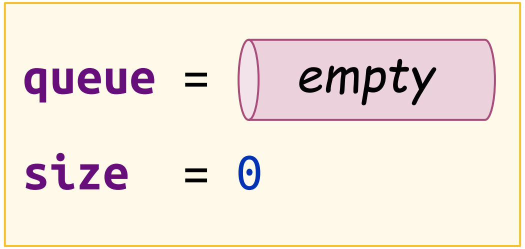
size = 0

```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                  ||  add(6)
                  ||    size.get(): 0 at BoundedQueueGPT.kt:10
                  ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                  ||    switch
add(2): true      ||
add(-8): true     ||
                  ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                  ||    result: true
```

Back to BoundedQueueGPT



queue = *empty*

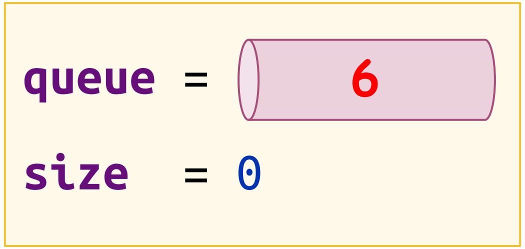
size = 0

```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                  ||  add(6)
                  ||    size.get(): 0 at BoundedQueueGPT.kt:10
                  ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                  ||    switch
add(2): true      ||
add(-8): true     ||
                  ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                  ||    result: true
```

Back to BoundedQueueGPT



```
queue = 6
size  = 0
```

```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                   ||  add(6)
                   ||    size.get(): 0 at BoundedQueueGPT.kt:10
                   ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                   ||    switch
add(2): true       ||
add(-8): true      ||
                   ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                   ||    result: true
```

Back to BoundedQueueGPT

queue = 6, 2

size = ~~0~~ 1

```kotlin
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                   ||  add(6)
                   ||    size.get(): 0 at BoundedQueueGPT.kt:10
                   ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                   ||    switch
add(2): true       ||
add(-8): true      ||
                   ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                   ||    result: true
```

Back to BoundedQueueGPT

queue = 6, 2, -8

size = ~~1~~ 2

```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                  ||  add(6)
                  ||    size.get(): 0 at BoundedQueueGPT.kt:10
                  ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                  ||    switch
add(2): true      ||
add(-8): true     ||
                  ||
                  ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                  ||    result: true
```

Back to BoundedQueueGPT

```
queue = 6, 2, -8
```

size = ~~2~~ 3

```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
               || add(6)
               ||   size.get(): 0 at BoundedQueueGPT.kt:10
               ||   queue.offer(6): true at BoundedQueueGPT.kt:12
               ||   switch
add(2): true   ||
add(-8): true  ||
               ||
               ||   size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
               ||   result: true
```

Back to BoundedQueueGPT

```
queue = 6, 2, -8

size  = 3
```

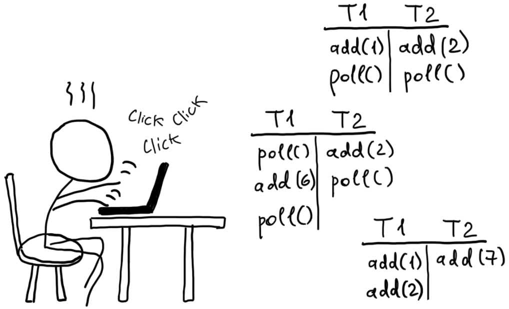
```
fun add(item: T): Boolean {
  if (size.get() == capacity)
    return false
  queue.offer(item)
  size.incrementAndGet()
  return true
}
```

= The following interleaving leads to the error =

```
                   ||  add(6)
                   ||    size.get(): 0 at BoundedQueueGPT.kt:10
                   ||    queue.offer(6): true at BoundedQueueGPT.kt:12
                   ||    switch
add(2): true       ||
add(-8): true      ||
                   ||
                   ||    size.incrementAndGet(): 3 at BoundedQueueGPT.kt:14
                   ||    result: true
```

Lincheck Overview

I. Generates concurrent scenarios (à la fuzzer)



T1	T2
add(1)	add(2)
poll()	poll()

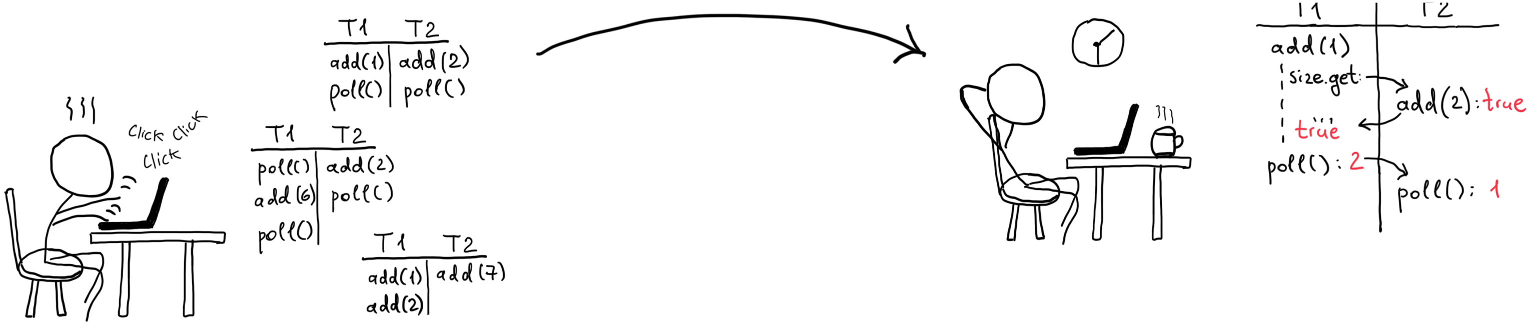
T1	T2
poll()	add(2)
add(6)	poll()
poll()	

T1	T2
add(1)	add(7)
add(2)	

Lincheck Overview

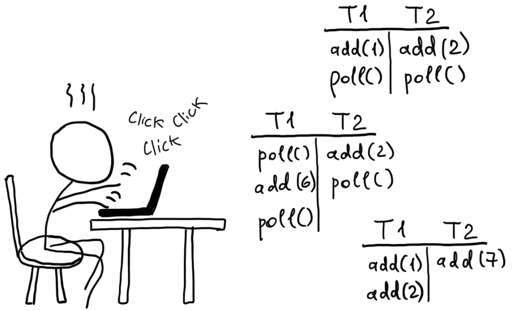
I.Generates concurrent scenarios (à la fuzzer)

II. Examines them either via stress testing or model checking



Lincheck Overview

I. Generates concurrent scenarios (à la fuzzer)

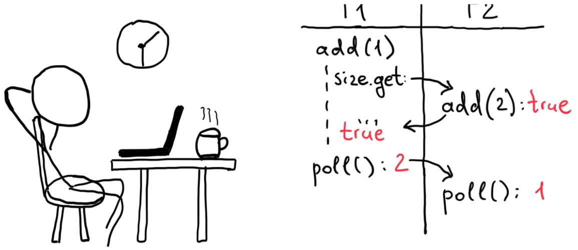


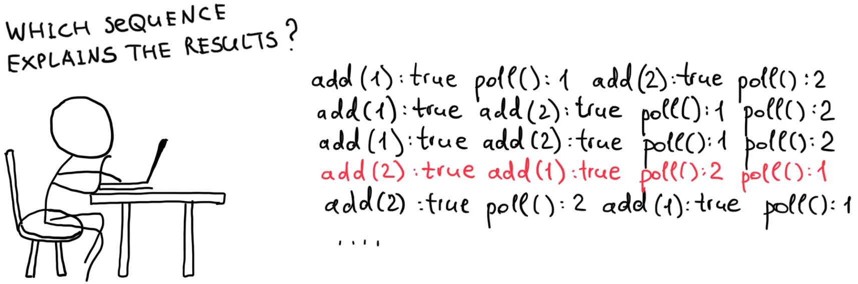
T1	T2
add(1)	add(2)
poll()	poll()

T1	T2
poll()	add(2)
add(6)	poll()
poll()	

T1	T2
add(1)	add(7)
add(2)	

II. Examines them either via stress testing or model checking





WHICH SEQUENCE EXPLAINS THE RESULTS?

add(1): true poll(): 1 add(2): true poll(): 2
add(1): true add(2): true poll(): 1 poll(): 2
add(1): true add(2): true poll(): 1 poll(): 2
add(2): true add(1): true poll(): 2 poll(): 1
add(2): true poll(): 2 add(1): true poll(): 1
....

III. Automatically verifies the outcome results

Bytecode Transformation

```
class Counter {
   @Volatile var value = 0

   fun incAndGet(): Int {
       value += 1
       return value
   }
}
```

Bytecode Transformation

```
class Counter {
    @Volatile var value = 0

    fun incAndGet(): Int {
        value += 1
        return value
    }
}
```

```
public incAndGet()I()
    ALOAD 0
    GETFIELD Counter.value : I // read
    ISTORE 1
    ALOAD 0
    ILOAD 1
    ICONST 1
    IADD
    PUTFIELD Counter.value : I // write
    ALOAD 0
    GETFIELD Counter.value : I // read
    IRETURN
```

Bytecode Transformation

```
class Counter {
   @Volatile var value = 0

   fun incAndGet(): Int {
       value += 1
       return value
   }
}
```

```
public incAndGet()I()
   ALOAD 0
   INVOKESTATIC Lincheck.beforeRead:()V
   GETFIELD Counter.value : I // read
   ISTORE 1
   ALOAD 0
   ILOAD 1
   ICONST 1
   IADD
   INVOKESTATIC Lincheck.beforeWrite:()V
   PUTFIELD Counter.value : I // write
   ALOAD 0
   INVOKESTATIC Lincheck.beforeRead:()V
   GETFIELD Counter.value : I // read
   IRETURN
```

Counter Test

```kotlin
class Counter {
    @Volatile var value = 0

    fun incAndGet(): Int {
        value += 1
        return value
    }
}
```

```kotlin
class CounterTest {
    private val c = Counter()

    @Operation
    fun incAndGet() = c.incAndGet()

    @Test
    fun test() = ModelCheckingOptions()
                 .check(this::class)
}
```

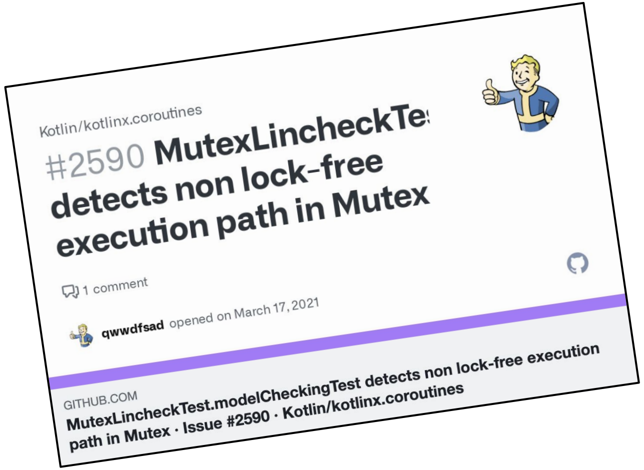
Counter Test

```
= Invalid execution results =
incAndGet(): 1 || incAndGet(): 1

= The following interleaving leads to the error =
                 || incAndGet()
                 ||    value.READ: 0 at Counter.incAndGet(Counter.kt:7)
                 ||    switch
 incAndGet(): 1  ||
                 ||    value.WRITE(1) at Counter.incAndGet(Counter.kt:7)
                 ||    value.READ: 1 at Counter.incAndGet(Counter.kt:8)
                 || result: 1
```

Counter Test

```
= Invalid execution results =
incAndGet(): 1 || incAndGet(): 1

= The following interleaving leads to the error =
                  ||  incAndGet()
                  ||     Lincheck.beforeRead(): CONTINUE
                  ||     value.READ: 0 at Counter.incAndGet(Counter.kt:7)
                  ||     Lincheck.beforeWrite(): SWITCH
                  ||     switch
 incAndGet(): 1   ||
                  ||     value.WRITE(1) at Counter.incAndGet(Counter.kt:7)
                  ||     Lincheck.beforeRead(): CONTINUE
                  ||     value.READ: 1 at Counter.incAndGet(Counter.kt:8)
                  ||  result: 1
```

How do we use Lincheck in Kotlin Coroutines?



Kotlin/kotlinx.coroutines

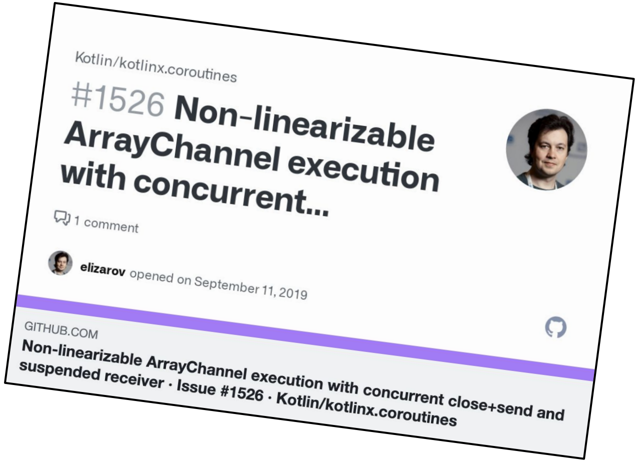
#2590 MutexLincheckTe
detects non lock-free
execution path in Mutex

1 comment

qwwdfsad opened on March 17, 2021

GITHUB.COM

MutexLincheckTest.modelCheckingTest detects non lock-free execution path in Mutex · Issue #2590 · Kotlin/kotlinx.coroutines



Kotlin/kotlinx.coroutines

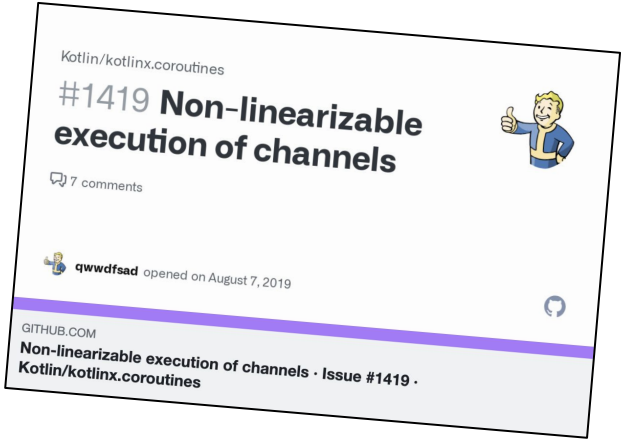
#1526 Non-linearizable
ArrayChannel execution
with concurrent...

1 comment

elizarov opened on September 11, 2019

GITHUB.COM

Non-linearizable ArrayChannel execution with concurrent close+send and suspended receiver · Issue #1526 · Kotlin/kotlinx.coroutines



Kotlin/kotlinx.coroutines

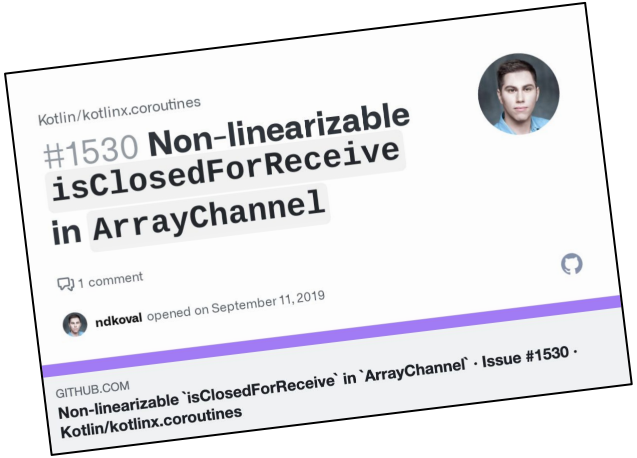
#1419 Non-linearizable
execution of channels

7 comments

qwwdfsad opened on August 7, 2019

GITHUB.COM

Non-linearizable execution of channels · Issue #1419 · Kotlin/kotlinx.coroutines



Kotlin/kotlinx.coroutines

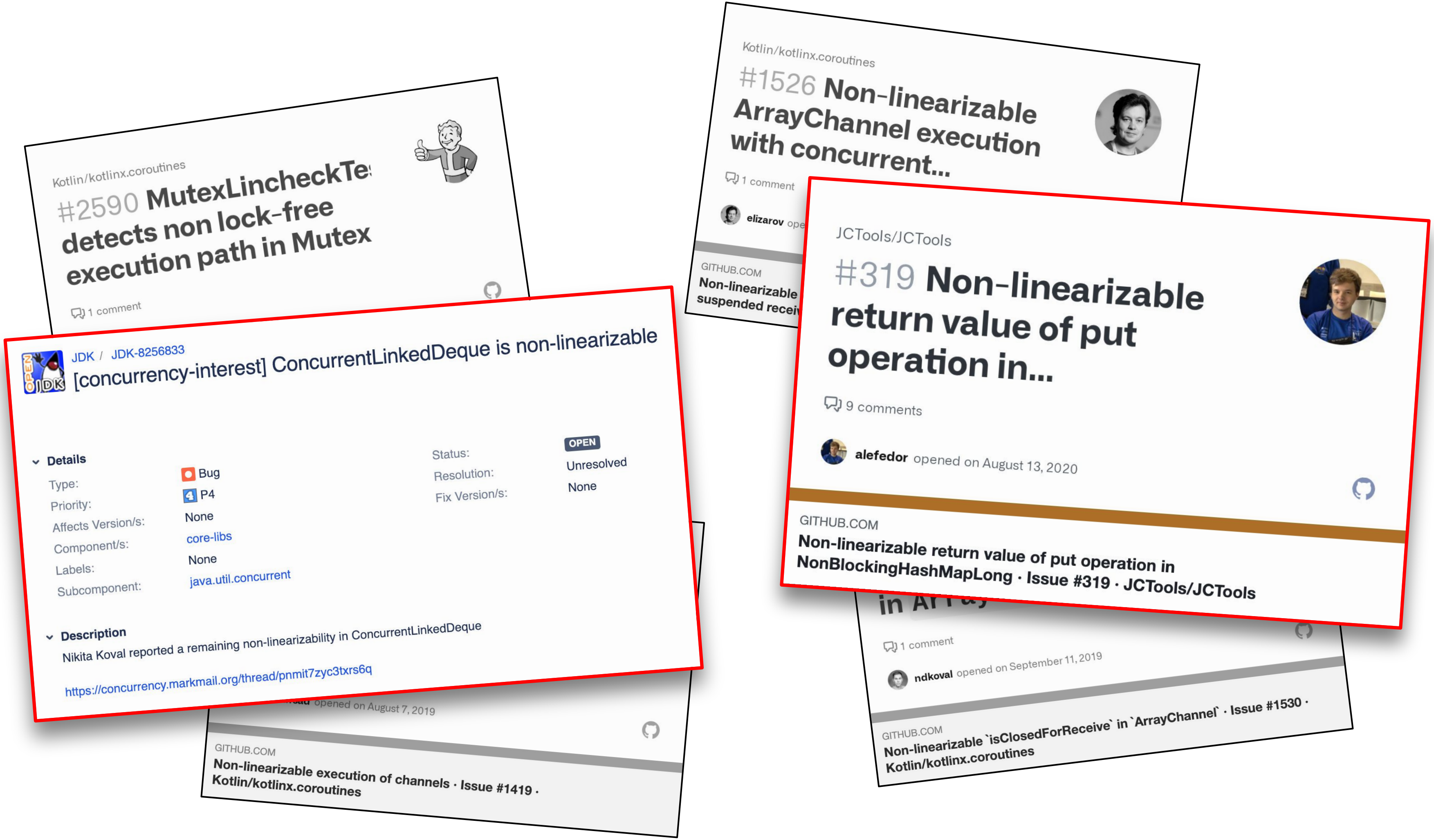
#1530 Non-linearizable `isClosedForReceive` in `ArrayChannel`

1 comment

ndkoval opened on September 11, 2019

GITHUB.COM

Non-linearizable `isClosedForReceive` in `ArrayChannel` · Issue #1530 · Kotlin/kotlinx.coroutines



Kotlin/kotlinx.coroutines

#2590 MutexLincheckTe
detects non lock-free execution path in Mutex

1 comment

JDK / JDK-8256833

[concurrency-interest] ConcurrentLinkedDeque is non-linearizable

Details

Type:	Bug
Priority:	P4
Affects Version/s:	None
Component/s:	core-libs
Labels:	None
Subcomponent:	java.util.concurrent
Status:	OPEN
Resolution:	Unresolved
Fix Version/s:	None

Description

Nikita Koval reported a remaining non-linearizability in ConcurrentLinkedDeque

https://concurrency.markmail.org/thread/pnmit7zyc3txrs6q

---

opened on August 7, 2019

GITHUB.COM

Non-linearizable execution of channels · Issue #1419 · Kotlin/kotlinx.coroutines

---

Kotlin/kotlinx.coroutines

#1526 Non-linearizable ArrayChannel execution with concurrent...

1 comment

elizarov ope

GITHUB.COM

Non-linearizable
suspended recei

JCTools/JCTools

#319 Non-linearizable return value of put operation in...

9 comments

alefedor opened on August 13, 2020

GITHUB.COM

Non-linearizable return value of put operation in NonBlockingHashMapLong · Issue #319 · JCTools/JCTools

---

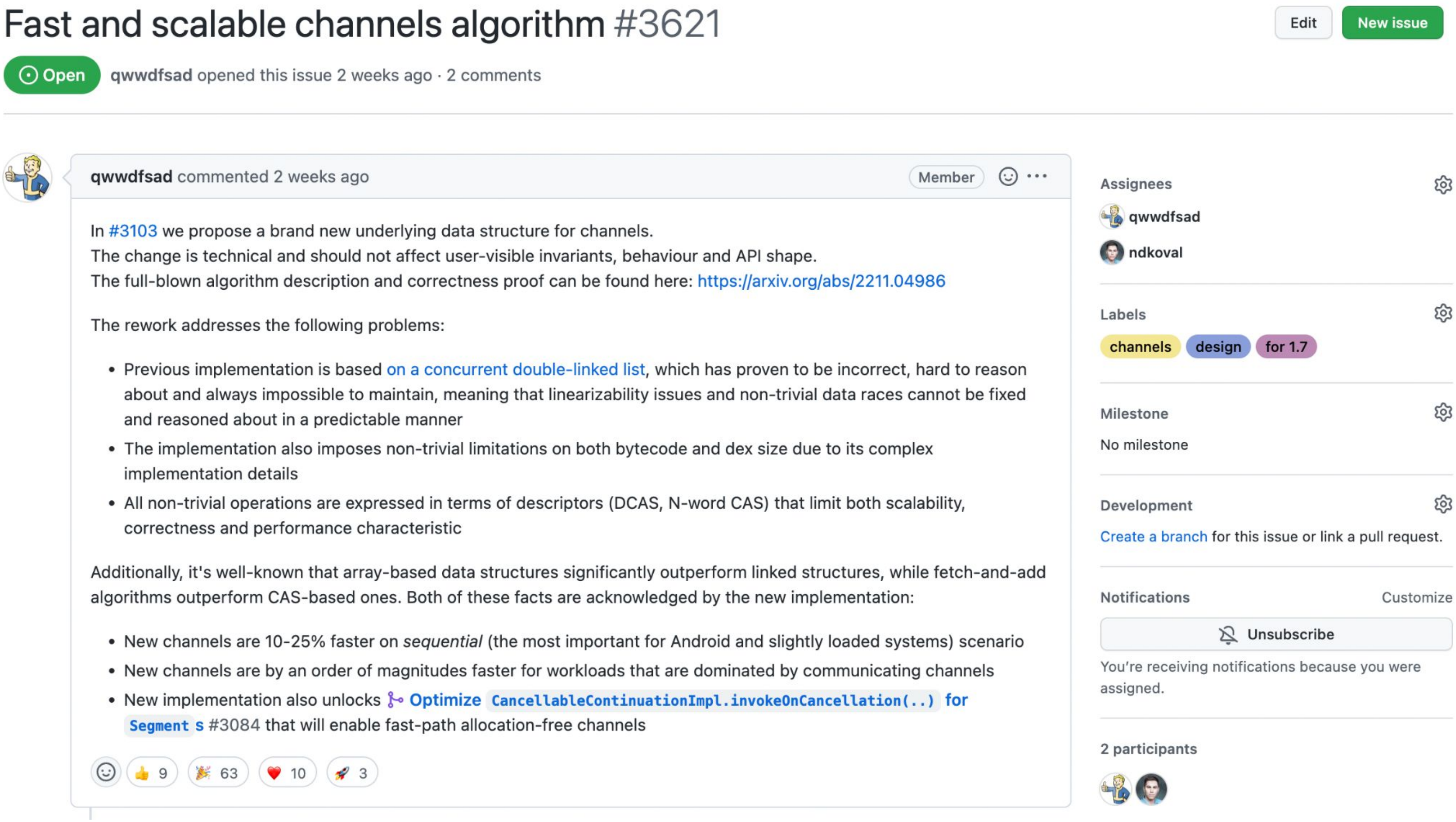
in Array

1 comment

ndkoval opened on September 11, 2019

GITHUB.COM

Non-linearizable `isClosedForReceive` in `ArrayChannel` · Issue #1530 · Kotlin/kotlinx.coroutines



Fast and scalable channels algorithm #3621

qwwdfsad opened this issue 2 weeks ago · 2 comments

qwwdfsad commented 2 weeks ago

In #3103 we propose a brand new underlying data structure for channels.
The change is technical and should not affect user-visible invariants, behaviour and API shape.
The full-blown algorithm description and correctness proof can be found here: https://arxiv.org/abs/2211.04986

The rework addresses the following problems:

- Previous implementation is based on a concurrent double-linked list, which has proven to be incorrect, hard to reason about and always impossible to maintain, meaning that linearizability issues and non-trivial data races cannot be fixed and reasoned about in a predictable manner
- The implementation also imposes non-trivial limitations on both bytecode and dex size due to its complex implementation details
- All non-trivial operations are expressed in terms of descriptors (DCAS, N-word CAS) that limit both scalability, correctness and performance characteristic

Additionally, it's well-known that array-based data structures significantly outperform linked structures, while fetch-and-add algorithms outperform CAS-based ones. Both of these facts are acknowledged by the new implementation:

- New channels are 10-25% faster on *sequential* (the most important for Android and slightly loaded systems) scenario
- New channels are by an order of magnitudes faster for workloads that are dominated by communicating channels
- New implementation also unlocks Optimize `CancellableContinuationImpl.invokeOnCancellation(..)` for `Segment` s #3084 that will enable fast-path allocation-free channels

Assignees

qwwdfsad
ndkoval

Labels

channels design for 1.7

Milestone

No milestone

Development

Create a branch for this issue or link a pull request.

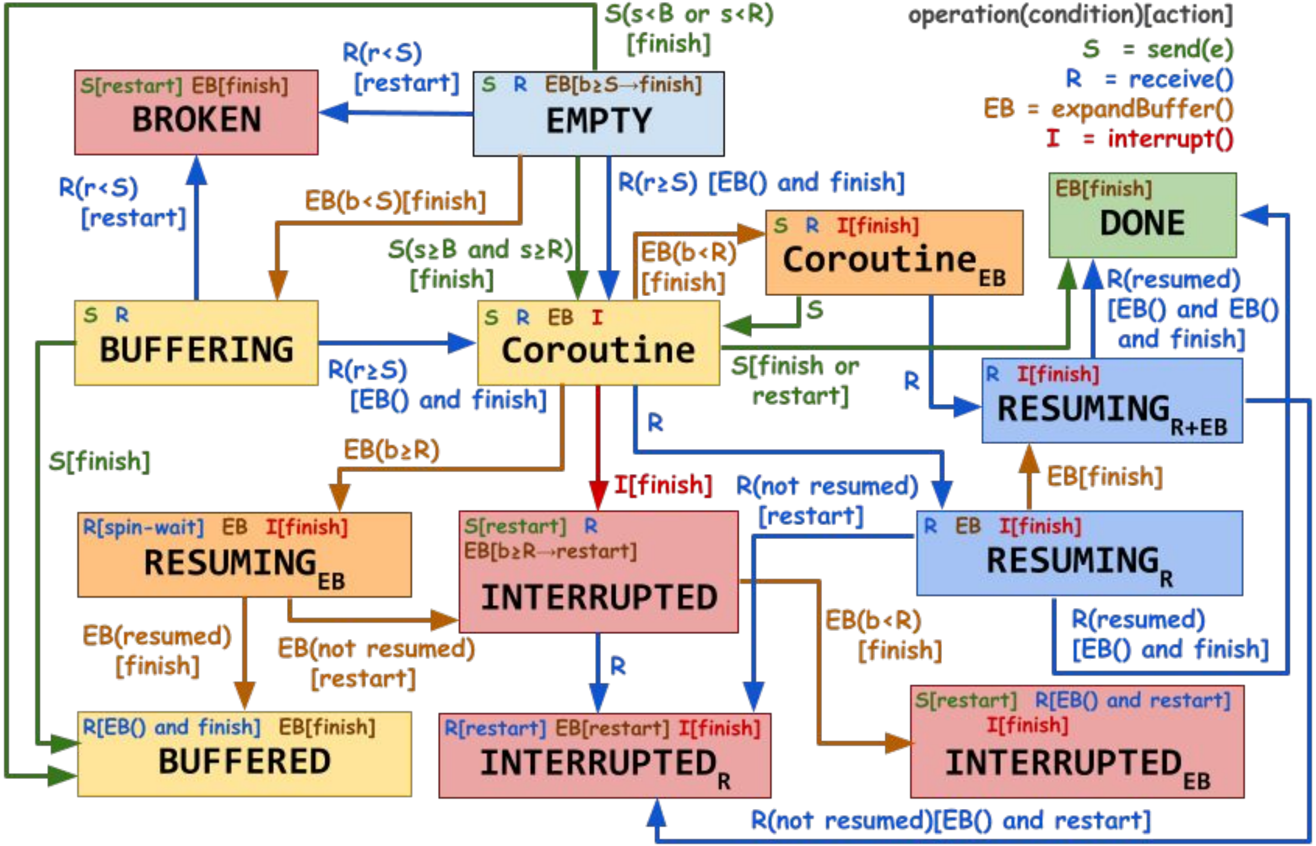
Notifications

You're receiving notifications because you were assigned.

2 participants

https://github.com/Kotlin/kotlinx.coroutines/issues/3621

Impossible to implement without Lincheck



Summary

- Never use ChatGPT to implement concurrent algorithms
- Always use Lincheck to test them
- Stay tuned for updates 🙂

Thank you, and don’t forget to vote

KotlinConf’23
Amsterdam

Scenario Generation: Single Consumer/Producer

```kotlin
import org.jctools.queues.atomic.*

class MPSCQueueTest {
   private val queue = MpscLinkedAtomicQueue<Int>()

   @Operation
   public fun offer(element: Int) = queue.offer(element)

   @Operation
   public fun poll(): Int? = queue.poll()

   @Operation
   public fun peek(): Int? = queue.peek()

   @Test
   fun test() = LincheckOptions().check(this::class)
}
```

Scenario Generation: Single Consumer/Producer

```
import org.jctools.queues.atomic.*

class MPSCQueueTest {
   private val queue = MpscLinkedAtomicQueue<Int>()

   @Operation
   public fun offer(element: Int) = queue.offer(element)

   @Operation
   public fun poll(): Int? = queue.poll()

   @Operation
   public fun peek(): Int? = queue.peek()

   @Test
   fun test() = LincheckOptions().check(this::class)
}
```

poll()	offer(3)
peak()	**peak()**
offer(1)	offer(5)
offer(-7)	**poll()**
peak()	offer(10)

Scenario Generation: Single Consumer/Producer

```kotlin
import org.jctools.queues.atomic.*

class MPSCQueueTest {
    private val queue = MpscLinkedAtomicQueue<Int>()

    @Operation
    public fun offer(element: Int) = queue.offer(element)

    @Operation(nonParallelGroup = "consumers")
    public fun poll(): Int? = queue.poll()

    @Operation(nonParallelGroup = "consumers")
    public fun peek(): Int? = queue.peek()

    @Test
    fun test() = LincheckOptions().check(this::class)
}
```

Scenario Generation: Single Consumer/Producer

```
import org.jctools.queues.atomic.*

class MPSCQueueTest {
   private val queue = MpscLinkedAtomicQueue<Int>()

   @Operation
   public fun offer(element: Int) = queue.offer(element)

   @Operation(nonParallelGroup = "consumers")
   public fun poll(): Int? = queue.poll()

   @Operation(nonParallelGroup = "consumers")
   public fun peek(): Int? = queue.peek()

   @Test
   fun test() = LincheckOptions().check(this::class)
}
```

poll()	offer(3)
poll()	offer(-4)
peak()	offer(5)
offer(-7)	offer(8)
peak()	offer(10)

Run ConcurrentLinkedDequeTest

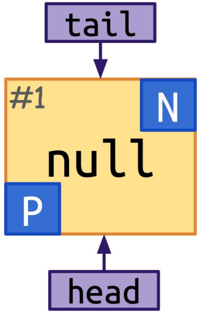
Invalid execution results:

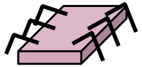
```
addLast(-6)      || addFirst(-8)
peekFirst(): -8  || pollLast(): -8
```

The following interleaving leads to the error:

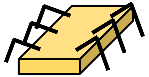
```
                 || addFirst(-8)
                 || pollLast(): -8
                 ||  pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
                 ||   last(): Node@1 at CLD.pollLast(CLD.java:936)
                 ||   item.READ: null at CLD.pollLast(CLD.java:938)
                 ||   prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
                 ||   item.READ: -8 at CLD.pollLast(CLD.java:938)
                 ||   next.READ: null at CLD.pollLast(CLD.java:940)
addLast(-6)      ||
peekFirst(): -8  ||
                 ||   item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
                 ||   unlink(Node@2) at CLD.pollLast(CLD.java:942)
                 ||  result: -8
```

Bug in ConcurrentLinkedDeque



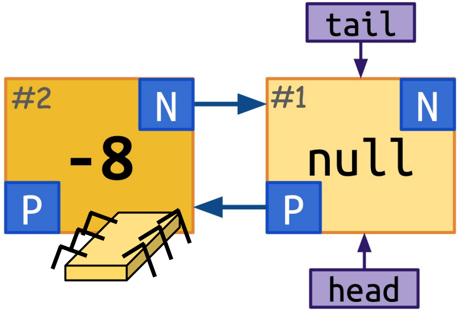


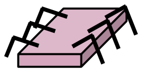
```
addLast(-6)
peekFirst(): -8
```



```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

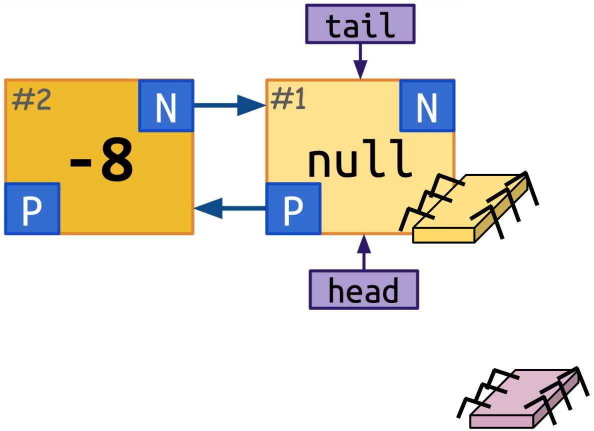
Bug in ConcurrentLinkedDeque





```
                   addFirst(-8)
                   pollLast(): -8
                    pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
                     last(): Node@1 at CLD.pollLast(CLD.java:936)
                     item.READ: null at CLD.pollLast(CLD.java:938)
                     prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
                     item.READ: -8 at CLD.pollLast(CLD.java:938)
                     next.READ: null at CLD.pollLast(CLD.java:940)
addLast(-6)
peekFirst(): -8

                     item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
                     unlink(Node@2) at CLD.pollLast(CLD.java:942)
                    result: -8
```

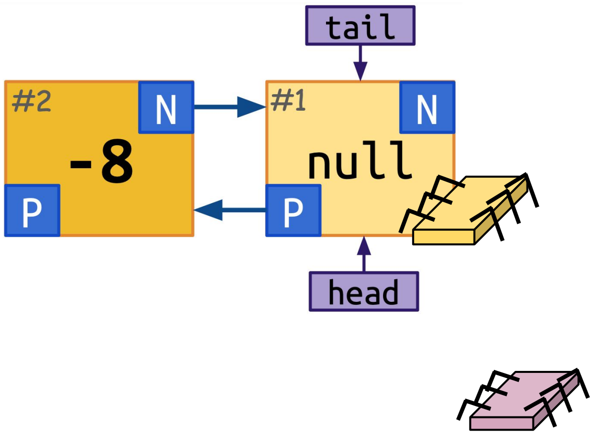
Bug in ConcurrentLinkedDeque



addLast(-6)
peekFirst(): -8

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

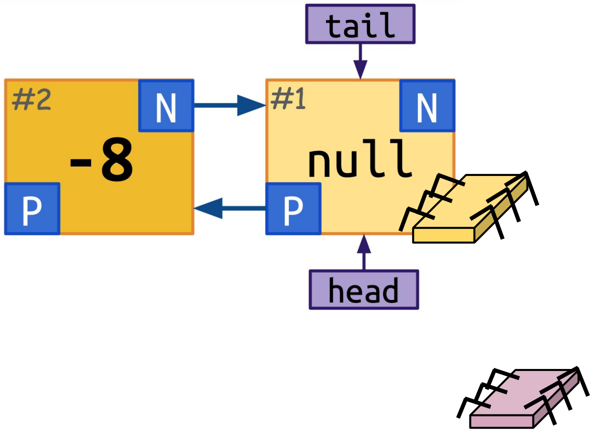
Bug in ConcurrentLinkedDeque



```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

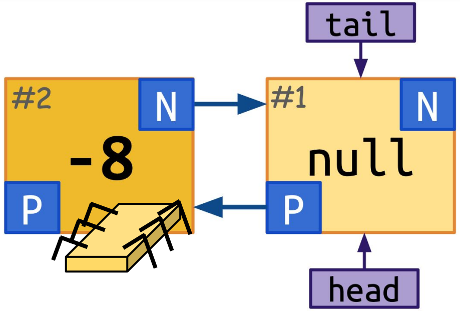
Bug in ConcurrentLinkedDeque

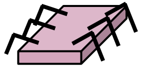


```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

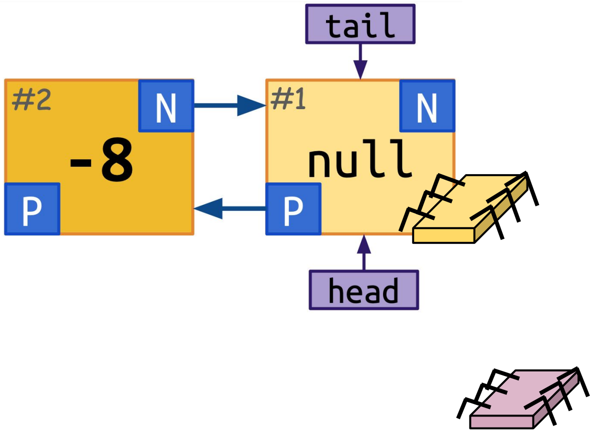




```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

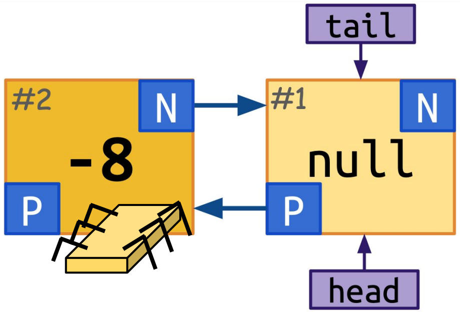
Bug in ConcurrentLinkedDeque

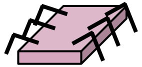


```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

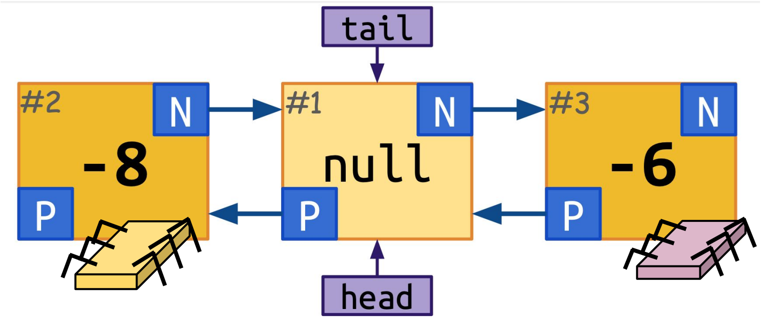




```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

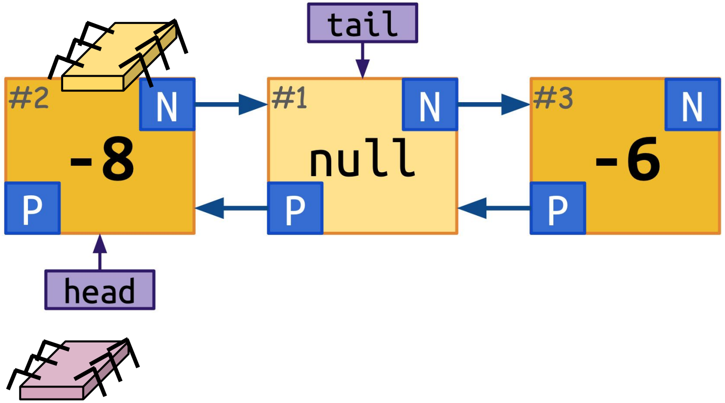
Bug in ConcurrentLinkedDeque



```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

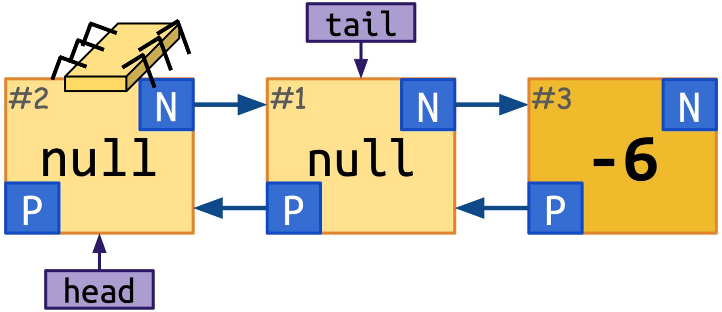
Bug in ConcurrentLinkedDeque



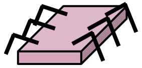
```
addLast(-6)
peekFirst(): -8
```

```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

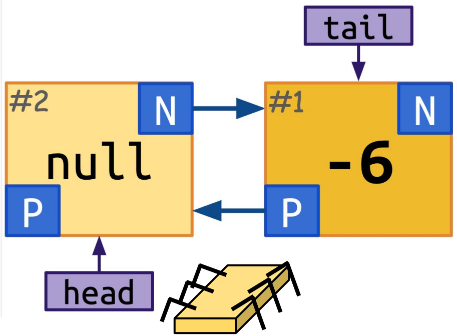


```
addLast(-6)
peekFirst(): -8
```

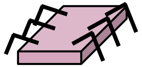


```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

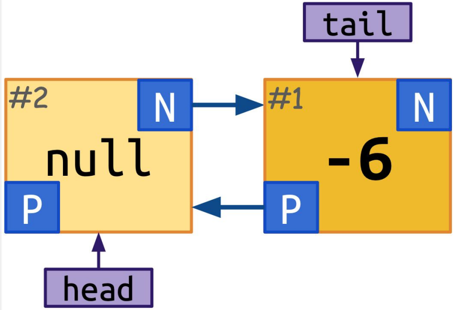


```
addLast(-6)
peekFirst(): -8
```



```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

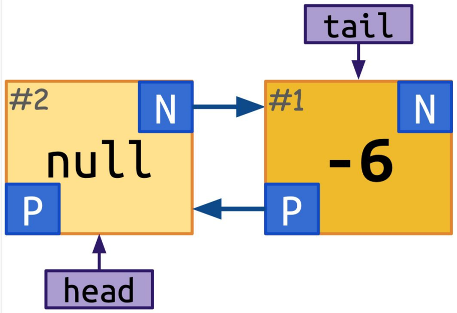


```
addLast(-6)
peekFirst(): -8
```

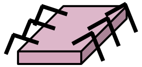
```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```

Bug in ConcurrentLinkedDeque

Cannot synchronize via this CAS



```
addLast(-6)
peekFirst(): -8
```



```
addFirst(-8)
pollLast(): -8
 pollLast(): -8 at DequeTest.pollLast(DequeTest.kt:35)
  last(): Node@1 at CLD.pollLast(CLD.java:936)
  item.READ: null at CLD.pollLast(CLD.java:938)
  prev.READ: Node@2 at CLD.pollLast(CLD.java:946)
  item.READ: -8 at CLD.pollLast(CLD.java:938)
  next.READ: null at CLD.pollLast(CLD.java:940)


  item.CAS(-8,null): true at CLD.pollLast(CLD.java:941)
  unlink(Node@2) at CLD.pollLast(CLD.java:942)
 result: -8
```