```
┌──────────────────────────── 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 ≜ Range("abcdefghijklmnopqrstuvwxyz")  We assume single-character keys.
Val  ≜ Nat          We assume values from Nat.
InitVal ≜ 0         We follow the convention in POPL'2017.
Oid ≜ Nat           We assume operation identifiers from Nat.

Operation ≜ [type : {"read", "write"}, key : Key, val : Val, oid : Oid]
R(k, v, oid)  ≜ [type ↦ "read", key ↦ k, val ↦ v, oid ↦ oid]
W(k, v, oid)  ≜ [type ↦ "write", key ↦ k, val ↦ v, oid ↦ oid]

Session ≜ Seq(Operation)  A session s ∈ Session is a sequence of operations.
History ≜ SUBSET Session  A history h ∈ History is a set of sessions.
├────────────────────────────────────────────────────────────────────┤
  Utility operators for operations.

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

ReadOps(h) ≜   Return the set of all read operations in history h ∈ History.
    {op ∈ Ops(h) : op.type = "read"}

ReadOpsOnKey(h, k) ≜   Return the set of all read operations on key k ∈ Key in history h ∈ History.
    {op ∈ Ops(h) : op.type = "read" ∧ op.key = k}

WriteOps(h) ≜   Return the set of all write operations in history h ∈ History.
    {op ∈ Ops(h) : op.type = "write"}

WriteOpsOnKey(h, k) ≜   Return the set of all write operations on key k ∈ Key in history h ∈ History
    {op ∈ Ops(h) : op.type = "write" ∧ op.key = k}
├────────────────────────────────────────────────────────────────────┤
  Well-formedness of history h ∈ History:

  − TODO: type invariants
  - uniqueness of oids

WellFormed(h) ≜
   ∧ h ∈ History
   ∧ Cardinality(Ops(h)) = ReduceSet(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 ∈ History is a union of total orders among operations in the same session
PO(h) ≜ UNION {Seq2Rel(s) : s ∈ h}

  The set of operations that preceed o ∈ Operation in program order in history h ∈ 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
```

```
                    $\land\ PrintT(\text{“arb: ”} \circ ToString(arb))$
Version 2: re-arrange clauses
$CCv2(h) \triangleq$  Check whether $h \in History$ satisfies $CCv$ (Causal Convergence)
    LET $ops \triangleq Ops(h)$
    IN  $\exists\, co \in$ SUBSET $(ops \times ops)$ :  $FIXME$: efficiency!!!
            $\land\ Respect(co, PO(h))$  $AxCausal$
            $\land\ IsStrictPartialOrder(co, ops)$
            $\land\ PrintT(\text{“co: ”} \circ ToString(co))$
            $\land\ \exists\, arb \in$ SUBSET $(ops \times ops)$ :  to generate; not to test
                $\land\ Respect(arb, co)$                $AxArb$
                $\land\ IsStrictTotalOrder(arb, ops)$
                $\land\ \forall\, o \in ops : AxCausalArb(co, arb, o)$  $AxCausalArb$
                $\land\ PrintT(\text{“arb: ”} \circ ToString(arb))$
Version 1: Following the definition of $POPL2017$
$CCv1(h) \triangleq$  Check whether $h \in History$ satisfies $CCv$ (Causal Convergence)
    LET $ops \triangleq Ops(h)$
    IN  $\exists\, co \in$ SUBSET $(ops \times ops)$ :  $FIXME$: efficiency!!!
            $\land\ \exists\, arb \in$ SUBSET $(ops \times ops)$ :
                $\land\ PrintT(\text{“co: ”} \circ ToString(co))$
                $\land\ PrintT(\text{“arb: ”} \circ ToString(arb))$
                $\land\ IsStrictPartialOrder(co, ops)$
                $\land\ IsStrictTotalOrder(arb, ops)$
                $\land\ Respect(co, PO(h))$          $AxCausal$
                $\land\ Respect(arb, co)$                $AxArb$
                $\land\ \forall\, o \in ops : AxCausalArb(co, arb, o)$  $AxCausalArb$
---------------------------------------------------------------------------
Specification of $CM$
$CM(h) \triangleq$  Check whether $h \in History$ satisfies $CM$ (Causal Memory)
    FALSE  $TODO$
---------------------------------------------------------------------------
Auxiliary operators used in the checking algorithms: We consider only differentiated histories.
$KeyOf(h) \triangleq$  the set of keys read or written in $h \in History$
    $\{op.key : op \in Ops(h)\}$

$IsDifferentiated(h) \triangleq$  Is $h \in History$ differentiated?
    $\forall\, k \in KeyOf(h)$ :
        LET $writes \triangleq WriteOpsOnKey(h, k)$
        IN  $\forall\, w1 \in writes, w2 \in writes$ :
                $\land\ w1.val \neq w2.val$
                $\land\ w1.val \neq InitVal$
Auxiliary relations used in the checking algorithms
$RF(h) \triangleq$  the read-from relation $TODO$: using infix symbolic operator???
    $\{\langle w, r \rangle \in WriteOps(h) \times ReadOps(h) : w.key = r.key \land w.val = r.val\}$
```

```
$CO(h) \triangleq$  the $CO$ order defined as the transitive closure of the union of $PO(h)$ and $RF(h)$
        $TC(PO(h) \cup RF(h))$

$CF(h) \triangleq$  the conflict relation
        LET $co \triangleq CO(h)$
            $rf \triangleq RF(h)$
            $reads \triangleq ReadOps(h)$
            $writes \triangleq WriteOps(h)$
        IN  $\{\langle w1, w2 \rangle \in writes \times writes :$
                $\land\ w1.key = w2.key$
                $\land\ w1.val \neq w2.val$
                $\land\ \exists\, r \in reads : \langle w1, r \rangle \in co \land \langle w2, r \rangle \in rf\}$

    All bad patterns defined in $POPL$'2017 (see Table 2 of $POPL$'2017)
$CyclicCO(h) \triangleq Cyclic(PO(h) \cup RF(h))$

$WriteCOInitRead(h) \triangleq$
    $\exists\, k \in KeyOf(h) :$
        $\exists\, r \in ReadOpsOnKey(h, k),\ w \in WriteOpsOnKey(h, k) :$
            $\land\ \langle w, r \rangle \in CO(h)$  $TODO$: for efficiency
            $\land\ r.val = InitVal$

$ThinAirRead(h) \triangleq$
    $\exists\, k \in KeyOf(h) :$
        $\exists\, r \in ReadOpsOnKey(h, k) :$
            $\land\ r.val \neq InitVal$
            $\land\ \forall\, w \in WriteOpsOnKey(h, k) : \langle w, r \rangle \notin RF(h)$

$WriteCORead(h) \triangleq$
    $\exists\, k \in KeyOf(h) :$
        $\exists\, w1, w2 \in WriteOpsOnKey(h, k),\ r1 \in ReadOpsOnKey(h, k) :$
            $\land\ \langle w1, w2 \rangle \in CO(h)$
            $\land\ \langle w2, r1 \rangle \in CO(h)$  $TODO$: efficiency
            $\land\ \langle w1, r1 \rangle \in RF(h)$

$CyclicCF(h) \triangleq$
    $Cyclic(CF(h) \cup CO(h))$

$WriteHBInitRead(h) \triangleq$  $TODO$:
    FALSE

$CyclicHB(h) \triangleq$  $TODO$:
    FALSE
    Checking algorithms of $POPL$'2017 (see Table 3 of $POPL$'2017)
$CCAlg(h) \triangleq$  Checking algorithm for $CC$ (Causal Consistency)
    $\land\ \neg CyclicCO(h)$
    $\land\ \neg WriteCOInitRead(h)$
    $\land \neg ThinAirRead(h)$
    $\land \neg WriteCORead(h)$

$CCvAlg(h) \triangleq$  Checking algorithm for $CCv$ (Causal Convergence)
    $\land \neg CyclicCO(h)$
    $\land \neg WriteCOInitRead(h)$
    $\land \neg ThinAirRead(h)$
    $\land \neg WriteCORead(h)$
    $\land \neg CyclicCF(h)$

$CMAlg(h) \triangleq$  $TODO$: Checking algorithm for $CM$ (Causal Memory)
    FALSE
```

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