Note Title 3/20/2008

Non-blocking and wait-free algorithms – using CAS directly

Why? Low overhead in uncontested case. –
Very narrow congestion windows.

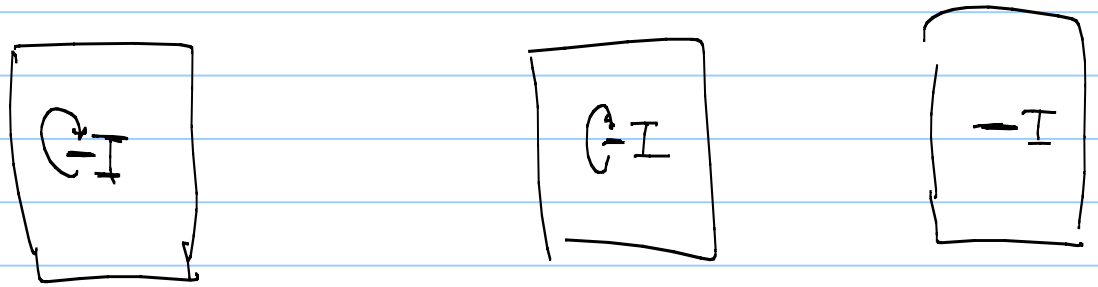
Why not? Hard –

Definition: Non-blocking means suspension of one thread cannot prevent other threads from making progress.

observe CAS – uncontested always succeeds.
– contested case, at least one thread makes progress

How do they work (in general)?

- Identify a single update that makes the transition from one valid state to the next.
- Be willing to redo some work if CAS fails

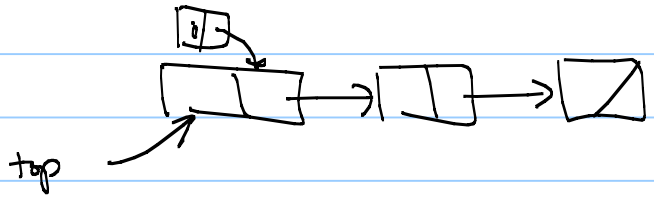


Simple example: counter

```
private CAS int value;
public int getValue() { return value.get(); }
public int increment() {
    int v;
    do {
        v = value.get();
    } while (v != value.CASwap(v, v+1));
    return v+1;
}
```

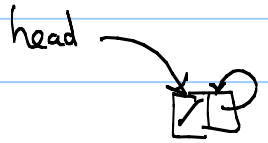
Contention window

Stack implemented as a linked list.




```
AtomicReference top;
public void push(item) {
    Node newHead = new Node(item);
    do {
        newHead.next = top.getC();
    } while (!top.CASet(newHead.next, newHead))
}
```

```
public T pop() {
    Node newHead;
    Node oldHead;
    do {
        oldHead = top.get();
        if (oldHead == null) return null;
        newHead = oldHead.next;
    } while (!top.CASet(oldHead, newHead));
    return oldHead.value
}
```

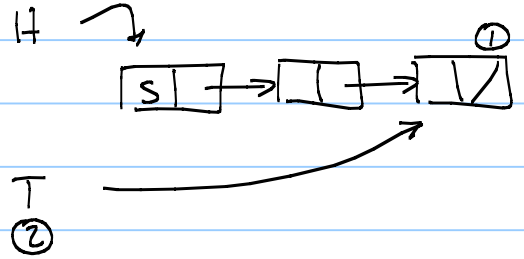


oldHead = top.get()
newHead = oldHead.next()
} while (!top.CASet(...))

A queue.

Insert at tail

Remove at Head



Consistent:

~~Tail pointer refers to the sentinel~~

Tail pointer refers to the last element } normal

or TP refers to the next-to-last element } update in Progress UIP

Observe normal case T.next == null

UIP case T.next ≠ null

insert

```
Node   newNode = Node (item)
while (true) {
    Node  curtail = tail.get();
    Node  tailNext = curtail.next.get();
    if (curtail == tail.get()) {
        if (tailNext != null) {
            tail.CASet(curtail, tailNext);
        } else {
            if (curtail.next.CASet(null, newNode)) {
                tail.CASet(curtail, newNode)
                return
            }
}       }
```