MODULE $AtomicPtrV2$

EXTENDS TLC, $Naturals$, $Sequences$

CONSTANTS $Node$, nil

VARIABLES $pointer$, $counter$, $objects$, pc, $local\_addr$, $last\_counter$

$vars \triangleq \langle pointer, counter, objects, pc, local\_addr, last\_counter \rangle$

$Object \triangleq [ref : Nat, extra : Nat, added : \text{BOOLEAN}, destroyed : Nat]$

$NullAddr \triangleq (\text{DOMAIN } objects) \cup \{nil\}$

$State \triangleq \{$
"Init", "SwapPointer", "IncreaseRefAgain",
"LoadPointer", "IncreaseRef",
"DecreaseLocalCounter", "ClearExtraRef",
"UseObject",
"DecreaseRef", "DestroyObject", "Terminated"$\}$

$TypeOK \triangleq$
$\land \quad objects \in Seq(Object)$
$\land \quad pointer \in \text{DOMAIN } objects$
$\land \quad counter \in Nat$
$\land \quad pc \in [Node \to State]$
$\land \quad local\_addr \in [Node \to NullAddr]$
$\land \quad last\_counter \in [Node \to Nat]$

$Init \triangleq$
$\land objects = \langle [ref \mapsto 1, extra \mapsto 0, added \mapsto \text{FALSE}, destroyed \mapsto 0] \rangle$
$\land pointer = 1$
$\land counter = 0$
$\land pc = [n \in Node \mapsto$ "Init"$]$
$\land local\_addr = [n \in Node \mapsto nil]$
$\land last\_counter = [n \in Node \mapsto 0]$

$goto(n, l) \triangleq$
$\land pc' = [pc \text{ EXCEPT } ![n] = l]$

$AllocateNewObject(n) \triangleq$
$\land pc[n] =$ "Init"
$\land goto(n,$ "SwapPointer"$)$
$\land objects' = Append(objects, [$
$ref \mapsto 1, extra \mapsto 0, added \mapsto \text{FALSE}, destroyed \mapsto 0])$

```
    ∧ local_addr′ = [local_addr EXCEPT ![n] = Len(objects′)]
    ∧ UNCHANGED ⟨counter, pointer⟩
    ∧ UNCHANGED last_counter

SwapPointer(n) ≜
    ∧ pc[n] = "SwapPointer"
    ∧ pointer′ = local_addr[n]
    ∧ local_addr′ = [local_addr EXCEPT ![n] = pointer]
    ∧ IF counter = 0
        THEN
            ∧ goto(n, "DecreaseRef")
            ∧ UNCHANGED counter
        ELSE
            ∧ goto(n, "IncreaseRefAgain")
            ∧ counter′ = 0
    ∧ last_counter′ = [last_counter EXCEPT ![n] = counter]
    ∧ UNCHANGED objects

IncreaseRefAgain(n) ≜
    LET
        addr ≜ local_addr[n]
        diff ≜ last_counter[n] − objects[addr].extra
    IN
        ∧ pc[n] = "IncreaseRefAgain"
        ∧ goto(n, "DecreaseRef")
        ∧ objects′ = [
            objects EXCEPT ![addr].ref = @ + diff, ![addr].added = TRUE]
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED local_addr
        ∧ UNCHANGED last_counter

LoadPointer(n) ≜
    ∧ pc[n] = "Init" ∨ pc[n] = "LoadPointer"
    ∧ counter′ = counter + 1
    ∧ local_addr′ = [local_addr EXCEPT ![n] = pointer]
    ∧ goto(n, "IncreaseRef")
    ∧ UNCHANGED objects
    ∧ UNCHANGED pointer
    ∧ UNCHANGED last_counter

IncreaseRef(n) ≜
    LET
```

```
        addr ≜ local_addr[n]
    IN
        ∧ pc[n] = "IncreaseRef"
        ∧ objects' = [objects EXCEPT ![addr].ref = @ + 1]
        ∧ goto(n, "DecreaseLocalCounter")
        ∧ UNCHANGED local_addr
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED last_counter

DecreaseLocalCounter(n) ≜
    ∧ pc[n] = "DecreaseLocalCounter"
    ∧ IF pointer = local_addr[n]
        THEN
            ∧ counter' = counter − 1
            ∧ goto(n, "UseObject")
        ELSE
            ∧ UNCHANGED counter
            ∧ goto(n, "ClearExtraRef")
    ∧ UNCHANGED local_addr
    ∧ UNCHANGED objects
    ∧ UNCHANGED pointer
    ∧ UNCHANGED last_counter

ClearExtraRef(n) ≜
    LET
        addr ≜ local_addr[n]
    IN
        ∧ pc[n] = "ClearExtraRef"
        ∧ IF objects[addr].added
            THEN objects' = [
                objects EXCEPT ![addr].ref = @ − 1]
            ELSE objects' = [
                objects EXCEPT ![addr].extra = @ + 1]
        ∧ goto(n, "UseObject")
        ∧ UNCHANGED local_addr
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED last_counter

UseObject(n) ≜
    ∧ pc[n] = "UseObject"
    ∧ goto(n, "DecreaseRef")
    ∧ UNCHANGED objects
```

```
    ∧ UNCHANGED counter
    ∧ UNCHANGED pointer
    ∧ UNCHANGED local_addr
    ∧ UNCHANGED last_counter

DecreaseRef(n) ≜
    LET
        addr ≜ local_addr[n]
    IN
        ∧ pc[n] = "DecreaseRef"
        ∧ objects' = [objects EXCEPT ![addr].ref = @ − 1]
        ∧ IF objects'[addr].ref = 0
            THEN goto(n, "DestroyObject")
            ELSE goto(n, "Terminated")
        ∧ UNCHANGED local_addr
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED last_counter

DestroyObject(n) ≜
    LET
        addr ≜ local_addr[n]
    IN
        ∧ pc[n] = "DestroyObject"
        ∧ goto(n, "Terminated")
        ∧ objects' = [objects EXCEPT ![addr].destroyed = @ + 1]
        ∧ UNCHANGED local_addr
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED last_counter

TerminateCond ≜
    ∧ ∀ n ∈ Node : pc[n] = "Terminated"

Terminated ≜
    ∧ TerminateCond
    ∧ UNCHANGED vars

Next ≜
    ∨ ∃ n ∈ Node :
        ∨ AllocateNewObject(n)
        ∨ SwapPointer(n)
        ∨ IncreaseRefAgain(n)
        ∨ LoadPointer(n)
```

$\lor\ IncreaseRef(n)$
$\lor\ DecreaseLocalCounter(n)$
$\lor\ ClearExtraRef(n)$
$\lor\ UseObject(n)$
$\lor\ DecreaseRef(n)$
$\lor\ DestroyObject(n)$
$\lor\ Terminated$

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

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

$FullyDestroyed \triangleq$
LET
$destroyedExceptLast(addr) \triangleq$
$addr \neq pointer \Rightarrow objects[addr].destroyed = 1 \land objects[addr].ref = 0$
$allDestroyed \triangleq$
$\forall\, addr \in \text{DOMAIN } objects : destroyedExceptLast(addr)$
IN
$TerminateCond \Rightarrow allDestroyed$

$UseObjectAlwaysValid \triangleq$
LET
$getObj(n) \triangleq objects[local\_addr[n]]$
$notUseAfterFree(n) \triangleq$
$\land\ getObj(n).destroyed = 0$
$\land\ getObj(n).ref > 0$
IN
$\forall\, n \in Node : pc[n] =$ "UseObject" $\Rightarrow notUseAfterFree(n)$

$IncreaseRefMustNotDestroyed \triangleq$
LET
$accessStates(n) \triangleq pc[n] =$ "IncreaseRef"
$getObj(n) \triangleq objects[local\_addr[n]]$
IN
$\forall\, n \in Node : accessStates(n) \Rightarrow getObj(n).destroyed = 0$

$AccessStateMustNotDestroyed \triangleq$
LET
$accessStates(n) \triangleq$
$\lor\ pc[n] =$ "IncreaseRef"
$\lor\ pc[n] =$ "IncreaseRefAgain"
$\lor\ pc[n] =$ "DecreaseLocalCounter"
$\lor\ pc[n] =$ "ClearExtraRef"

$\lor\ pc[n] = \text{“UseObject”}$
$\lor\ pc[n] = \text{“DecreaseRef”}$
$\lor\ pc[n] = \text{“DestroyObject”}$

$getObj(n) \triangleq objects[local\_addr[n]]$

IN

$\forall\, n \in Node : accessStates(n) \Rightarrow getObj(n).destroyed = 0$

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$Sym \triangleq Permutations(Node)$