Algorithm for `ConcurrentLinkedSet`

Hanlin He*, Lizhong Zhang†

November 28, 2018

The goal of `ConcurrentLinkedSet` is to implement a concurrent linked list that supports four operations, namely `contains`, `insert`, `delete` and `replace`, using lazy synchronization approach.

Based on code of `add` and `remove` methods shown in the textbook, we designed the `replace` method as shown in algorithm 1. The key idea is:

- Add an addition field `replaceNode` in `Node` class an element.
- When creating the new node used in replace, refer the `replaceNode` to the node to be replaced.
- Logically, the replaced node was removed if it was marked. Likewise, the newly added node during `replace` was logically in the list only if the `replaceNode` node was marked.

*hxh160630@utdallas.edu
†lxz160730@utdallas.edu

Algorithm 1 Algorithm for `replace`

```
function replace(old, new)
    // if two elements are the same, semantically identical with add(x).
    if old ≡ new then
        add(new)
    end if
    while true do
        oldWindow ← locateWindow(old)
        newWindow ← locateWindow(new)
        Lock all nodes in oldWindow and newWindow from left to right.
        Validate both windows.
        if old ∉ oldWindow then
            if new ∉ newWindow then
                Add new node directly (like add(new)).
            else
                return false
            end if
        else
            if new ∈ newWindow then
                Delete old node directly (like delete(new)).
            else
                Create new node with replaceNode point to old node.
                Add new in newWindow.
                Set marked flag to true in old node.      ▷ Linearization point.
                Bypass old node.
                Unset replaceNode field in new
            end if
        end if
        Unlock all nodes in oldWindow and newWindow from right to left.
        return true
    end while
end function
```

To test the implementation, we run 4 to 8 thread each randomly modify (add/remove/replace) elements in the linked set. For each modification, we check if the linked list was still sorted, and abort if not. After each thread finished, we will do one final check if the linked list is sorted, and then output the duration for all iteration.