MODULE *CCTest*

Test of *CC* Module

EXTENDS *CC*

Test case: The following histories are from Figure 2 of the *POPL*'2017 paper.

Naming Conventions:

– *ha*: history of Figure 2(a)
– *hasa*: session a of history ha

TODO:

- to add more test cases
- to automatically generate test cases that do or do not satisfy the specs

  - consider Section 3.2 of *POPL*'2017
  – *ref*: the *MonkeyDB* paper

$hasa \triangleq \langle W(\text{“x”}, 1, 1), R(\text{“x”}, 2, 2)\rangle$
$hasb \triangleq \langle W(\text{“x”}, 2, 3), R(\text{“x”}, 1, 4)\rangle$
$ha \triangleq \{hasa, hasb\}$ *CM* but not *CCv*

$hbsa \triangleq \langle W(\text{“z”}, 1, 1), W(\text{“x”}, 1, 2), W(\text{“y”}, 1, 3)\rangle$
$hbsb \triangleq \langle W(\text{“x”}, 2, 4), R(\text{“z”}, 0, 5), R(\text{“y”}, 1, 6), R(\text{“x”}, 2, 7)\rangle$
$hb \triangleq \{hbsa, hbsb\}$ *CCv* but not *CM*

$hcsa \triangleq \langle W(\text{“x”}, 1, 1)\rangle$
$hcsb \triangleq \langle W(\text{“x”}, 2, 2), R(\text{“x”}, 1, 3), R(\text{“x”}, 2, 4)\rangle$
$hc \triangleq \{hcsa, hcsb\}$ *CC* but not *CM* nor *CCv*

$hdsa \triangleq \langle W(\text{“x”}, 1, 1), R(\text{“y”}, 0, 2), W(\text{“y”}, 1, 3), R(\text{“x”}, 1, 4)\rangle$
$hdsb \triangleq \langle W(\text{“x”}, 2, 5), R(\text{“y”}, 0, 6), W(\text{“y”}, 2, 7), R(\text{“x”}, 2, 8)\rangle$
$hd \triangleq \{hdsa, hdsb\} \backslash *$ *CC*, *CM*, and *CCv* but no *SC*

$hdsa \triangleq \langle W(\text{“x”}, 1, 1), W(\text{“y”}, 2, 2), R(\text{“y”}, 2, 3)\rangle$
$hdsb \triangleq \langle W(\text{“y”}, 1, 4), R(\text{“x”}, 1, 5), R(\text{“y”}, 1, 6)\rangle$
$hd \triangleq \{hdsa, hdsb\}$ *CC*, *CM*, and *CCv* but no *SC*

$hesa \triangleq \langle W(\text{“x”}, 1, 1), W(\text{“y”}, 1, 2)\rangle$
$hesb \triangleq \langle R(\text{“y”}, 1, 3), W(\text{“x”}, 2, 4)\rangle$
$hesc \triangleq \langle R(\text{“x”}, 2, 5), R(\text{“x”}, 1, 6)\rangle$
$he \triangleq \{hesa, hesb, hesc\}$ not *CC* (nor *CM*, nor *CCv*)

$all \triangleq \{ha, hb, hc, hd, he\}$
$satCC \triangleq \{ha, hb, hc, hd\}$
$satCM \triangleq \{ha, hd\}$
$satCCv \triangleq \{hb, hd\}$

$all \triangleq \{ha, hc, hd, he\}$
$satCC \triangleq \{ha, hc, hd\}$

```
satCM  ≜ {ha, hd}
satCCv ≜ {hd}


 Small scale
 all ≜ {ha, hc}
 satCC ≜ {ha, hc}
 satCM ≜ {ha}
 satCCv ≜ {}

├──────────────────────────────────────────────────────────────┤
WellFormedTest ≜
    ∀ h ∈ all : WellFormed(h)
├──────────────────────────────────────────────────────────────┤
 Test the self-defined EnumerateRO
EasyPO(s) ≜
    LET rels ≜ SUBSET (s × s)
    IN  {po ∈ rels : IsStrictPartialOrder(po, s)}

EnumeratePOTest ≜
  LET pos ≜ partialOrderSubset({0, 1})
    LET ops ≜ {W("x", 2, 0), R("x", 1, 1), R("x", 1, 2)}
        pos1 ≜ EasyPO(ops)
        pos2 ≜ StrictPartialOrderSubset(ops)
    IN  ∧ pos1 = pos2
        ∧ ∀ po ∈ pos1 :
            PrintT("po: " ∘ ToString(po))
├──────────────────────────────────────────────────────────────┤
 Test of utility operators for operations
OpsTest ≜
    ∧ PrintT("OpsTest Begin")
     on history ha
    ∧ Ops(ha) = {W("x", 1, 1), R("x", 2, 2), W("x", 2, 3), R("x", 1, 4)}
    ∧ ReadOps(ha) = {R("x", 2, 2), R("x", 1, 4)}
    ∧ ReadOpsOnKey(ha, "x") = {R("x", 2, 2), R("x", 1, 4)}
    ∧ WriteOps(ha) = {W("x", 1, 1), W("x", 2, 3)}
    ∧ WriteOpsOnKey(ha, "x") = {W("x", 1, 1), W("x", 2, 3)}
     on history he
    ∧ Ops(he) = {W("x", 1, 1), W("y", 1, 2), R("y", 1, 3), W("x", 2, 4), R("x", 2, 5), R("x", 1, 6)}
    ∧ ReadOps(he) = {R("y", 1, 3), R("x", 2, 5), R("x", 1, 6)}
    ∧ ReadOpsOnKey(he, "x") = {R("x", 2, 5), R("x", 1, 6)}
    ∧ WriteOps(he) = {W("x", 1, 1), W("y", 1, 2), W("x", 2, 4)}
    ∧ WriteOpsOnKey(he, "y") = {W("y", 1, 2)}
    ∧ PrintT("OpsTest End")
├──────────────────────────────────────────────────────────────┤
 Test of the auxiliary definitions for the axioms
```

```
CardOfProgramOrderOfHistory(h) ≜
    LET CardOfProgramOrderOfSession(s) ≜
          IF Len(s) ≤ 1 THEN 0 ELSE Sum(1 .. Len(s) − 1)
    IN    ReduceSet(LAMBDA s, x : CardOfProgramOrderOfSession(s) + x, h, 0)

THEOREM ProgramOrderCardinalityTheorem ≜   test of PO(h)
    ∀ h ∈ {ha, hb, hc, hd, he} :
        Cardinality(PO(h)) = CardOfProgramOrderOfHistory(h)

StrictPOPastTest ≜   test of POPast(h, o)
    ∧ PrintT("POPastTest Begin")
    ∧ StrictPOPast(ha, R("x", 2, 2))  = {W("x", 1, 1)}
    ∧ StrictPOPast(hb, R("y", 1, 6))  = {W("x", 2, 4), R("z", 0, 5)}
    ∧ StrictPOPast(hc, W("x", 2, 2)) = {}
   ∧ StrictPOPast(hd, R("x", 1, 4)) = {W("x", 1, 1), R("y", 0, 2), W("y", 1, 3)}
    ∧ StrictPOPast(hd, R("y", 2, 3))  = {W("x", 1, 1), W("y", 2, 2)}
    ∧ StrictPOPast(he, W("x", 2, 4)) = {R("y", 1, 3)}
    ∧ PrintT("POPastTest End")

CausalPastTest ≜   TODO: test of CausalPast(co, o)
    ∧ PrintT("CausalPastTest Begin")
    ∧ LET co ≜ CO(ha)
              o ≜ R("x", 2, 2)
         IN  CausalPast(co, o) = {W("x", 1, 1), R("x", 2, 2), W("x", 2, 3)}
    ∧ PrintT("CausalPastTest End")

CausalHistTest ≜   TODO: test of CausalHist(co, o)
    ∧ PrintT("CausalHistTest Begin")
    ∧ LET co ≜ CO(ha)
              o ≜ R("x", 2, 2)
         IN  CausalHist(co, o) = {⟨W("x", 1, 1), R("x", 2, 2)⟩, ⟨W("x", 2, 3), R("x", 2, 2)⟩}
    ∧ PrintT("CausalHistTest End")

CausalArbTest ≜   TODO: test of CausalArb(co, ar, o)
    ∧ PrintT("CausalArbTest Begin")
    ∧ FALSE
    ∧ PrintT("CausalArbTest End")

AuxiliaryTest ≜   test the auxiliary
    ∧ StrictPOPastTest
    ∧ CausalPastTest
    ∧ CausalHistTest
     ∧ CausalArbTest

├──────────────────────────────────────────────────────────────────┤
   Test of axioms
```

```
RWRegSemanticsTest ≜   test of RWRegSemanticsTest(seq, o)
    ∧ PrintT("RWRegSemanticsTest Begin")
    seq = ⟨⟩
    ∧ RWRegSemantics(⟨⟩, R("x", InitVal, 1))
    ∧ RWRegSemantics(⟨⟩, W("x", 1, 1))
    ∧ ¬RWRegSemantics(⟨⟩, R("x", 2, 1))
    no W("x", _, _) in seq
    ∧ RWRegSemantics(⟨W("y", 1, 1), W("z", 1, 2), W("y", 1, 3)⟩, R("x", InitVal, 4))
    ∧ RWRegSemantics(⟨W("y", 1, 1), W("z", 1, 2), W("y", 1, 3)⟩, W("x", 1, 4))
    ∧ ¬RWRegSemantics(⟨W("y", 1, 1), W("z", 1, 2), W("y", 1, 3)⟩, R("x", 1, 4))
    contains W("x", _, _) in seq
    ∧ RWRegSemantics(⟨W("x", 1, 1), W("y", 1, 2), W("x", 2, 3), W("z", 1, 4)⟩, R("x", 2, 5))
    ∧ ¬RWRegSemantics(⟨W("x", 1, 1), W("y", 1, 2), W("x", 2, 3), W("z", 1, 4)⟩, R("x", 1, 5))
    ∧ PrintT("RWRegSemanticsTest End")

AxCausalValueTest ≜   TODO: test of AxCausalValue()
    ∧ PrintT("AxCausalValueTest Begin")
    ∧ LET co  ≜ CO(ha)
          o1  ≜ W("x", 1, 1)
          o2  ≜ R("x", 2, 2)
          o3  ≜ W("x", 2, 3)
           o4 ≜ R("x", 1, 4)
       IN    ∧ AxCausalValue(co, o1)
             ∧ AxCausalValue(co, o2)
             ∧ AxCausalValue(co, o3)
             ∧ AxCausalValue(co, o4)
    ∧ PrintT("AxCausalValueTest End")

AxCausalSeqTest ≜   Test of AxCausalSeq
    ∧ PrintT("AxCausalSeqTest Begin")
    ∧ LET co  ≜ CO(ha)
          o1  ≜ W("x", 1, 1)
          o2  ≜ R("x", 2, 2)
          o3  ≜ W("x", 2, 3)
           o4 ≜ R("x", 1, 4)
       IN    ∧ AxCausalSeq(ha, co, o1)
             ∧ AxCausalSeq(ha, co, o2)
             ∧ AxCausalSeq(ha, co, o3)
             ∧ AxCausalSeq(ha, co, o4)
    ∧ PrintT("AxCausalSeqTest End")


AxCausalArbTest ≜   TODO: test of AxCausalArb()
    ∧ PrintT("AxCausalArbTest Begin")
    ∧ FALSE
    ∧ PrintT("AxCausalArbTest End")
```

```
AxiomsTest ≜   Test the axioms
      ∧ RWRegSemanticsTest
      ∧ AxCausalValueTest
      ∧ AxCausalSeqTest
      ∧ AxCausalArbTest
```

Test of the relations defined for bad patterns

```
oidHB(h) ≜   All happened-before relation for o ∈  history h repsented by oid
     LET oidHBo(o) ≜ {⟨o1.oid − 1, o2.oid − 1⟩ : ⟨o1, o2⟩ ∈ HBo(h, o)}
     IN   {⟨o.oid − 1, oidHBo(o)⟩ : o ∈ Ops(h)}

HBTest ≜
      ∧   PrintT("HBTest Begin")
      ∧   PrintT(oidHB(ha))
      ∧   PrintT(oidHB(hb))
      ∧   PrintT(oidHB(hc))
      ∧   PrintT(oidHB(hd))
      ∧   PrintT(oidHB(he))
      ∧   PrintT("HBTest End")
```

Test of the definitions of causal consistency

ha: 4; *hb*: 7; *hc*: 4; *hd*: 8; he: 6

```
CCDefTest ≜
      ∧ PrintT("CCDefTest Begin")
      ∧ ∀ h ∈ satCC :
           ∧ PrintT(h)
           ∧ CC(h)
      ∧ ∀ h ∈ all \ satCC :
           ∧ PrintT(h)
           ∧ ¬CC(h)
      ∧ PrintT("CCDefTest End")

CCvDefTest ≜
      ∧ PrintT("CCvDefTest Begin")
      ∧ ∀ h ∈ satCCv :
           ∧ PrintT(h)
           ∧ CCv(h)
      ∧ ∀ h ∈ all \ satCCv :
           ∧ PrintT(h)
           ∧ ¬CCv(h)
      ∧ PrintT("CCvDefTest End")

CMDefTest ≜
      ∧ PrintT("CMDefTest Begin")
```

```
    $\land \forall h \in satCM :$
        $\land PrintT(h)$
        $\land CM(h)$
    $\land \forall h \in all \setminus satCM :$
        $\land PrintT(h)$
        $\land \neg CM(h)$
    $\land PrintT($“CMDefTest End”$)$

$CausalDefTest \triangleq$
    $\land CCDefTest$
    $\land CCvDefTest$
    $\land CMDefTest$

$BigCausalDefTest \triangleq$
    $\land BigCC(hb)$
    $\land BigCCv(hb)$
    $\land \neg BigCM(hb)$
|--------------------------------------------------------------------------|
  Test of the checking algorithms for causal consistency

  ha: 4; hb: 7; hc: 4; hd: 8; he: 6
$CCAlgTest \triangleq$   Test of the checking algorithm CCAlg for CC (Causal Consistency)
    LET $sat \triangleq satCC$
    IN  $\land \forall h \in sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land CCAlg(h)$
        $\land \forall h \in all \setminus sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land \neg CCAlg(h)$

$CCvAlgTest \triangleq$   Test of the checking algorithm CCvAlg for CCv (Causal Convergence)
    LET $sat \triangleq satCCv$
    IN  $\land \forall h \in sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land CCvAlg(h)$
        $\land \forall h \in all \setminus sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land \neg CCvAlg(h)$

$CMAlgTest \triangleq$   Test of the checking algorithm CMAlg for CM (Causal Memory)
    LET $sat \triangleq satCM$
    IN  $\land \forall h \in sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land CMAlg(h)$
        $\land \forall h \in all \setminus sat :$
            $\land PrintT(ToString(h) \circ$ “ is differentiated: ” $\circ ToString(IsDifferentiated(h)))$
            $\land \neg CMAlg(h)$
```

```
CausalAlgTest ≜
      ∧ CCAlgTest
      ∧ CCvAlgTest
      ∧ CMAlgTest

----------------------------------------------------------------
VARIABLES x  keep it so that the model can be run
================================================================
\* Modification History
\* Last modified Fri May 28 16:02:20 CST 2021 by Young
\* Last modified Thu Apr 22 15:12:59 CST 2021 by hengxin
\* Created Fri Apr 09 11:53:33 CST 2021 by hengxin
```