```
1  ---------- MODULE CC ----------
   TLA+ specification of Causal Consistency variants, including CC, CM, and CCv.
   See the paper "On Verifying Causal Consistency" (POPL'2017).
8  EXTENDS Naturals, Sequences, FiniteSets, Functions, FiniteSetsExt,
9          RelationUtils, TLC

11 CONSTANTS Keys, Vals
12 InitVal ≜ 0   we follow the convention in POPL'2017

14  oid: unique operation identifier
15 Operation ≜ [type : {"read", "write"}, key : Keys, val : Vals, oid : Nat]
16 R(k, v, oid) ≜ [type ↦ "read", key ↦ k, val ↦ v, oid ↦ oid]
17 W(k, v, oid) ≜ [type ↦ "write", key ↦ k, val ↦ v, oid ↦ oid]

19 Session ≜ Seq(Operation)   A session s ∈ Session is a sequence of operations.
20 History ≜ SUBSET Session   A history h ∈ History is a set of sessions.
21 ----------
   Utilities.
25 Ops(h) ≜   Return the set of all operations in history h ∈ History.
26     UNION {Range(s) : s ∈ h}

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

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

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

37 WriteOpsOnKey(h, k) ≜   Return the set of all write operations on key k ∈ Keys in history h ∈ History
38     {op ∈ Ops(h) : op.type = "write" ∧ op.key = k}
39 ----------
   Well-formedness of history h ∈ History:
   - TODO: type invariants
   - uniqueness of oids
46 WellFormed(h) ≜
47  ∧ h ∈ History
48     ∧ Cardinality(Ops(h)) = ReduceSet(LAMBDA s, x : Len(s) + x, h, 0)
49 ----------
   Auxiliary definitions for the axioms used in the definitions of causal consistency
53  The program order of h ∈ History is a union of total orders among operations in the same session
54 PO(h) ≜ UNION {Seq2Rel(s) : s ∈ h}

56  The set of operations that preceed o ∈ Operation in program order in history h ∈ History
57 POPast(h, o) ≜ InverseImage(PO(h), o)
```

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

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

65  The restriction of arbitration arb to the operations in the causal past of operation o ∈ Operation
66  CausalArb(co, arb, o) ≜ arb | CausalPast(co, o)
67 ├──────────────────────────────────────────────────────────────────────────┤
    Axioms used in the defintions of causal consistency
71  RWRegSemantics(seq, o) ≜  Is o ∈ Operation legal when it is appended to seq
72      IF o.type = "write" THEN TRUE
73       ELSE LET wseq ≜ SelectSeq(seq, LAMBDA op : op.type = "write" ∧ op.key = o.key)
74            IN  IF wseq = ⟨⟩ THEN o.val = InitVal
75                ELSE  o.val = wseq[Len(wseq)].val

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

81  AxCausalArb(co, arb, o) ≜
82      LET seq ≜ AnyLinearExtension(CausalArb(co, arb, o), CausalPast(co, o))  it is unique
83      IN  RWRegSemantics(seq, o)
84 ├──────────────────────────────────────────────────────────────────────────┤
    Specification of CC
88  CC(h) ≜  Check whether h ∈ History satisfies CC (Causal Consistency)
89       LET ops ≜ Ops(h)
90       IN  ∃ co ∈ SUBSET (ops × ops) :  TODO: to generate (given a chain decomposition)
91               ∧ Respect(co, PO(h))                AxCausal
92               ∧ IsStrictPartialOrder(co, ops)
93               ∧ PrintT("co: " ∘ ToString(co))
94               ∧ ∀ o ∈ ops : AxCausalValue(co, o)          AxCausalValue
95 ├──────────────────────────────────────────────────────────────────────────┤
    Specification of CCv


    To generate possible ordering relations, not to enumerate and test them
103 CCv(h) ≜  Check whether h ∈ History satisfies CCv (Causal Convergence)
104      LET ops ≜ Ops(h)
105      IN  ∃ co ∈ SUBSET (ops × ops) :  TODO: to generate (given a chain decomposition)
106              ∧ Respect(co, PO(h))                AxCausal
107              ∧ IsStrictPartialOrder(co, ops)
108              ∧ PrintT("co: " ∘ ToString(co))
109              ∧ ∃ arb ∈ {Seq2Rel(le) : le ∈ AllLinearExtensions(co, ops)} :  AxArb
110                    ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)  AxCausalArb
111                    ∧ PrintT("arb: " ∘ 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 $\exists\, co \in \text{SUBSET } (ops \times ops) :$ *FIXME*: efficiency!!!
    $\wedge\, Respect(co,\, PO(h))$ *AxCausal*
    $\wedge\, IsStrictPartialOrder(co,\, ops)$
    $\wedge\, PrintT(\text{“co: ”} \circ ToString(co))$
    $\wedge \exists\, arb \in \text{SUBSET } (ops \times ops) :$ to generate; not to test
      $\wedge\, Respect(arb,\, co)$ *AxArb*
      $\wedge\, IsStrictTotalOrder(arb,\, ops)$
      $\wedge \forall\, o \in ops : AxCausalArb(co,\, arb,\, o)$ *AxCausalArb*
      $\wedge\, PrintT(\text{“arb: ”} \circ ToString(arb))$

Version 1: Following the definition of *POPL*2017

$CCv1(h) \triangleq$ Check whether $h \in$ *History* satisfies *CCv* (Causal Convergence)
  LET $ops \triangleq Ops(h)$
  IN $\exists\, co \in \text{SUBSET } (ops \times ops) :$ *FIXME*: efficiency!!!
    $\wedge \exists\, arb \in \text{SUBSET } (ops \times ops) :$
      $\wedge\, PrintT(\text{“co: ”} \circ ToString(co))$
      $\wedge\, PrintT(\text{“arb: ”} \circ ToString(arb))$
      $\wedge\, IsStrictPartialOrder(co,\, ops)$
      $\wedge\, IsStrictTotalOrder(arb,\, ops)$
      $\wedge\, Respect(co,\, PO(h))$ *AxCausal*
      $\wedge\, Respect(arb,\, co)$ *AxArb*
      $\wedge \forall\, o \in ops : AxCausalArb(co,\, arb,\, o)$ *AxCausalArb*

Specification of *CM*

$CM(h) \triangleq$ Check whether $h \in$ *History* satisfies *CM* (Causal Memory)
  FALSE *TODO*

The checking algorithms in *POPL*'2017.

$IsDifferentiated(h) \triangleq$ Is $h \in$ *History* differentiated?
  $\forall\, k \in Keys :$
    LET $writes \triangleq WriteOpsOnKey(h,\, k)$
    IN $\forall\, w1 \in writes,\, w2 \in writes :$
      $\wedge\, w1.val \neq w2.val$
      $\wedge\, w1.val \neq InitVal$

$RF(h) \triangleq$ the read-from relation
  $\{\langle w,\, r\rangle \in WriteOps(h) \times ReadOps(h) : w.key = r.key \wedge w.val = r.val\}$

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

All bad patterns defined in *POPL*'2017

TODO: to implement $Cyclic(R)$ in *RelationUtils.tla*

$CyclicCO(h) \triangleq$ FALSE

$Cyclic(PO(h) \cup RF(h))$

$WriteCOInitRead(h) \triangleq$

$\exists\, k \in Keys :$

$\exists\, r \in ReadOpsOnKey(h,\, k),\, w \in WriteOpsOnKey(h,\, k) :$

$\land \langle w,\, r \rangle \in CO(h)$ *TODO*: for efficiency

$\land\, r.val = InitVal$

$ThinAirRead(h) \triangleq$

$\exists\, k \in Keys :$

$\exists\, r \in ReadOpsOnKey(h,\, k) :$

$\land\, r.val \neq InitVal$

$\land \forall\, w \in WriteOpsOnKey(h,\, k) : \langle w,\, r \rangle \notin RF(h)$

$WriteCORead(h) \triangleq$

$\exists\, k \in Keys :$

$\exists\, w1,\, w2 \in WriteOpsOnKey(h,\, k),\, r1 \in ReadOpsOnKey(h,\, k) :$

$\land \langle w1,\, w2 \rangle \in CO(h)$

$\land \langle w2,\, r1 \rangle \in CO(h)$ *TODO*: efficiency

$\land \langle w1,\, r1 \rangle \in RF(h)$

$CyclicHB(h) \triangleq$ *TODO*:

FALSE

$CyclicCF(h) \triangleq$ *TODO*:

FALSE

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

* Modification *History*

* Last modified *Mon Apr* 19 16:38:28 *CST* 2021 by *hengxin*

* Created *Tue Apr* 01 10:24:07 *CST* 2021 by *hengxin*