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",
"IncreaseRef", "DecreaseLocalCounter", "ClearExtraRef",
"UseObject",
"DecreaseRef", "DestroyObject", "Terminated"$\}$

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

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

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

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

$allocNew(n) \triangleq$
$\land objects' = Append(objects, newObject)$
$\land local\_addr' = [local\_addr \text{ EXCEPT } ![n] = Len(objects')]$

```
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].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"
    ∧ 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"
```

$$
\begin{array}{l}
\mathit{Terminated} \triangleq \\
\quad \land \mathit{TerminateCond} \\
\quad \land \text{UNCHANGED } \mathit{vars}
\end{array}
$$

$$
\begin{array}{l}
\mathit{Next} \triangleq \\
\quad \lor \exists\, n \in \mathit{Node} : \\
\qquad \lor \mathit{AllocateNewObject}(n) \\
\qquad \lor \mathit{SwapPointer}(n) \\
\qquad \lor \mathit{IncreaseRefAgain}(n) \\
\qquad \lor \mathit{LoadPointer}(n) \\
\qquad \lor \mathit{IncreaseRef}(n) \\
\qquad \lor \mathit{DecreaseLocalCounter}(n) \\
\qquad \lor \mathit{ClearExtraRef}(n) \\
\qquad \lor \mathit{UseObject}(n) \\
\qquad \lor \mathit{DecreaseRef}(n) \\
\qquad \lor \mathit{DestroyObject}(n) \\
\quad \lor \mathit{Terminated}
\end{array}
$$

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

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

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

$$
\begin{array}{l}
\mathit{UseObjectAlwaysValid} \triangleq \\
\quad \text{LET} \\
\qquad \mathit{getObj}(n) \triangleq \mathit{objects}[\mathit{local\_addr}[n]] \\
\qquad \mathit{notUseAfterFree}(n) \triangleq \\
\qquad\quad \land \mathit{getObj}(n).\mathit{destroyed} = 0 \\
\qquad\quad \land \mathit{getObj}(n).\mathit{ref} > 0 \\
\quad \text{IN} \\
\qquad \forall\, n \in \mathit{Node} : \mathit{pc}[n] = \text{“UseObject”} \Rightarrow \mathit{notUseAfterFree}(n)
\end{array}
$$

$$
\begin{array}{l}
\mathit{IncreaseRefMustNotDestroyed} \triangleq \\
\quad \text{LET} \\
\qquad \mathit{accessStates}(n) \triangleq \mathit{pc}[n] = \text{“IncreaseRef”}
\end{array}
$$

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

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"
$\lor\ pc[n] =$ "UseObject"
$\lor\ pc[n] =$ "DecreaseRef"
$\lor\ pc[n] =$ "DestroyObject"

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

IN

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

$AlwaysTerminate \triangleq \Diamond TerminateCond$

$IncreaseRefLeadToUseObject \triangleq$

$\forall\, n \in Node :$
$pc[n] =$ "IncreaseRef" $\leadsto pc[n] =$ "UseObject"

$Sym \triangleq Permutations(Node)$