```
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, PartialOrderExt

11 Key ≜ Range("abcdefghijklmnopqrstuvwxyz")  We assume single-character keys.
12 Val ≜ Nat      We assume values from Nat.
13 InitVal ≜ 0    We follow the convention in POPL'2017.
14 Oid ≜ Nat      We assume operation identifiers from Nat.

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

20 Session ≜ Seq(Operation)  A session s ∈ Session is a sequence of operations.
21 History ≜ SUBSET Session  A history h ∈ History is a set of sessions.
22 ----------
   Utility operators for operations.
26 Ops(h) ≜  Return the set of all operations in history h ∈ History.
27     UNION {Range(s) : s ∈ h}

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

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

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

38 WriteOpsOnKey(h, k) ≜  Return the set of all write operations on key k ∈ Key in history h ∈ History
39     {op ∈ Ops(h) : op.type = "write" ∧ op.key = k}
40 ----------
   Well-formedness of history h ∈ History:
   - TODO: type invariants
   - uniqueness of oids
47 WellFormed(h) ≜
48   ∧ h ∈ History
49     ∧ LET ops ≜ Ops(h)
50           nops ≜ Cardinality(ops)
51           oids ≜ {o.oid : o ∈ ops}
52       IN  ∧ ∀ op ∈ ops :  Type invariants
53             ∨ op.type = "write"
54             ∨ op.type = "read"
55           ∧ nops = Cardinality(oids)  Uniqueness of oids
```

```
56                  ∧ nops = ReduceSet(LAMBDA s, x : Len(s) + x, h, 0)
57 ├──────────────────────────────────────────────────────────────────────┤
   Auxiliary definitions for the axioms used in the definitions of causal consistency
61  The program order of h ∈ History is a union of total orders among operations in the same session
62 PO(h) ≜ UNION {Seq2Rel(s) : s ∈ h}

64  The set of operations that preceed o ∈ Operation in program order in history h ∈ History
65 StrictPOPast(h, o) ≜ InverseImage(PO(h), o)
66 POPast(h, o) ≜ StrictPOPast(h, o) ∪ {o}  Original definition in paper, including itself


69  The set of operations that preceed o ∈ Operation in causal order co
70 StrictCausalPast(co, o) ≜ InverseImage(co, o)
71 CausalPast(co, o) ≜ StrictCausalPast(co, o) ∪ {o}  Original definition in paper, including itself

73  The restriction of causal order co to the operations in the causal past of operation o ∈ Operation
74 StrictCausalHist(co, o) ≜ co | StrictCausalPast(co, o)
75 CausalHist(co, o) ≜ co | CausalPast(co, o)  Original definition in paper

77  The restriction of arbitration arb to the operations in the causal past of operation o ∈ Operation
78 StrictCausalArb(co, arb, o) ≜ arb | StrictCausalPast(co, o)
79 CausalArb(co, arb, o) ≜ arb | CausalPast(co, o)  Original definition in paper
80 ├──────────────────────────────────────────────────────────────────────┤
   Axioms used in the defintions of causal consistency
84 RWRegSemantics(seq, o) ≜  Is o ∈ Operation legal when it is appended to seq
85     IF o.type = "write" THEN TRUE ELSE
86     LET wseq ≜ SelectSeq(seq, LAMBDA op : op.type = "write" ∧ op.key = o.key)
87         IN  IF wseq = ⟨⟩ THEN o.val = InitVal
88             ELSE  o.val = wseq[Len(wseq)].val

90 PreSeq(seq, o) ≜  All of the operations before o in sequence seq
91     LET so ≜ Seq2Rel(seq)
92     IN  SelectSeq(seq, LAMBDA op : ⟨op, o⟩ ∈ so)

94 RWRegSemanticsOperations(seq, ops) ≜  For ops ⊆ Range(seq), is ∀ o ∈ ops legal
95     ∀ o ∈ ops :
96         LET preSeq ≜ PreSeq(seq, o)
97         IN  RWRegSemantics(preSeq, o)

99 AxCausalValue(co, o) ≜
100    LET seqs ≜ AllLinearExtensions(StrictCausalHist(co, o), StrictCausalPast(co, o))
101    IN  ∃ seq ∈ seqs : RWRegSemantics(seq, o)

103 AxCausalSeq(h, co, o) ≜
104    LET popast ≜ POPast(h, o)
105        seqs ≜ AllLinearExtensions(CausalHist(co, o), CausalPast(co, o))
106    IN  ∃ seq ∈ seqs : RWRegSemanticsOperations(seq, popast)
```

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

 Directory to store files recording strict partial order relations
 POFilePath ≜ "E:\\Programs\\Python-Programs\\Event-Structure-Enumerator\\POFile\\"
POFilePath ≜ "D:\\Education\\Programs\\Python\\EnumeratePO\\POFile\\"

 A set of all subset of the Cartesian Product of ops × ops,
 each of which represent a strict partial order(irreflexive and transitive)
StrictPartialOrderSubset(ops) ≜
    PartialOrderSubset(ops, POFilePath)

StrictPartialOrderSubsetNo(ops, i) ≜
    PartialOrderSubsetNoPart(ops, POFilePath, i)

Parts ≜ {0, 1, 2, 3, 4, 5, 6}
```

---

```
Specification of CC

Final Version: Enumerate all possible strict partial order subsets
CC(h) ≜  Check whether h ∈ History satisfies CC (Causal Consistency)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ StrictPartialOrderSubset(ops) :  Optimized implementation
            ∧ Respect(co, PO(h))                AxCausal
        ∧ PrintT("co : " ∘ ToString(co))
            ∧ ∀ o ∈ ops : AxCausalValue(co, o)   AxCausalValue

BigCC(h) ≜
    LET ops ≜ Ops(h)
     IN  ∧ Cardinality(Ops(h)) = 7
         ∧ ∃ part ∈ Parts :
            ∃ co ∈ StrictPartialOrderSubsetNo(ops, part) :  Optimized implementation
                ∧ Respect(co, PO(h))                AxCausal
                ∧ PrintT("co: " ∘ ToString(co))
                ∧ ∀ o ∈ ops : AxCausalValue(co, o)   AxCausalValue

Version 1: Following the definition of POPL2017
CC1(h) ≜  Check whether h ∈ History satisfies CC (Causal Consistency)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :  Raw implementation: Cartesian Product
            ∧ Respect(co, PO(h))                AxCausal
            ∧ IsStrictPartialOrder(co, ops)
             ∧ PrintT("co : " ∘ ToString(co))
            ∧ ∀ o ∈ ops : AxCausalValue(co, o)   AxCausalValue
```

```
Specification of CCv

Final Version: Enumerate all possible strict partial order subsets
CCv(h) ≜ Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ StrictPartialOrderSubset(ops) : Optimized implementation
            ∧ Respect(co, PO(h))                AxCausal
            ∧ PrintT("co : " ∘ ToString(co))
            ∧ ∃ arb ∈ {Seq2Rel(le) : le ∈ AllLinearExtensions(co, ops)} : AxArb
                ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)   AxCausalArb
                ∧ PrintT("arb : " ∘ ToString(arb))

BigCCv(h) ≜
    LET ops ≜ Ops(h)
    IN  ∧ Cardinality(Ops(h)) = 7
        ∧ ∃ part ∈ Parts :
            ∃ co ∈ StrictPartialOrderSubsetNo(ops, part) : Optimized implementation
                ∧ Respect(co, PO(h))                AxCausal
                ∧ PrintT("co: " ∘ ToString(co))
                ∧ ∃ arb ∈ {Seq2Rel(le) : le ∈ AllLinearExtensions(co, ops)} : AxArb
                    ∧ ∀ o ∈ ops : AxCausalArb(co, arb, o)   AxCausalArb
                    ∧ PrintT("arb: " ∘ ToString(arb))

Version 3: If exists, arbitration order is one of the linear extensions of co on the set ops
CCv3(h) ≜ Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) : Raw implementation: Cartesian Product
            ∧ 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) :
            ∧ 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 *POPL*2017

```
CCv1(h) ≜  Check whether h ∈ History satisfies CCv (Causal Convergence)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :
            ∧ ∃ 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*

Final Version: Enumerate all possible strict partial order subsets

```
CM(h) ≜  Check whether h ∈ History satisfies CM (Causal Memory)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ StrictPartialOrderSubset(ops) :
          ∧ Respect(co, PO(h))                AxCausal
          ∧ ∀ o ∈ ops : AxCausalSeq(h, co, o)   AxCausalSeq

BigCM(h) ≜
    LET ops ≜ Ops(h)
    IN  ∧ Cardinality(Ops(h)) = 7
        ∧ ∃ part ∈ Parts :
            ∃ co ∈ StrictPartialOrderSubsetNo(ops, part) :  Optimized implementation
              ∧ Respect(co, PO(h))                AxCausal
              ∧ ∀ o ∈ ops : AxCausalSeq(h, co, o)   AxCausalSeq
```

Version 1: Following the definition of *POPL*2017

```
CM1(h) ≜  Check whether h ∈ History satisfies CM (Causal Memory)
    LET ops ≜ Ops(h)
    IN  ∃ co ∈ SUBSET (ops × ops) :
          ∧ IsStrictPartialOrder(co, ops)
          ∧ Respect(co, PO(h))                AxCausal
          ∧ ∀ o ∈ ops : AxCausalSeq(h, co, o)   AxCausalSeq
```

---

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