```
Load-Locked(addr)

Lock()
add addr to "locked list"
load from addr
Memory-Barrier(acquire)
mark addr in "locked bitmap"
Unlock()
```

```
Store-Conditional(addr, value)

Lock()
if $addr \in$ "locked list"
    unmark addr in "locked bitmap"
    remove addr from "locked list"
    store value into addr
    Unlock()
    return success
else
    Unlock()
    return failure
```

```
Store(addr, value)

if mark[addr]
    /* slow path */
    Lock()
    unmark addr in "locked bitmap"
    remove addr from "locked list"
    Unlock()
Memory-Barrier(acquire)
store value into addr
```

Figure 1. A flawed implementation of *load-locked/store-conditional*, attempting to make the Store fast path wait free. The locked cachelines are stored in a "locked list" and also marked (e.g. in a bitmap) for use by Store. However, if Store reads the mark before Load-Locked has set it, Store-Conditional will not notice the conflict.

```
Load-Locked(addr)

Lock()
add addr to "locked list"
mark addr in "locked bitmap"
Memory-Barrier(full)
load from addr
Unlock()
```

```
Store-Conditional(addr, value)

Lock()
if $addr \in$ "locked list"
    unmark addr in "locked bitmap"
    remove addr from "locked list"
    store value into addr
    Unlock()
    return success
else
    Unlock()
    return failure
```

```
Store(addr, value)

if mark[addr]
    /* slow path */
    Lock()
    unmark addr in "locked bitmap"
    remove addr from "locked list"
    Unlock()
Memory-Barrier(acquire)
store value into addr
```

Figure 2. Another flawed implementation of *load-locked/store-conditional*. Again Store can read the mark before Load-Locked has set it. If it stores the new value after Load-Locked has read the memory, Store-Conditional will not notice the conflict.

```
Load-Locked(addr)

Lock()
add addr to "locked list"
mark addr in "locked bitmap"
load from addr
Unlock()
```

```
Store-Conditional(addr, value)

Lock()
if $addr \in$ "locked list"
    remove addr from "locked list"
    store value into addr
    Unlock()
    return success
else
    Unlock()
    return failure
```

```
Store(addr, value)

transaction
    if mark[addr]
        abort
    store value into addr
on abort do
    Lock()
    unmark addr in "locked bitmap"
    remove addr from "locked list"
    store value into addr
    Unlock()
```

Figure 3. An example of a working implementation of *load-locked/store-conditional*, which however requires hardware transactional memory capabilities.