MODULE $AtomicPtrV2$

EXTENDS TLC, $Naturals$, $Sequences$

CONSTANTS $Node$, nil

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

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

ref is the global refcount
ignored is the ignored counter, increased with pumped flag is FALSE
pumped flag to signal that *store*() thread has already increase the global refcount

$Object \triangleq [ref : Nat, ignored : Nat, pumped : \text{BOOLEAN}, destroyed : Nat]$

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

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

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

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

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

$newObject \triangleq [ref \mapsto 1, ignored \mapsto 0, pumped \mapsto \text{FALSE}, destroyed \mapsto 0]$

$allocNew(n) \triangleq$

```
    ∧ objects′ = Append(objects, newObject)
    ∧ local_addr′ = [local_addr EXCEPT ![n] = Len(objects′)]

reuseObject(n) ≜
    ∃ addr ∈ DOMAIN objects :
        ∧ objects[addr].destroyed = 1
        ∧ objects′ = [objects EXCEPT ![addr] = newObject]
        ∧ local_addr′ = [local_addr EXCEPT ![n] = addr]

AllocateNewObject(n) ≜
    ∧ pc[n] = "Init"
    ∧ goto(n, "SwapPointer")
    ∧ ∨ allocNew(n)
      ∨ reuseObject(n)
    ∧ 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].ignored
    IN
        ∧ pc[n] = "IncreaseRefAgain"
        ∧ goto(n, "DecreaseRef")
        ∧ objects′ = [
            objects EXCEPT ![addr].ref = @ + diff, ![addr].pumped = TRUE]
        ∧ UNCHANGED counter
        ∧ UNCHANGED pointer
        ∧ UNCHANGED local_addr
```

```
        ∧ UNCHANGED last_counter

LoadPointer(n) ≜
    ∧ pc[n] = "Init"
    ∧ 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].pumped
```

```
            THEN objects' = [
                objects EXCEPT ![addr].ref = @ - 1]
            ELSE objects' = [
                objects EXCEPT ![addr].ignored = @ + 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
```

$TerminateCond \triangleq$
$\quad \land \forall\, n \in Node : pc[n] =$ "Terminated"

$Terminated \triangleq$
$\quad \land\ TerminateCond$
$\quad \land$ UNCHANGED $vars$

$Next \triangleq$
$\quad \lor \exists\, n \in Node :$
$\qquad \lor\ AllocateNewObject(n)$
$\qquad \lor\ SwapPointer(n)$
$\qquad \lor\ IncreaseRefAgain(n)$
$\qquad \lor\ LoadPointer(n)$
$\qquad \lor\ IncreaseRef(n)$
$\qquad \lor\ DecreaseLocalCounter(n)$
$\qquad \lor\ ClearExtraRef(n)$
$\qquad \lor\ UseObject(n)$
$\qquad \lor\ DecreaseRef(n)$
$\qquad \lor\ DestroyObject(n)$
$\quad \lor\ Terminated$

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

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

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

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

$IncreaseRefMustNotDestroyed \triangleq$

```
    LET
        accessStates(n) ≜ pc[n] = "IncreaseRef"
        getObj(n) ≜ objects[local_addr[n]]
    IN
        ∀ n ∈ Node : accessStates(n) ⇒ getObj(n).destroyed = 0

AccessStateMustNotDestroyed ≜
    LET
        accessStates(n) ≜
            ∨ pc[n] = "IncreaseRef"
            ∨ pc[n] = "IncreaseRefAgain"
            ∨ pc[n] = "DecreaseLocalCounter"
            ∨ pc[n] = "ClearExtraRef"
            ∨ pc[n] = "UseObject"
            ∨ pc[n] = "DecreaseRef"
            ∨ pc[n] = "DestroyObject"

        getObj(n) ≜ objects[local_addr[n]]
    IN
        ∀ n ∈ Node : accessStates(n) ⇒ getObj(n).destroyed = 0

AlwaysTerminate ≜ ◇TerminateCond

IncreaseRefLeadToUseObject ≜
    ∀ n ∈ Node :
      pc[n] = "IncreaseRef" ⇝ pc[n] = "UseObject"

Sym ≜ Permutations(Node)
```