1 Linked-List

The original algorithm by Harris is presented in Figure 1. Harris approach uses an Atomic-Markable-Reference object, in which the next field of a Node, in addition to a reference to the next node in the list, is also marked or unmarked. The two fields can be update atomically, either together or individually. This can be done by using the most-significant-bit of next for the marking. For simplicity, we assume node.next to return the reference, while one can ask whether it is marked or unmarked. Therefore, whenever writing to node.next, or performing CAS, both the reference and the marking state should be mention.

```
Procedure Find(int key)
    Data: Node* pred, curr, succ
 1  retry: while true do
 2      pred = head;
 3      curr = head.next;
 4      while true do
 5          succ = curr.next;
 6          if curr.next is marked then
 7              if pred.next.CAS(unmarked curr,unmarked succ) == false then
 8                  go to retry;
 9              end
10              curr = succ;
11          else
12              if curr.key ≥ key then
13                  return <pred,curr>;
14              end
15              pred = curr;
16              curr = succ;
17          end
18      end
19  end
```

Shared variables: Node* head

Procedure Insert(int key)

```
   Data: Node* pred, curr;
         Node node = new Node (key);
20 while true do
21 |  <pred, curr> = find(key);
22 |  if curr.key == key then
23 |  |  return false;
24 |  else
25 |  |  node.next = unmarked curr;
26 |  |  if pred.next.CAS (unmarked curr, unmarked node) then
27 |  |  |  return true;
28 |  |  end
29 |  end
30 end
```

Procedure Delete(int key)

```
   Data: Node* pred, curr, succ
31 while true do
32 |  <pred, curr> = find(key);
33 |  if curr.key != key then
34 |  |  return false;
35 |  else
36 |  |  succ = curr.next;
37 |  |  if curr.next.CAS (unmarked succ, marked succ) then
38 |  |  |  pred.next.CAS (unmarked curr, unmarked succ);
39 |  |  |  return true;
40 |  |  end
41 |  end
42 end
```

Procedure Contains(int key)

```
   Data: Node* curr = head
43 while curr.key ¡ key do
44 |  curr = curr.next;
45 end
46 return (curr.key == key && curr.next is unmarked);
```

Figure 1: Harris Non-Blocking Algorithm