MODULE CC

TLA+ specification of Causal Consistency variants, including CC, CM, and CCv.

See the paper "On Verifying Causal Consistency" ($POPL$'2017).

EXTENDS $Naturals$, $Sequences$, $FiniteSets$, $Functions$, $FiniteSetsExt$, $RelationUtils$, TLC

$Key \triangleq Range(\text{"abcdefghijklmnopqrstuvwxyz"})$ We assume single-character keys.
$Val \triangleq Nat$ We assume values from Nat.
$InitVal \triangleq 0$ We follow the convention in $POPL$'2017.
$Oid \triangleq Nat$ We assume operation identifiers from Nat.

$Operation \triangleq [type : \{\text{"read"}, \text{"write"}\}, key : Key, val : Val, oid : Oid]$
$R(k, v, oid) \triangleq [type \mapsto \text{"read"}, key \mapsto k, val \mapsto v, oid \mapsto oid]$
$W(k, v, oid) \triangleq [type \mapsto \text{"write"}, key \mapsto k, val \mapsto v, oid \mapsto oid]$

$Session \triangleq Seq(Operation)$ A session $s \in Session$ is a sequence of operations.
$History \triangleq \text{SUBSET } Session$ A history $h \in History$ is a set of sessions.

Utilities.

$Ops(h) \triangleq$ Return the set of all operations in history $h \in History$.
  UNION $\{Range(s) : s \in h\}$

$ReadOps(h) \triangleq$ Return the set of all read operations in history $h \in History$.
  $\{op \in Ops(h) : op.type = \text{"read"}\}$

$ReadOpsOnKey(h, k) \triangleq$ Return the set of all read operations on key $k \in Key$ in history $h \in History$.
  $\{op \in Ops(h) : op.type = \text{"read"} \land op.key = k\}$

$WriteOps(h) \triangleq$ Return the set of all write operations in history $h \in History$.
  $\{op \in Ops(h) : op.type = \text{"write"}\}$

$WriteOpsOnKey(h, k) \triangleq$ Return the set of all write operations on key $k \in Key$ in history $h \in History$
  $\{op \in Ops(h) : op.type = \text{"write"} \land op.key = k\}$

Well-formedness of history $h \in History$:

– $TODO$: type invariants
- uniqueness of oids

$WellFormed(h) \triangleq$
  $\land h \in History$
  $\land Cardinality(Ops(h)) = ReduceSet(\text{LAMBDA } s, x : Len(s) + x, h, 0)$

Auxiliary definitions for the axioms used in the definitions of causal consistency

The program order of $h \in History$ is a union of total orders among operations in the same session
$PO(h) \triangleq \text{UNION } \{Seq2Rel(s) : s \in h\}$

The set of operations that preceed $o \in Operation$ in program order in history $h \in History$

```
POPast(h, o) ≜ InverseImage(PO(h), o)

 The set of operations that preceed o ∈ Operation in causal order co
CausalPast(co, o) ≜ InverseImage(co, o)

 The restriction of causal order co to the operations in the causal past of operation o ∈ Operation
CausalHist(co, o) ≜ co | CausalPast(co, o)

 The restriction of arbitration arb to the operations in the causal past of operation o ∈ Operation
CausalArb(co, arb, o) ≜ arb | CausalPast(co, o)
```

Axioms used in the defintions of causal consistency

```
RWRegSemantics(seq, o) ≜   Is o ∈ Operation legal when it is appended to seq
    IF o.type = "write" THEN TRUE
     ELSE LET wseq ≜ SelectSeq(seq, LAMBDA op : op.type = "write" ∧ op.key = o.key)
          IN  IF wseq = ⟨⟩ THEN o.val = InitVal
              ELSE  o.val = wseq[Len(wseq)].val

AxCausalValue(co, o) ≜
    LET seqs ≜ AllLinearExtensions(CausalHist(co, o), CausalPast(co, o))
    IN  TRUE ∈ {RWRegSemantics(seq, o) : seq ∈ seqs}   TODO: shortcut implementation of anyTrue for efficiency

AxCausalArb(co, arb, o) ≜
    LET seq ≜ AnyLinearExtension(CausalArb(co, arb, o), CausalPast(co, o))  it is unique
    IN  RWRegSemantics(seq, o)
```

Specification of *CC*

```
CC(h) ≜   Check whether h ∈ History satisfies CC (Causal Consistency)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :   TODO: to generate (given a chain decomposition)
            ∧ Respect(co, PO(h))                    AxCausal
            ∧ IsStrictPartialOrder(co, ops)
            ∧ PrintT("co: " ∘ ToString(co))
            ∧ ∀ o ∈ ops : AxCausalValue(co, o)      AxCausalValue
```

Specification of *CCv*

To generate possible ordering relations, not to enumerate and test them

```
CCv(h) ≜   Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :   TODO: to generate (given a chain decomposition)
            ∧ Respect(co, PO(h))                    AxCausal
            ∧ IsStrictPartialOrder(co, ops)
            ∧ PrintT("co: " ∘ ToString(co))
            ∧ ∃ arb ∈ {Seq2Rel(le) : le ∈ AllLinearExtensions(co, ops)} :   AxArb
                ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)   AxCausalArb
```

```
                    ∧ PrintT("arb: " ∘ ToString(arb))
  Version 2: re-arrange clauses
CCv2(h) ≜  Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :  FIXME: efficiency!!!
            ∧ Respect(co, PO(h))  AxCausal
            ∧ IsStrictPartialOrder(co, ops)
            ∧ PrintT("co: " ∘ ToString(co))
            ∧ ∃ arb ∈ SUBSET (ops × ops) :  to generate; not to test
                ∧ Respect(arb, co)                AxArb
                ∧ IsStrictTotalOrder(arb, ops)
                ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)  AxCausalArb
                ∧ PrintT("arb: " ∘ ToString(arb))
  Version 1: Following the definition of POPL2017
CCv1(h) ≜  Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :  FIXME: efficiency!!!
            ∧ ∃ arb ∈ SUBSET (ops × ops) :
                ∧ PrintT("co: " ∘ ToString(co))
                ∧ PrintT("arb: " ∘ ToString(arb))
                ∧ IsStrictPartialOrder(co, ops)
                ∧ IsStrictTotalOrder(arb, ops)
                ∧ Respect(co, PO(h))          AxCausal
                ∧ Respect(arb, co)                AxArb
                ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)  AxCausalArb
├──────────────────────────────────────────────────────────────────────┤
  Specification of CM
CM(h) ≜  Check whether h ∈ History satisfies CM (Causal Memory)
    FALSE  TODO
├──────────────────────────────────────────────────────────────────────┤
  Auxiliary operators used in the checking algorithms: We consider only differentiated histories.
KeyOf(h) ≜  the set of keys read or written in h ∈ History
  {op.key : op ∈ Ops(h)}

IsDifferentiated(h) ≜  Is h ∈ History differentiated?
  ∀ k ∈ KeyOf(h) :
    LET writes ≜ WriteOpsOnKey(h, k)
    IN  ∀ w1 ∈ writes, w2 ∈ writes :
            ∧ w1.val ≠ w2.val
            ∧ w1.val ≠ InitVal
  Auxiliary relations used in the checking algorithms
RF(h) ≜  the read-from relation
    {⟨w, r⟩ ∈ WriteOps(h) × ReadOps(h) : w.key = r.key ∧ w.val = r.val}
```

```
CO(h) ≜ the CO order defined as the transitive closure of the union of PO(h) and RF(h)
    TC(PO(h) ∪ RF(h))

All bad patterns defined in POPL'2017 (see Table 2 of POPL'2017)

CyclicCO(h) ≜ Cyclic(PO(h) ∪ RF(h))

WriteCOInitRead(h) ≜
  ∃ k ∈ KeyOf(h) :
    ∃ r ∈ ReadOpsOnKey(h, k), w ∈ WriteOpsOnKey(h, k) :
      ∧ ⟨w, r⟩ ∈ CO(h)  TODO: for efficiency
      ∧ r.val = InitVal

ThinAirRead(h) ≜
  ∃ k ∈ KeyOf(h) :
    ∃ r ∈ ReadOpsOnKey(h, k) :
      ∧ r.val ≠ InitVal
      ∧ ∀ w ∈ WriteOpsOnKey(h, k) : ⟨w, r⟩ ∉ RF(h)

WriteCORead(h) ≜
  ∃ k ∈ KeyOf(h) :
    ∃ w1, w2 ∈ WriteOpsOnKey(h, k), r1 ∈ ReadOpsOnKey(h, k) :
      ∧ ⟨w1, w2⟩ ∈ CO(h)
      ∧ ⟨w2, r1⟩ ∈ CO(h)  TODO: efficiency
      ∧ ⟨w1, r1⟩ ∈ RF(h)

CyclicHB(h) ≜  TODO:
  FALSE

CyclicCF(h) ≜  TODO:
  FALSE
  Cyclic(CF(h) ∪ CO(h))

Checking algorithms of POPL'2017 (see Table 3 of POPL'2017)

CCAlg(h) ≜  Checking algorithm for CC (Causal Consistency)
  ∧ CyclicCO(h)
  ∧ WriteCOInitRead(h)
  ∧ ThinAirRead(h)
  ∧ WriteCORead(h)

\* Modification Historyy
\* Last modified Tue Apr 20 13:26:56 CST 2021 by hengxin
\* Created Tue Apr 01 10:24:07 CST 2021 by hengxin
```