Concurrent Libraries with Foresight

Guy Golan-Gueta, G. Ramlinga, Mooly Sagiv, Eran Yahav

Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation 2013

Nadja Müller
Concepts of Concurrent Computation 2014
Contents

• Problem Statement
• Foresight-Based Synchronization
  o Client Protocol
  o Implementing Libraries with Foresight
• Evaluation
Problem Statement

The aim is to extend a linearizable library to allow clients to perform an arbitrary composite operation that appears to execute atomically.

A composite operation is a sequence of library operations. A linearizable library provides operations that appear to execute atomically.
Correction Condition for Concurrency Control

• Serializable execution
  A Serializable execution of two threads is one that is equivalent to either thread T1 executing completely before T2 executes or vice versa.

• No deadlocks

• No rollbacks
Example Library Maps

Class Maps {
    int createNewMap();
    int put (int mapId, int k, int v);
    int get(int mapId, int k);
    int remove(int mapId, int k);
    bool isEmpty (int mapId);
    int size (int mapId);
}
Contents

• Problem Statement
• Foresight-Based Synchronization
  o Client Protocol
  o Implementing Libraries with Foresight
• Evaluation
Clients

• Multiple threads
  o Statements changing only thread-local state
  o Statements that invoke a library operation
• No shared state except the state of the library
• Follows the client protocol
Client Protocol

Provide foresight information provided `mayUse operations`:
• stand for set of library functions the client may use
• The Client must have called the appropriate `mayUse` function before executing a library function
• The declared set should only shrink as the execution proceeds

Example:
• `mayUseAll()`: `CreateNewMap`, `put`, `get`, `remove`, `isEmpty`, `size`
• `mayUseMap(int m)`: `put`, `get`, `remove`, `isEmpty`, `size` on map `m`
• `mayUseKey(int m, int k)`: `put`, `get`, `remove` on map `m` with key `k`
• `mayUseNone()`: no library Operation
Example

If (get(m, x) == get(m, y)) {
    remove(m, x);
} Else {
    remove(m, x);
    remove(m, y);
}
Example

If (get(m,x) == get(m,y)) {
    remove(m,x); mayUseNone();
} Else {
    remove(m,x);
    remove(m,y); mayUseNone();
}
Example

mayUseMap(m);
If (get(m,x) == get(m,y)){
    remove(m,x); mayUseNone();
Else{
    remove(m,x);
    remove(m,y); mayUseNone();
}

Example

mayUseMap(m);
If (get(m,x) == get(m,y)){
    mayUseKey(m,x); remove(m,x); mayUseNone();
Else{
    remove(m,x);
    mayUseKey(m,y); remove(m,y); mayUseNone();
}
Library

• The Library is extended with additional procedures, which are used for synchronization.

• The Extension should have the following properties:
  o Progress
  o If the client follows the Client Protocol and is completable, then every execution is completable and serializable
Library Extension Implementation

- Translate semantic properties into Tree Structure
  - Every child allows a subset of library operations of its parent
  - Different parameterization allows finer granularity
- MayUse functions follow the Locking Algorithm and make sure no child is locked before proceeding

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1
Locking Algorithm

First invocation of a mayUse operation m, locking node P(m):
- Obtain a lock of the root
- Follow the path in the tree, locking each node in the path including P(m)
- Unlock all nodes except P(m)

Invocation of a mayUse operation m’ by a thread holding the lock on P(m):
- Lock all nodes in path from P(m) to P(m’)

Invocation of mayUseNone():
- Release all locks
Example

- `mayUseMap(1);`
  
  If \( \text{get}(1,1) == \text{get}(1,1) \) {
    `mayUseKey(1,1);`
    `remove(m,x);`
    `mayUseNone();`
  }

1: `mayUseAll()`
2: `mayUseMap(mapId), mapId %2 = 0`
3: `mayUseMap(mapId), mapId %2 = 1`
4: `mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0`
5: `mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1`
6: `mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0`
7: `mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1`

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

- `mayUseMap(1);`
  ```
  If (get(1,1) == get(1,1)) {
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
  }
  ```

1: `mayUseAll()`
2: `mayUseMap(mapId), mapId %2 = 0`
3: `mayUseMap(mapId), mapId %2 = 1`
4: `mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0`
5: `mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1`
6: `mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0`
7: `mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1`

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

- mayUseMap(1);
  If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
  }

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

- mayUseMap(1);
  If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
  }

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

- \texttt{mayUseMap(1);}
  - If (get(1,1) == get(1,1))
    - \texttt{mayUseKey(1,1);}
    - \texttt{remove(m,x);} \\
    - \texttt{mayUseNone();}

1: \texttt{mayUseAll()}
2: \texttt{mayUseMap(mapId), mapId \%2 = 0}
3: \texttt{mayUseMap(mapId), mapId \%2 = 1}
4: \texttt{mayUseKey(mapId, k), mapId \%2 = 0, k\%2 = 0}
5: \texttt{mayUseKey(mapId, k), mapId \%2 = 0, k\%2 = 1}
6: \texttt{mayUseKey(mapId, k), mapId \%2 = 1, k\%2 = 0}
7: \texttt{mayUseKey(mapId, k), mapId \%2 = 1, k\%2 = 1}

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

mayUseMap(1);

- If (get(1,1) == get(1,1)){
  mayUseKey(1,1);
  remove(m,x);
  mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId, mapId % 2 = 0
3: mayUseMap(mapId, mapId % 2 = 1
4: mayUseKey(mapId, k), mapId % 2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId % 2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId % 2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId % 2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

mayUseMap(1);
If (get(1,1) == get(1,1)){
  mayUseKey(1,1);
  remove(m,x);
  mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis:  Unlocked
Orange:   Locked by another thread
Red:      Locked by own thread
Example

```
mayUseMap(1);
if (get(1,1) == get(1,1)) {
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}
```

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

mayUseMap(1);
If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

mayUseMap(1);
If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1
Example

mayUseMap(1);  
If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Example

mayUseMap(1);
If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
mayUseMap(1);
If (get(1,1) == get(1,1)){
    mayUseKey(1,1);
    remove(m,x);
    mayUseNone();
}

1: mayUseAll()
2: mayUseMap(mapId), mapId %2 = 0
3: mayUseMap(mapId), mapId %2 = 1
4: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 0
5: mayUseKey(mapId, k), mapId %2 = 0, k%2 = 1
6: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 0
7: mayUseKey(mapId, k), mapId %2 = 1, k%2 = 1

Turquis: Unlocked
Orange: Locked by another thread
Red: Locked by own thread
Contents

• Problem Statement

• Foresight-Based Synchronization
  o Client Protocol
  o Implementing Libraries with Foresight

• Evaluation
Evaluation

• ComputelIfAbsent-pattern:
  
  If(!map.containsKey(key)) {
      value = someComputation;
      map.put(key, value);
  }

Evaluation

• **ComputeIfAbsent-pattern:**
  
  ```
  If(!map.containsKey(key)){
    value = someComputation;
    map.put(key, value);
  }
  ```

• At most 25% slower than the hand-crafted fine-grained locking
Questions?