

Múltiplas Variáveis de Condição

Sistemas Distribuídos



Armazém

```
class Warehouse {  
    private Lock l = new ReentrantLock();  
    private Map<String, Item> stock = new HashMap<>();  
  
    private class Item {  
        int q = 0;  
        Condition isEmpty = l.newCondition();  
    }  
  
    public void supply (String item, int quantity);  
    public void consume (String[] items)  
        throws InterruptedException;  
}
```

Armazém

● Método Warehouse.supply

```
void supply (String item, int quantity) {  
    Item it = this.get(item);  
    it.q += quantity;  
}
```

● Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {  
    for (String item : items) {  
        Item it = this.get(item);  
        it.q -= 1;  
    }  
}
```

Armazém

- Cliente Greedy
- Método Warehouse.supply

```
public void supply (String item, int quantity) {  
    l.lock();  
    Item it = get(item);  
    it.q += quantity;  
    it.isEmpty.signalAll();  
    l.unlock();  
}
```

Armazém

- Cliente Greedy
- Método Warehouse.supply

```
public void supply (String item, int quantity) {  
    l.lock();  
    Item it = get(item);  
    it.q += quantity;  
    it.isEmpty.signalAll();  
    l.unlock();  
}
```



Armazém

- Cliente Greedy
- Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {  
    l.lock();  
    for (String item : items) {  
        Item it = this.get(item);  
        while (it.q == 0) {  
            it.isEmpty.await();  
        }  
        it.q -= 1;  
    }  
    l.unlock();  
}
```

Armazém

- Cliente Greedy
- Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {  
    l.lock();  
    for (String item : items) {  
        Item it = this.get(item);  
        while (it.q == 0) {  
            it.isEmpty().await();  
        }  
        it.q -= 1;  
    }  
    l.unlock();  
}
```

consume_

Abstração do operador
consumir, para um único item.



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock:	em espera p/ lock:
Wait-Set:	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T ₁	em espera p/ lock: T _{2,T3}
Wait-Set:	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



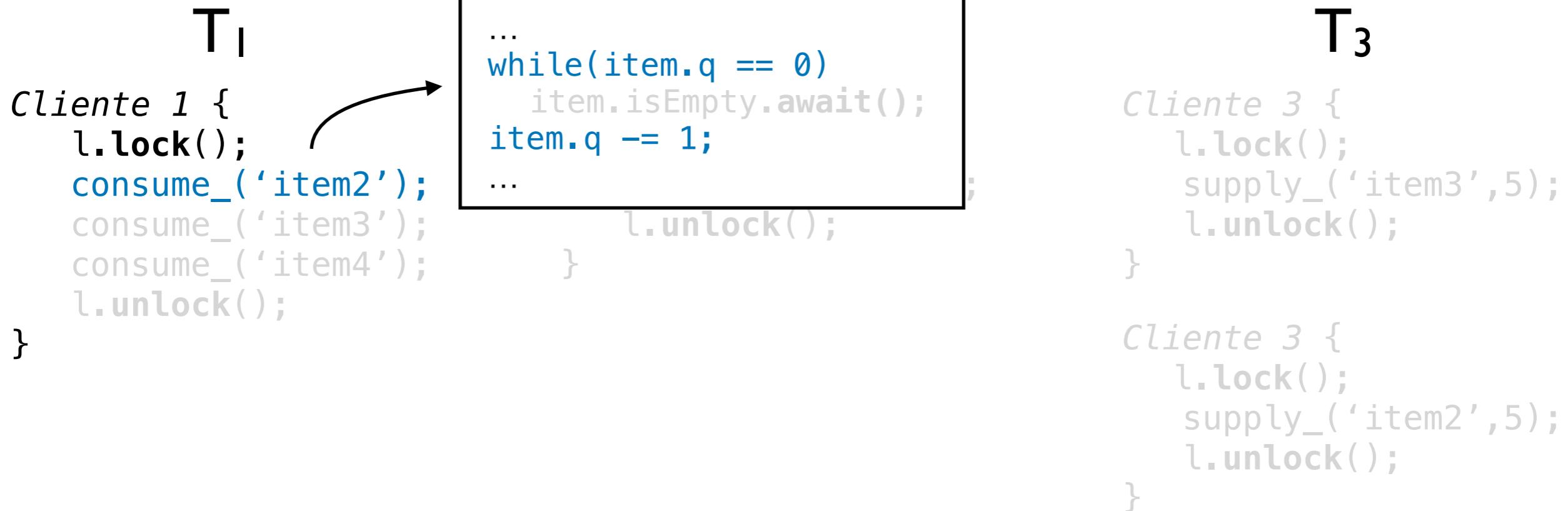
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T1	em espera p/ lock: T2,T3
Wait-Set:	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5

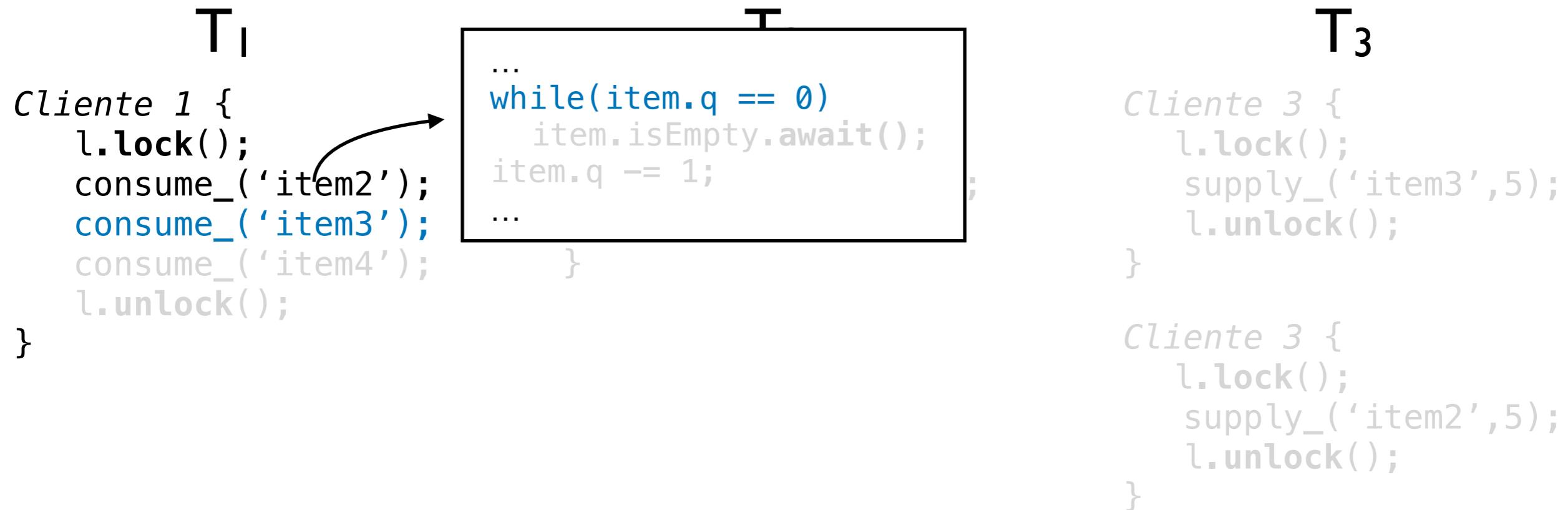
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T1	em espera p/ lock: T2,T3
Wait-Set:	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5

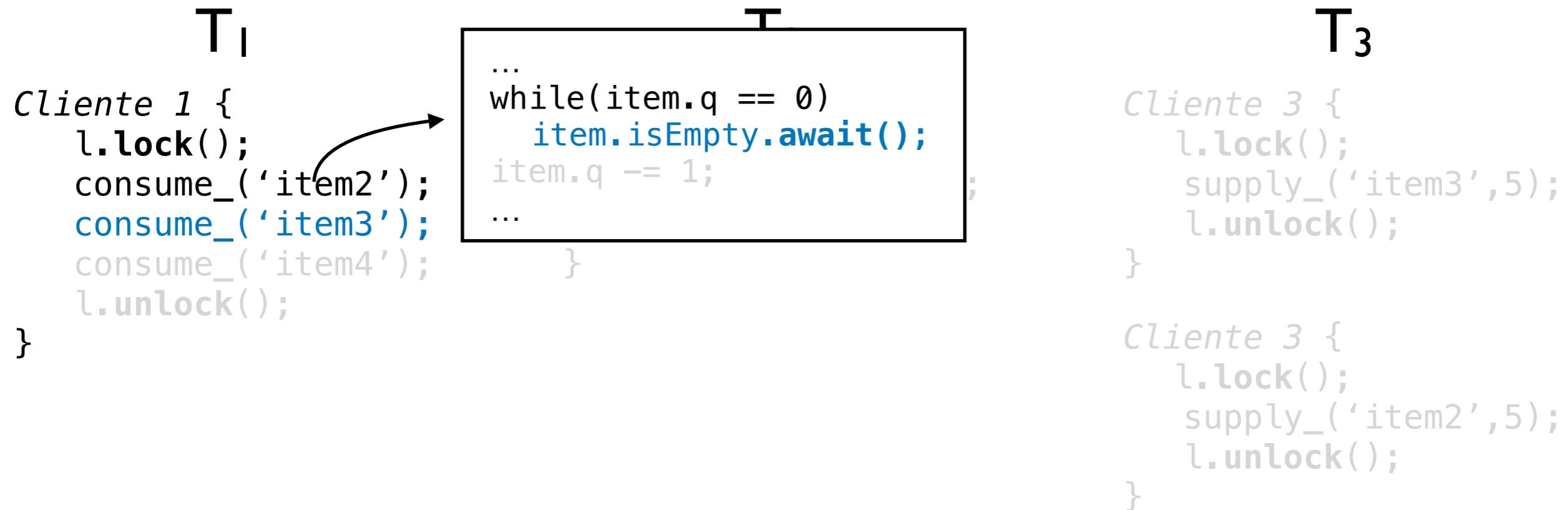
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock:	em espera p/ lock: T2, T3
Wait-Set:	T1(item3)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

```
...  
while(item.q == 0)  
    item.isEmpty.await();  
item.q -= 1;  
...  
l.unlock();  
}  
5);
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T₂ **em espera p/ lock:** T₃
Wait-Set: T₁(item3)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5



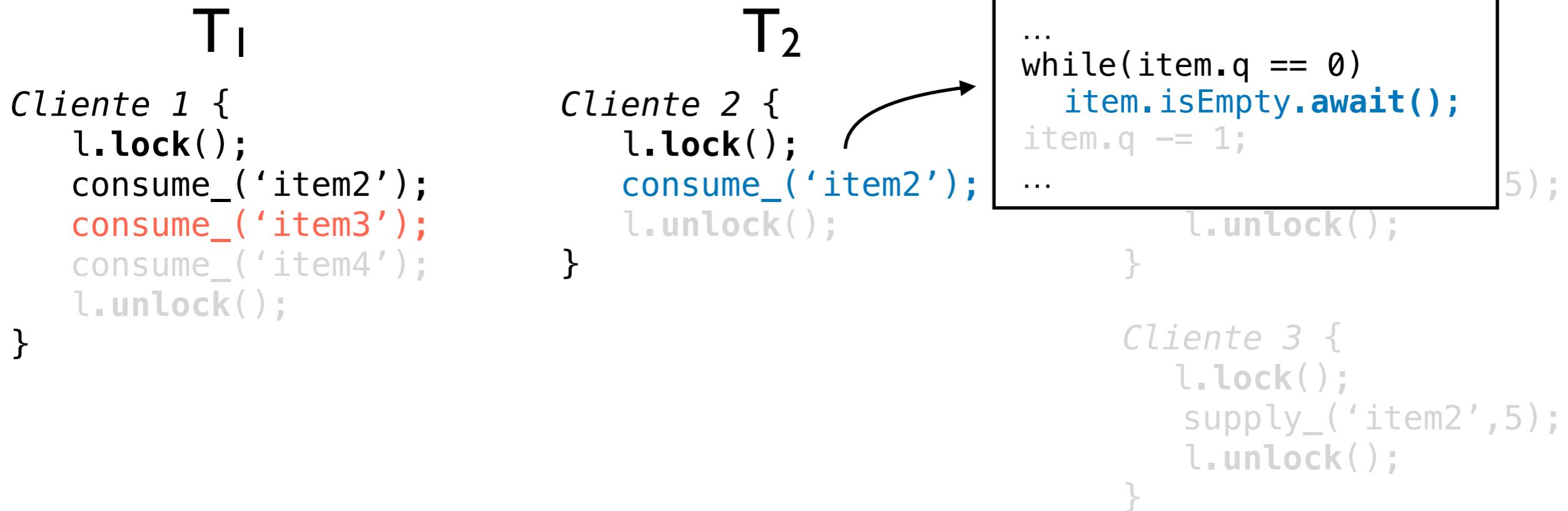
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: em espera p/ lock: T3
Wait-Set: T1(item3), T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3', 5);  
    l.unlock();  
}
```

Embora o Cliente 1 esteja à espera do 'item3', este já consumiu o 'item2'. Desta forma, o Cliente 2 ficará bloqueado até ao reabastecimento do 'item2'.

Lock: em espera p/ lock: T3

Wait-Set: T1(item3), T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T3	em espera p/ lock:
Wait-Set: T1(item3), T2(item2)	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5

Cliente Greedy

Cliente 1: consume ([‘item2’, ‘item3’, ‘item4’])

Cliente 2: consume ('item2')

Cliente 3: supply ('item3', 5)

Cliente 3: supply ('item2', 5)

T_I

```
    ...
    item.q += quantity;
    item.isEmpty.signalAll();
    ...
    l.unlock();
}
```

```
T3  
cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T3 *em espera p/ lock:*

Wait-Set: T1(item3), T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5

Cliente Greedy

Cliente 1: consume ([‘item2’, ‘item3’, ‘item4’])

Cliente 2: consume ('item2')

Cliente 3: supply ('item3', 5)

Cliente 3: supply ('item2', 5)

T_I

```
    ...
    item.q += quantity;
    item.isEmpty.signalAll();
    ...
    l.unlock();
}
```

```
cliente 3 {
    l.lock();
    supply_('item3',5);
    l.unlock();
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T3 *em espera p/ lock:*

Wait-Set: T1(item3), T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5

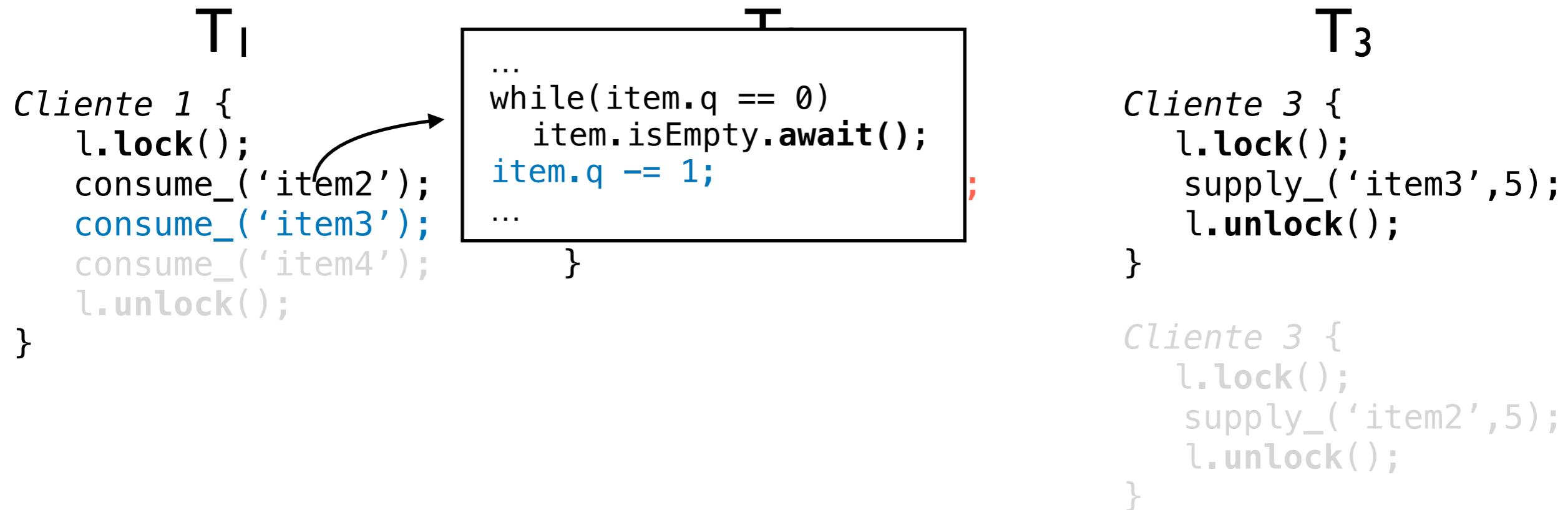
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T_1	em espera p/ lock: T_3
Wait-Set: $T_2(\text{item2})$	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	4	5



Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3', 5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2', 5);  
    l.unlock();  
}
```

Lock: T1	em espera p/ lock: T3
Wait-Set: T2(item2)	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	4	4

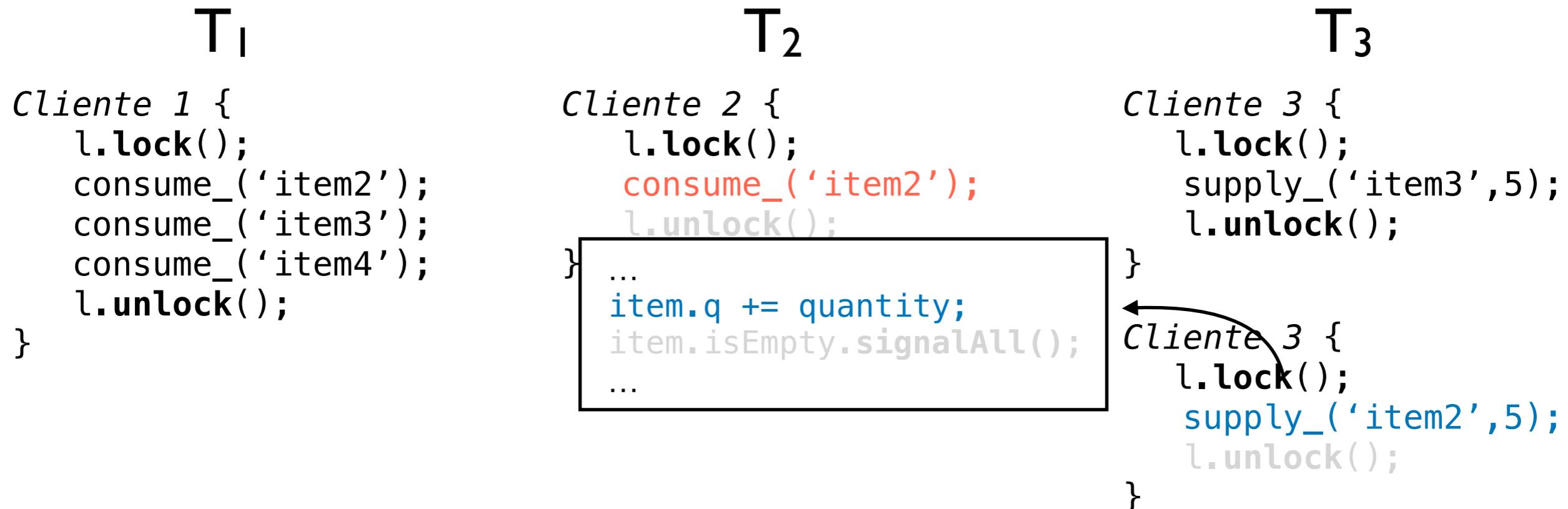
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T3 **em espera p/ lock:**

Wait-Set: T2(item2)

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	4	4

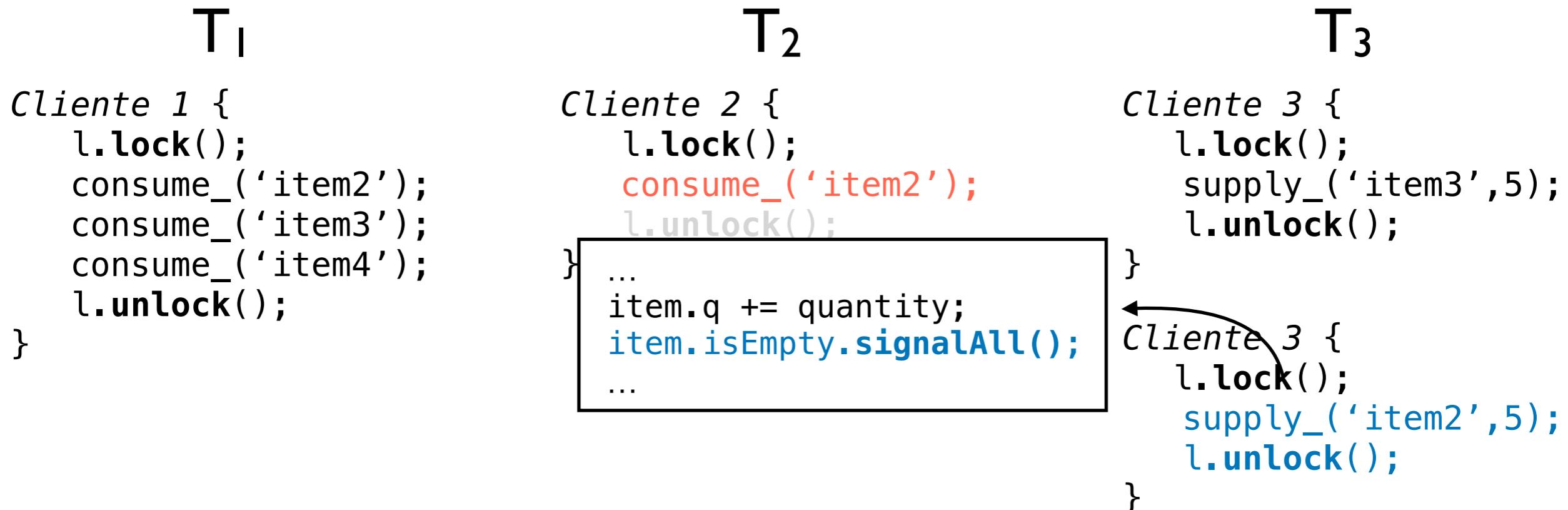
Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T3	em espera p/ lock:
Wait-Set: T2(item2)	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	4	4

Cliente Greedy

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {
    l.lock();
    consume_('item2');
    consume_('item3');
    consume_('item4');
    l.unlock();
}
```

T₂

```
Cliente 2 {
    l.lock();
    consume_('item2');
    l.unlock();
}
```

...

```
... while(item.q == 0)
    item.isEmpty.await();
    item.q -= 1;
...
    l.unlock();
}
```

T₃

```
Cliente 3 {
    l.lock();
    supply_('item2',5);
    l.unlock();
}
```

Lock: T₂ **em espera p/ lock:**

Wait-Set:

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	4	4

Armazém

Cliente Cooperativo

Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {  
    l.lock();  
    int i = 0;  
    while (i < items.length) {  
        Item it = this.get(items[i]);  
        i++;  
        while (it.q == 0) {  
            it.isEmpty().await();  
            i = 0;  
        }  
    }  
    for (String item : items) {  
        this.get(item).q -= 1;  
    }  
    l.unlock();  
}
```

Armazém

- Cliente Cooperativo
- Método Warehouse.consume

```
void consume (String[] items) throws InterruptedException {  
    l.lock();  
    int i = 0;  
    while (i < items.length) {  
        Item it = this.get(items[i]);  
        i++;  
        while (it.q == 0) {  
            it.isEmpty().await();  
            i = 0;  
        }  
    }  
    for (String item : items) {  
        this.get(item).q -= 1;  
    }  
    l.unlock();  
}
```

preconsume_

Operador que regista a intenção de consumir um item.

consume_

Abstração do operador consumir, para um único item.





Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock:	em espera p/ lock:
Wait-Set:	registro:

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T ₁	em espera p/ lock: T _{2,T3}
Wait-Set:	registro: T ₁ = 0

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



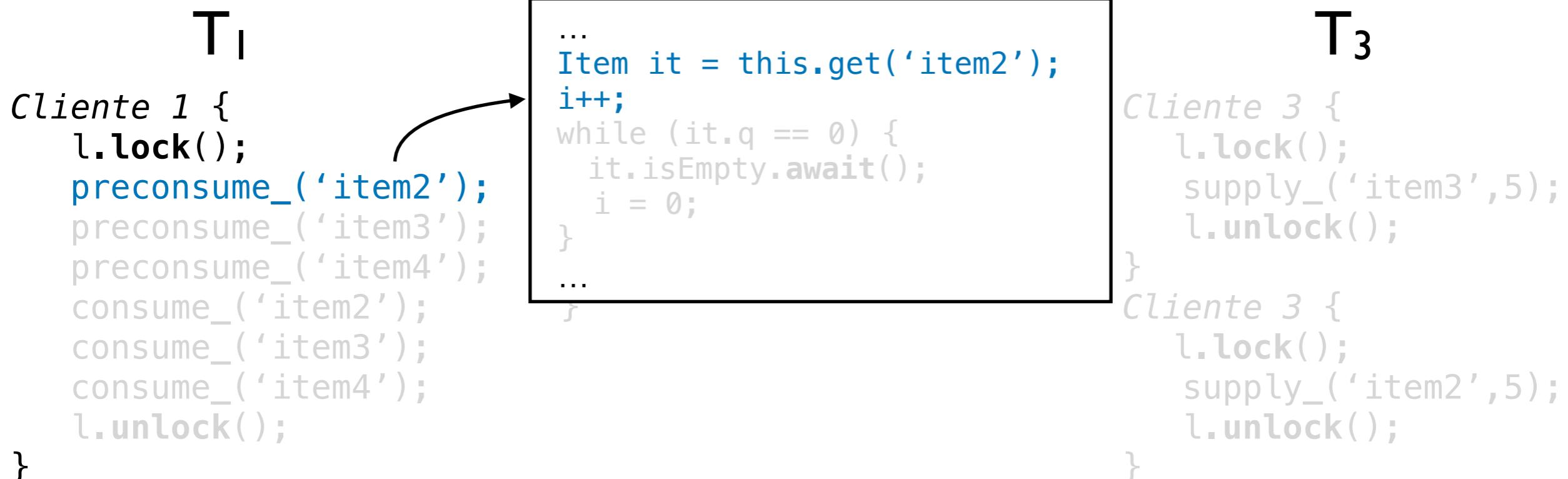
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T_1	em espera p/ lock: T_2, T_3
Wait-Set:	registro: $T_1 = I$

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



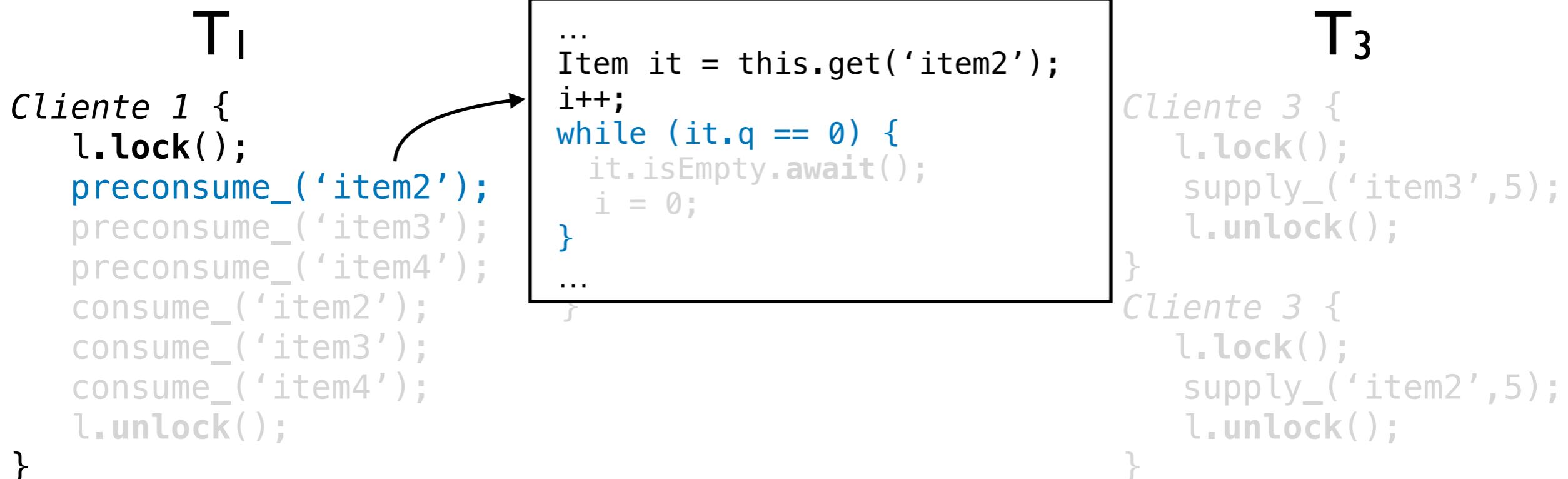
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T1	em espera p/ lock: T2,T3
Wait-Set:	registro: T1 = I

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



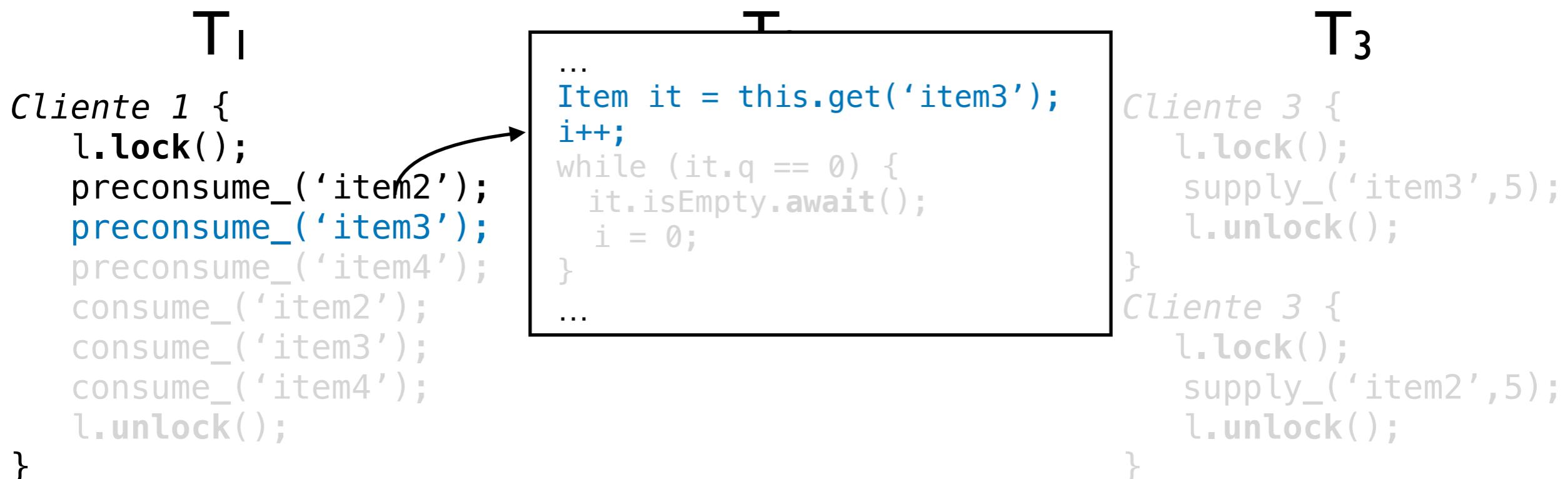
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T1 **em espera p/ lock:** T2,T3

Wait-Set: **registro:** T1 = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



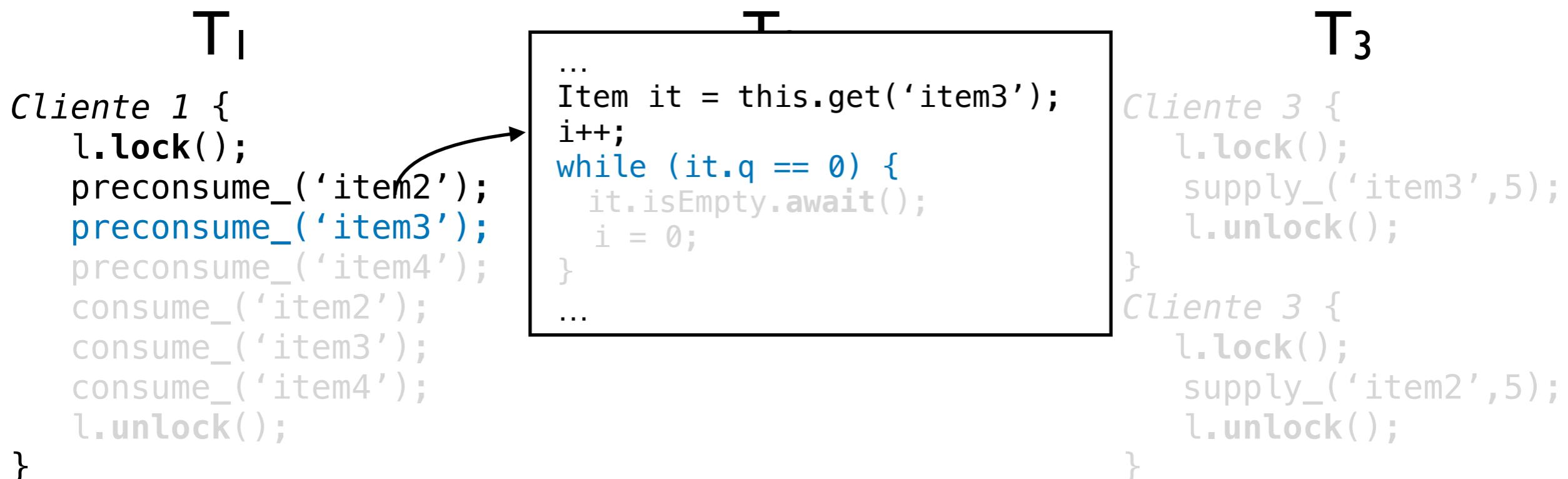
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T1	em espera p/ lock: T2,T3
Wait-Set:	registro: T1 = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



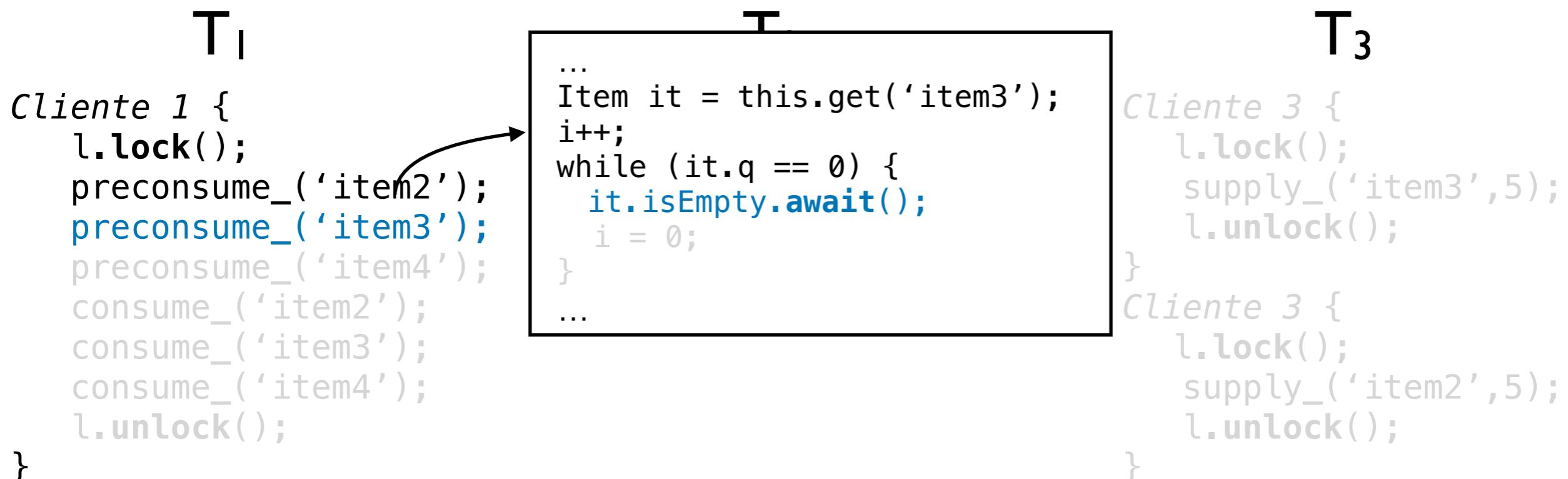
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock:	em espera p/ lock: T2,T3
Wait-Set: T1(item3) registro: T1 = 2	

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2'  
    consume_('item2');  
    l.unlock();  
}
```

```
...  
Item it = this.get('item2');  
i++;  
while (it.q == 0) {  
    it.isEmpty.await();  
    i = 0;  
}  
...  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T₂ **em espera p/ lock:** T₃
Wait-Set: T₁(item3) **registro:** T₁ = 2; T₂ = 1

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2'  
    consume_('item2');  
    l.unlock();  
}
```

```
...  
Item it = this.get('item2');  
i++;  
while (it.q == 0) {  
    it.isEmpty.await();  
    i = 0;  
}  
...  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

}

Lock: T2 **em espera p/ lock:** T3
Wait-Set: T1(item3) **registro:** T1 = 2;T2 = 1

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	1	0	5



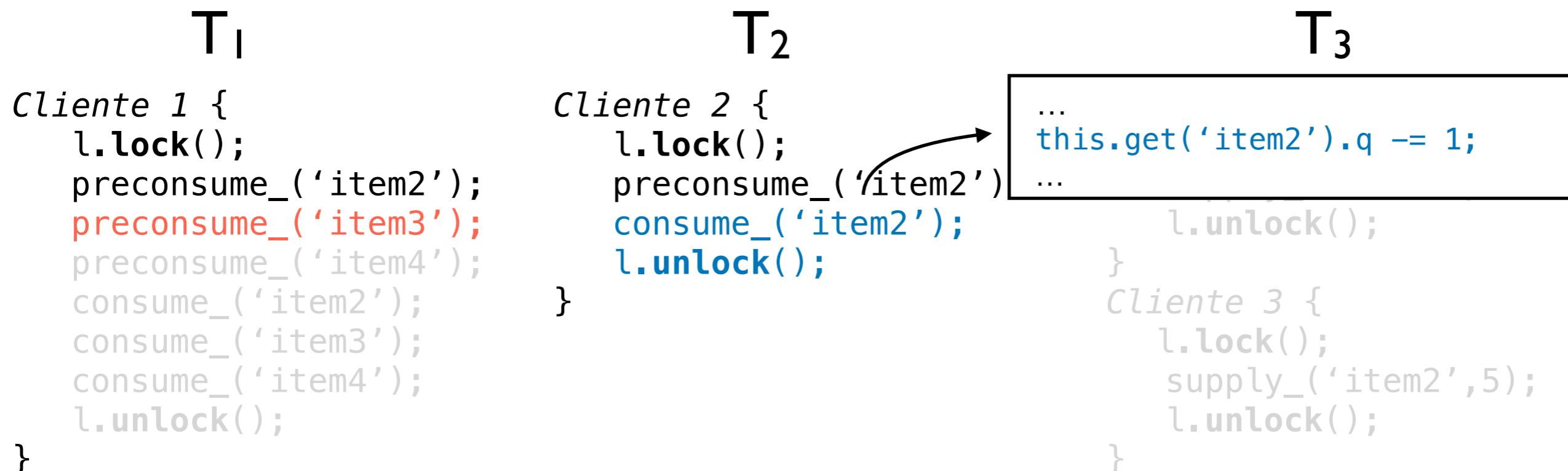
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: em espera p/ lock: T3

Wait-Set: T1(item3) **registro:** T1 = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	0	5



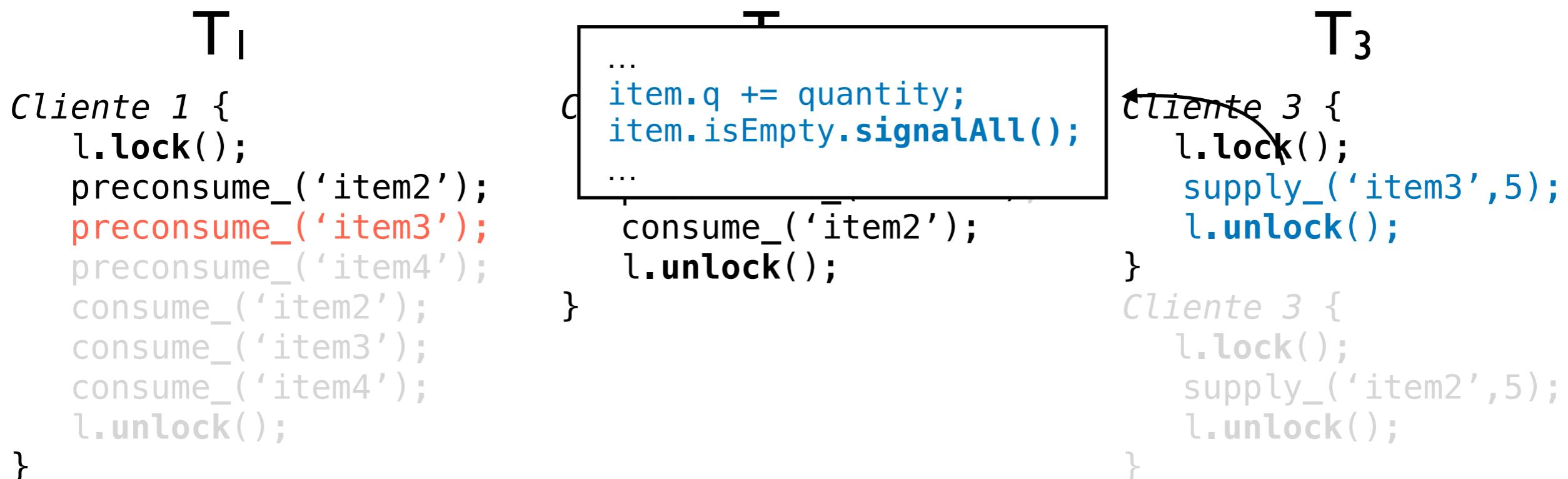
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T₃ **em espera p/ lock:** T₁

Wait-Set:

registro: T₁ = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	0	5	5



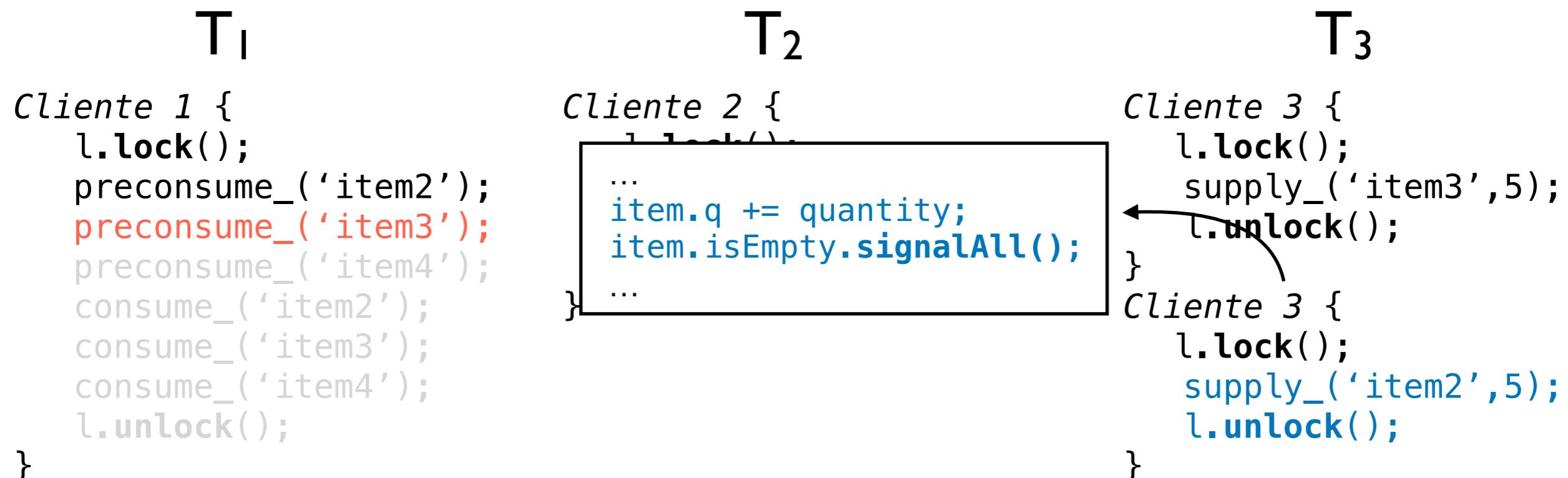
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T₃ **em espera p/ lock:**

Wait-Set: T₁(item3) **registro:** T₁ = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5



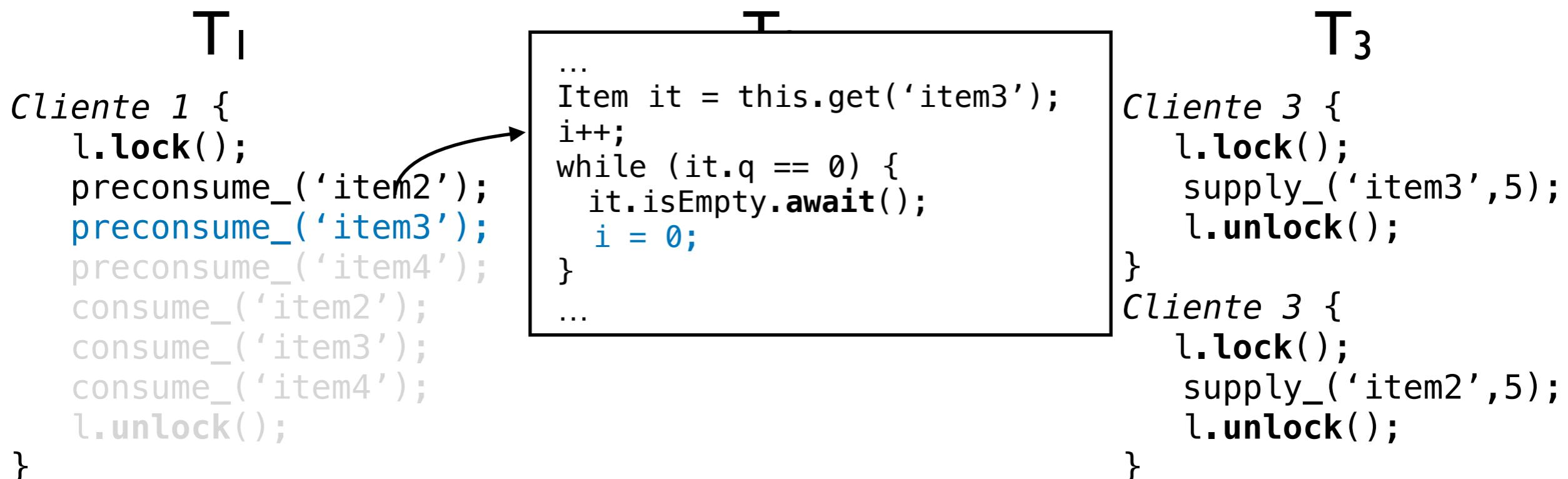
Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)



Lock: T₁ em espera p/ lock:

Wait-Set: registro: T₁ = 0

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T1 **em espera p/ lock:**
Wait-Set: **registro:** T1 = l

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T1	em espera p/ lock:
Wait-Set:	registro: T1 = 2

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4');  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    l.unlock();  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T1	em espera p/ lock:
Wait-Set:	registro: T1 = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	5	5	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2')  
    preconsume_('item3')  
    preconsume_('item4')  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    ...  
    this.get('item2').q -= 1;  
    ...  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3', 5);  
    l.unlock();  
}
```

```
Cliente 3 {  
    l.lock();  
    supply_('item2', 5);  
    l.unlock();  
}
```

Lock: T1	em espera p/ lock:
Wait-Set:	registro: T1 = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	5	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3')  
    preconsume_('item4')  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    ...  
    this.get('item3').q -= 1;  
    ...
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3',5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2',5);  
    l.unlock();  
}
```

Lock: T1	em espera p/ lock:
Wait-Set:	registro: T1 = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	4	5



Cliente Cooperativo

Cliente 1: consume(['item2', 'item3', 'item4'])

Cliente 2: consume(['item2'])

Cliente 3: supply('item3', 5)

Cliente 3: supply('item2', 5)

T₁

```
Cliente 1 {  
    l.lock();  
    preconsume_('item2');  
    preconsume_('item3');  
    preconsume_('item4')  
    consume_('item2');  
    consume_('item3');  
    consume_('item4');  
    l.unlock();  
}
```

T₂

```
Cliente 2 {  
    l.lock();  
    preconsume_('item2');  
    consume_('item2');  
    ...  
    this.get('item4').q -= 1;  
    ...  
}
```

T₃

```
Cliente 3 {  
    l.lock();  
    supply_('item3', 5);  
    l.unlock();  
}  
Cliente 3 {  
    l.lock();  
    supply_('item2', 5);  
    l.unlock();  
}
```

Lock:	em espera p/ lock:
Wait-Set:	registro: T1 = 3

String	'item1'	'item2'	'item3'	'item4'
Item.quantity	10	4	4	4

Sugestões

- Tornar o stock de cada produto limitado
- Suportar o acesso concorrente ao armazém
- Adicionar os métodos de registar e remover produtos do armazém