```
──────────────────── MODULE AtomicPtr ────────────────────
EXTENDS TLC, Integers, Sequences

CONSTANTS Node, nil

VARIABLES pc, ref_states, pointer, local_ref

vars ≜ ⟨pc, ref_states, pointer, local_ref⟩


RefState ≜ [ref_count : 0 . . 20, is_zero : BOOLEAN , destroyed : 0 . . 30]

State ≜ {
    "Init",
    "LoadPointer", "IncreaseRefCount",
    "UseObject",
    "Decrease", "TryToSetZero", "Destroy",
    "StartSwapPtr", "UpdatePointer",
    "Terminated"
}

NullRefAddr ≜ (DOMAIN ref_states) ∪ {nil}

TypeOK ≜
    ∧   pc ∈ [Node → State]
    ∧   ref_states ∈ Seq(RefState)
    ∧   pointer ∈ DOMAIN ref_states
    ∧   local_ref ∈ [Node → NullRefAddr]


Init ≜
    ∧ pc = [n ∈ Node ↦ "Init"]
    ∧ local_ref = [n ∈ Node ↦ nil]
    ∧ ref_states = ⟨[ref_count ↦ 1, is_zero ↦ FALSE, destroyed ↦ 0]⟩
    ∧ pointer = 1


goto(n, l) ≜
    pc′ = [pc EXCEPT ![n] = l]


LoadPointerOrSwapPtr(n) ≜
    ∧ pc[n] = "Init"
    ∧ ∨ goto(n, "LoadPointer")
      ∨ goto(n, "StartSwapPtr")
    ∧ UNCHANGED pointer
    ∧ UNCHANGED ref_states
    ∧ UNCHANGED local_ref
```

```
$LoadPointer(n) \triangleq$
    $\land pc[n] =$ "LoadPointer"
    $\land local\_ref' = [local\_ref$ EXCEPT $![n] = pointer]$
    $\land goto(n,$ "IncreaseRefCount"$)$
    $\land$ UNCHANGED $ref\_states$
    $\land$ UNCHANGED $pointer$

$IncreaseRefCount(n) \triangleq$
    LET
        $addr \triangleq local\_ref[n]$
        $is\_zero \triangleq ref\_states[addr].is\_zero$
    IN
        $\land pc[n] =$ "IncreaseRefCount"
        $\land ref\_states' = [ref\_states$ EXCEPT $![addr].ref\_count = @ + 1]$
        $\land$ IF $is\_zero$
            THEN $goto(n,$ "LoadPointer"$)$
            ELSE $goto(n,$ "UseObject"$)$
        $\land$ UNCHANGED $local\_ref$
        $\land$ UNCHANGED $pointer$

$UseObject(n) \triangleq$
    $\land pc[n] =$ "UseObject"
    $\land goto(n,$ "Decrease"$)$
    $\land$ UNCHANGED $local\_ref$
    $\land$ UNCHANGED $pointer$
    $\land$ UNCHANGED $ref\_states$

$DecreaseRef(n) \triangleq$
    LET
        $addr \triangleq local\_ref[n]$
        $old\_state \triangleq ref\_states[addr]$
        $new\_count \triangleq old\_state.ref\_count - 1$
        $new\_state \triangleq [old\_state$ EXCEPT $!.ref\_count = new\_count]$
    IN
        $\land pc[n] =$ "Decrease"
        $\land ref\_states' = [ref\_states$ EXCEPT $![addr] = new\_state]$
        $\land$ IF $new\_count = 0$
            THEN $goto(n,$ "TryToSetZero"$)$
            ELSE $goto(n,$ "Terminated"$)$
        $\land$ UNCHANGED $pointer$
        $\land$ UNCHANGED $local\_ref$

$TryToSetZero(n) \triangleq$
```

```
    LET
        $addr \triangleq local\_ref[n]$
        $old\_count \triangleq ref\_states[addr].ref\_count$
        $old\_is\_zero \triangleq ref\_states[addr].is\_zero$
    IN
        $\land\ pc[n] = $ "TryToSetZero"
        $\land$ IF $old\_count = 0 \land old\_is\_zero = $ FALSE
            THEN
                $\land\ ref\_states' = [ref\_states$ EXCEPT $![addr].is\_zero = $ TRUE$]$
                $\land\ goto(n,$ "Destroy"$)$
            ELSE
                $\land$ UNCHANGED $ref\_states$
                $\land\ goto(n,$ "Terminated"$)$
        $\land$ UNCHANGED $pointer$
        $\land$ UNCHANGED $local\_ref$

$DestroyObject(n) \triangleq$
    LET
        $addr \triangleq local\_ref[n]$
    IN
        $\land\ pc[n] = $ "Destroy"
        $\land\ goto(n,$ "Terminated"$)$
        $\land\ ref\_states' = [ref\_states$ EXCEPT $![addr].destroyed = @ + 1]$
        $\land$ UNCHANGED $pointer$
        $\land$ UNCHANGED $local\_ref$

$StartSwapPtr(n) \triangleq$
    LET
        $new\_state \triangleq [ref\_count \mapsto 1,\ is\_zero \mapsto$ FALSE$,\ destroyed \mapsto 0]$
        $new\_addr \triangleq Len(ref\_states) + 1$
    IN
        $\land\ pc[n] = $ "StartSwapPtr"
        $\land\ goto(n,$ "UpdatePointer"$)$
        $\land\ ref\_states' = Append(ref\_states,\ new\_state)$
        $\land\ local\_ref' = [local\_ref$ EXCEPT $![n] = new\_addr]$
        $\land$ UNCHANGED $pointer$

$UpdatePointer(n) \triangleq$
    LET
        $addr \triangleq local\_ref[n]$
    IN
        $\land\ pc[n] = $ "UpdatePointer"
        $\land\ goto(n,$ "Decrease"$)$
        $\land\ pointer' = addr$
        $\land\ local\_ref' = [local\_ref\ \text{EXCEPT}\ ![n] = pointer]$
        $\land\ \text{UNCHANGED}\ ref\_states$

$TerminateCond \triangleq$
    $\land\ \forall\, n \in Node : pc[n] = \text{“Terminated”}$

$Terminated \triangleq$
    $\land\ TerminateCond$
    $\land\ \text{UNCHANGED}\ vars$

$Next \triangleq$
    $\lor\ \exists\, n \in Node :$
        $\lor\ LoadPointerOrSwapPtr(n)$

        $\lor\ LoadPointer(n)$
        $\lor\ IncreaseRefCount(n)$
        $\lor\ UseObject(n)$

        $\lor\ DecreaseRef(n)$
        $\lor\ TryToSetZero(n)$
        $\lor\ DestroyObject(n)$

        $\lor\ StartSwapPtr(n)$
        $\lor\ UpdatePointer(n)$
    $\lor\ Terminated$

$Spec \triangleq Init \land \Box[Next]_{vars}$

$FairSpec \triangleq Spec \land \mathrm{WF}_{vars}(Next)$

$nonPrimaryRefStateDestroyed(i) \triangleq$
    $i \neq pointer \Rightarrow ref\_states[i].destroyed = 1$

$DestroyOnce \triangleq$
    $TerminateCond \Rightarrow$
        $(\forall\, i \in \text{DOMAIN}\ ref\_states : nonPrimaryRefStateDestroyed(i))$

$UseStateNotDestroyed \triangleq$
    $\forall\, n \in Node :$
        $pc[n] = \text{“UseObject”} \Rightarrow ref\_states[local\_ref[n]].destroyed = 0$

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$IncreaseAlwaysLeadToUsable \triangleq$
```

$$\forall\, n \in \mathit{Node} :$$
$$\quad pc[n] = \text{“IncreaseRefCount”} \leadsto pc[n] = \text{“UseObject”}$$