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*

$hdsa \triangleq \langle W(\text{“x”}, 1, 1),\ R(\text{“y”}, 0, 2),\ W(\text{“y”}, 1, 3),\ R(\text{“x”}, 1, 4)\rangle$
$hdsb \triangleq \langle W(\text{“x”}, 2, 5),\ R(\text{“y”}, 0, 6),\ W(\text{“y”}, 2, 7),\ R(\text{“x”}, 2, 8)\rangle$
$hd \triangleq \{hdsa, hdsb\}$ *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*)

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

$WellFormedTest \triangleq$
$\quad \forall\, h \in all : WellFormed(h)$

Test of utility operators for operations

$OpsTest \triangleq$
$\quad \wedge\ PrintT(\text{“OpsTest Begin”})$
on history ha
$\quad \wedge\ Ops(ha) = \{W(\text{“x”}, 1, 1),\ R(\text{“x”}, 2, 2),\ W(\text{“x”}, 2, 3),\ R(\text{“x”}, 1, 4)\}$
$\quad \wedge\ ReadOps(ha) = \{R(\text{“x”}, 2, 2),\ R(\text{“x”}, 1, 4)\}$

```
    ∧ 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)

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

CausalPastTest ≜  TODO: test of CausalPast(co, o)
    ∧ PrintT("CausalPastTest Begin")
    ∧ FALSE
    ∧ PrintT("CausalPastTest End")

CausalHistTest ≜  TODO: test of CausalHist(co, o)
    ∧ PrintT("CausalHistTest Begin")
    ∧ FALSE
    ∧ PrintT("CausalHistTest End")

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

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()
    ∧ FALSE

AxCausalArbTest ≜  TODO: test of AxCausalArb()
    ∧ FALSE
```

Test of the definitions of causal consistency

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

```
CCDefTest ≜
    ∧ PrintT("CCDefTest Begin")
    ∧ PrintT(CC(ha))
    ∧ PrintT(CC(hc))
   ∧ PrintT(¬CC(he)) \* too slow
   ∧ LET sat ≜ {ha, hb, hc, hd}
     IN   ∧ ∀ h ∈ sat: CC(h)
          ∧ ∀ h ∈ all \ sat: ¬CC(h)
    ∧ PrintT("CCDefTest End")

CCvDefTest ≜
    ∧ PrintT("CCvDefTest Begin")
    ∧ PrintT(¬CCv(ha))
   ∧ CCv(hb)
    ∧ PrintT(¬CCv(hc))
   ∧ CCv(hd)
   ∧ PrintT(¬CCv(he))

   LET sat ≜ {hb, hd}
   IN   ∧ ∀ h ∈ sat: CCv(h)
        ∧ ∀ h ∈ all \ sat: ¬CCv(h)
    ∧ PrintT("CCvDefTest End")
```

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 $sat \triangleq \{ha, hb, hc, hd\}$

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 $sat \triangleq \{hb, hd\}$

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: " $\circ\ ToString(IsDifferentiated(h)))$

$\land \neg CCvAlg(h)$

VARIABLES x keep it so that the model can be run

* Modification *History*

* Last modified *Thu Apr* 22 15:12:59 *CST* 2021 by *hengxin*

* Created *Fri Apr* 09 11:53:33 *CST* 2021 by *hengxin*