MODULE *CJupiter*

Model of our own *CJupiter* protocol.

$\text{EXTENDS } \mathit{Integers}, \mathit{OT}, \mathit{TLC}, \mathit{AdditionalFunctionOperators}, \mathit{AdditionalSequenceOperators}$

CONSTANTS

- Client, — the set of client replicas
- Server, — the (unique) server replica
- Char, — set of characters allowed
- $\mathit{InitState}$ — the initial state of each replica

$\mathit{Replica} \triangleq \mathit{Client} \cup \{\mathit{Server}\}$

$\mathit{List} \triangleq \mathit{Seq}(\mathit{Char} \cup \mathit{Range}(\mathit{InitState}))$ — all possible lists/strings

$\mathit{MaxLen} \triangleq \mathit{Cardinality}(\mathit{Char}) + \mathit{Len}(\mathit{InitState})$ — the max length of lists in any states;
We assume that all inserted elements are unique.

$\mathit{ClientNum} \triangleq \mathit{Cardinality}(\mathit{Client})$

$\mathit{Priority} \triangleq \text{CHOOSE } f \in [\mathit{Client} \rightarrow 1 \,..\, \mathit{ClientNum}] : \mathit{Injective}(f)$

ASSUME

$\land\ \mathit{Range}(\mathit{InitState}) \cap \mathit{Char} = \{\}$ — due to the uniqueness requirement

$\land\ \mathit{Priority} \in [\mathit{Client} \rightarrow 1 \,..\, \mathit{ClientNum}]$

The set of all operations. Note: The positions are indexed from 1.

$\mathit{Rd} \triangleq [\mathit{type} : \{\text{“Rd”}\}]$

$\mathit{Del} \triangleq [\mathit{type} : \{\text{“Del”}\}, \mathit{pos} : 1 \,..\, \mathit{MaxLen}]$

$\mathit{Ins} \triangleq [\mathit{type} : \{\text{“Ins”}\}, \mathit{pos} : 1 \,..\, (\mathit{MaxLen} + 1), \mathit{ch} : \mathit{Char}, \mathit{pr} : 1 \,..\, \mathit{ClientNum}]$ — *pr*: priority

$\mathit{Op} \triangleq \mathit{Ins} \cup \mathit{Del}$

Cop: operation of type *Op* with context

$\mathit{Oid} \triangleq [c : \mathit{Client}, \mathit{seq} : \mathit{Nat}]$ — operation identifier

$\mathit{Cop} \triangleq [\mathit{op} : \mathit{Op} \cup \{\mathit{Nop}\}, \mathit{oid} : \mathit{Oid}, \mathit{ctx} : \text{SUBSET } \mathit{Oid}, \mathit{sctx} : \text{SUBSET } \mathit{Oid}]$

tb: Is *cop*1 totally ordered before *cop*2?

At a given replica $r \in \mathit{Replica}$, these can be determined in terms of *sctx*.

$$
\begin{aligned}
&\mathit{tb}(\mathit{cop}1, \mathit{cop}2, r) \triangleq \\
&\quad \lor\ \mathit{cop}1.\mathit{oid} \in \mathit{cop}2.\mathit{sctx} \\
&\quad \lor\ \land\ \mathit{cop}1.\mathit{oid} \notin \mathit{cop}2.\mathit{sctx} \\
&\quad\quad\ \land\ \mathit{cop}2.\mathit{oid} \notin \mathit{cop}1.\mathit{sctx} \\
&\quad\quad\ \land\ \mathit{cop}1.\mathit{oid}.c \neq r
\end{aligned}
$$

OT of two operations of type *Cop*.

$\mathit{COT}(\mathit{lcop}, \mathit{rcop}) \triangleq [\mathit{lcop} \text{ EXCEPT } !.\mathit{op} = \mathit{Xform}(\mathit{lcop}.\mathit{op}, \mathit{rcop}.\mathit{op}), !.\mathit{ctx} = @ \cup \{\mathit{rcop}.\mathit{oid}\}]$

VARIABLES

For the client replicas:

$cseq$, $cseq[c]$: local sequence number at client $c \in Client$

For the server replica:

$soids$, the set of operations the *Server* has executed

For all replicas: the n-ary ordered state space

css, $css[r]$: the n-ary ordered state space at replica $r \in Replica$

cur, $cur[r]$: the current node of css at replica $r \in Replica$

$state$, $state[r]$: state (the list content) of replica $r \in Replica$

For communication between the *Server* and the Clients:

$cincoming$, $cincoming[c]$: incoming channel at the client $c \in Client$

$sincoming$, incoming channel at the *Server*

For model checking:

$chins$ a set of chars to insert

$comm \triangleq$ INSTANCE $CSComm$ WITH $Msg \leftarrow Cop$

$eVars \triangleq \langle chins \rangle$ variables for the environment

$cVars \triangleq \langle cseq \rangle$ variables for the clients

$sVars \triangleq \langle soids \rangle$ variables for the server

$dsVars \triangleq \langle css, cur, state \rangle$ variables for the data structure: the n-ary ordered state space

$commVars \triangleq \langle cincoming, sincoming \rangle$ variables for communication

$vars \triangleq \langle eVars, cVars, sVars, commVars, dsVars \rangle$ all variables

---

An css is a directed graph with labeled edges.

It is represented by a record with node field and edge field.

Each node is characterized by its context, a set of operations.

Each edge is labeled with an operation. For clarity, we denote edges by records instead of tuples.

$IsCSS(G) \triangleq$

$\quad \land G = [node \mapsto G.node, edge \mapsto G.edge]$

$\quad \land G.node \subseteq (\text{SUBSET } Oid)$

$\quad \land G.edge \subseteq [from : G.node, to : G.node, cop : Cop]$

$TypeOK \triangleq$

For the client replicas:

$\quad \land cseq \in [Client \rightarrow Nat]$

For the server replica:

$\quad \land soids \subseteq Oid$

For all replicas: the n-ary ordered state space

$\quad \land \forall r \in Replica : IsCSS(css[r])$

$\quad \land cur \in [Replica \rightarrow \text{SUBSET } Oid]$

$\land\ state \in [Replica \rightarrow List]$

For communication between the server and the clients:

$\land\ comm!TypeOK$

For model checking:

$\land\ chins \subseteq Char$

The *Init* predicate.

$Init \triangleq$

For the client replicas:

$\land\ cseq = [c \in Client \mapsto 0]$

For the server replica:

$\land\ soids = \{\}$

For all replicas: the n-ary ordered state space

$\land\ css = [r \in Replica \mapsto [node \mapsto \{\{\}\},\ edge \mapsto \{\}]]$
$\land\ cur = [r \in Replica \mapsto \{\}]$
$\land\ state = [r \in Replica \mapsto InitState]$

For communication between the server and the clients:

$\land\ comm!Init$

For model checking:

$\land\ chins = Char$

Locate the node in *rcss* which matches the context *ctx* of cop.

rcss: the *css* at replica $r \in Replica$

$Locate(cop,\ rcss) \triangleq \text{CHOOSE}\ n \in (rcss.node) : n = cop.ctx$

xForm: iteratively transform cop with a path through the *css* at replica $r \in Replica$, following the first edges.

$xForm(cop,\ r) \triangleq$
$\quad \text{LET}\ rcss \triangleq css[r]$
$\quad\quad u \triangleq Locate(cop,\ rcss)$
$\quad\quad v \triangleq u \cup \{cop.oid\}$
$\quad\quad \text{RECURSIVE}\ xFormHelper(\_,\ \_,\ \_,\ \_)$
$\quad\quad$ 'h' stands for "helper"; *xcss*: *eXtra css* created during transformation
$\quad\quad xFormHelper(uh,\ vh,\ coph,\ xcss) \triangleq$
$\quad\quad\quad \text{IF}\ uh = cur[r]$
$\quad\quad\quad\quad \text{THEN}\ xcss$
$\quad\quad\quad\quad \text{ELSE}\ \text{LET}\ fedge \triangleq \text{CHOOSE}\ e \in rcss.edge :$
$\quad\quad\quad\quad\quad\quad\quad \land\ e.from = uh$
$\quad\quad\quad\quad\quad\quad\quad \land\ \forall\ uhe \in rcss.edge :$
$\quad\quad\quad\quad\quad\quad\quad\quad (uhe.from = uh \land uhe \neq e) \Rightarrow tb(e.cop,\ uhe.cop,\ r)$
$\quad\quad\quad\quad\quad\quad uprime \triangleq fedge.to$
$\quad\quad\quad\quad\quad\quad fcop \triangleq fedge.cop$

```
                        coph2fcop ≜ COT(coph, fcop)
                        fcop2coph ≜ COT(fcop, coph)
                         vprime ≜ vh ∪ {fcop.oid}
                    IN   xFormHelper(uprime, vprime, coph2fcop,
                           [xcss EXCEPT !.node = @ ∘ ⟨vprime⟩,
                                        the order of recording edges here is important
                                        !.edge = @ ∘ ⟨[from ↦ vh, to ↦ vprime, cop ↦ fcop2coph],
                                                [from ↦ uprime, to ↦ vprime, cop ↦ coph2fcop]⟩])
   IN  xFormHelper(u, v, cop, [node ↦ ⟨v⟩,
                              edge ↦ ⟨[from ↦ u, to ↦ v, cop ↦ cop]⟩])
```

Perform cop at replica $r \in Replica$.

```
Perform(cop, r) ≜
   LET xcss ≜ xForm(cop, r)
       xn ≜ xcss.node
       xe ≜ xcss.edge
       xcur ≜ Last(xn)
       xcop ≜ Last(xe).cop
   IN  ∧ css′ = [css EXCEPT ![r].node = @ ∪ Range(xn),
                            ![r].edge = @ ∪ Range(xe)]
       ∧ cur′ = [cur EXCEPT ![r] = xcur]
       ∧ state′ = [state EXCEPT ![r] = Apply(xcop.op, @)]
──────────────────────────────────────────────────────────────
```

Client $c \in Client$ issues an operation op.

```
DoOp(c, op) ≜   op: the raw operation generated by the client c ∈ Client
     ∧ cseq′ = [cseq EXCEPT ![c] = @ + 1]
     ∧ LET cop ≜ [op ↦ op, oid ↦ [c ↦ c, seq ↦ cseq′[c]], ctx ↦ cur[c], sctx ↦ {}]
         IN  ∧ Perform(cop, c)
             ∧ comm!CSend(cop)

DoIns(c) ≜
   ∃ ins ∈ Ins :
     ∧ ins.pos ∈ 1 .. (Len(state[c]) + 1)
     ∧ ins.ch ∈ chins
     ∧ ins.pr = Priority[c]
     ∧ chins′ = chins \ {ins.ch}   We assume that all inserted elements are unique.
     ∧ DoOp(c, ins)
     ∧ UNCHANGED sVars

DoDel(c) ≜
   ∃ del ∈ Del :
     ∧ del.pos ∈ 1 .. Len(state[c])
     ∧ DoOp(c, del)
     ∧ UNCHANGED ⟨sVars, eVars⟩

Do(c) ≜
```

$\quad\quad \lor\ DoIns(c)$

$\quad\quad \lor\ DoDel(c)$

Client $c \in Client$ receives a message from the *Server*.

$Rev(c) \ \triangleq$

$\quad \land\ comm!CRev(c)$

$\quad \land\ \text{LET } cop \ \triangleq\ Head(cincoming[c])$ the received original operation

$\quad\quad \text{IN } \ Perform(cop,\ c)$

$\quad \land\ \text{UNCHANGED } \langle eVars,\ cVars,\ sVars \rangle$

The *Server* receives a message.

$SRev \ \triangleq$

$\quad \land\ comm!SRev$

$\quad \land\ \text{LET } cop \ \triangleq\ [Head(sincoming) \text{ EXCEPT } !.sctx = soids]$ set its *sctx* field

$\quad\quad \text{IN } \ \land\ soids' = soids \cup \{cop.oid\}$

$\quad\quad\quad\quad \land\ Perform(cop,\ Server)$

$\quad\quad\quad\quad \land\ comm!SSendSame(cop.oid.c,\ cop)$ broadcast the original operation

$\quad \land\ \text{UNCHANGED } \langle eVars,\ cVars \rangle$

The next-state relation.

$Next \ \triangleq$

$\quad \lor\ \exists\, c \in Client : Do(c) \lor Rev(c)$

$\quad \lor\ SRev$

The *Spec*. (*TODO*: Check the fairness condition.)

$Spec \ \triangleq\ Init \land \Box[Next]_{vars} \land \mathrm{WF}_{vars}(Next)$

The compactness of *CJupiter*: the *css* at all replicas are essentially the same.

$IgnoreSctx(rcss) \ \triangleq$

$\quad [rcss \text{ EXCEPT } !.edge = \{[e \text{ EXCEPT } !.cop.sctx = \{\}] : e \in @\}]$

$Compactness \ \triangleq$

$\quad comm!EmptyChannel \Rightarrow Cardinality(\{IgnoreSctx(css[r]) : r \in Replica\}) = 1$

$\text{THEOREM } Spec \Rightarrow Compactness$

---

* Modification History

* *Last* modified *Wed Oct* 24 10:41:24 *CST* 2018 by *hengxin*

* Created Sat *Sep* 01 11:08:00 *CST* 2018 by *hengxin*