MODULE *TwoPhaseCommit*

This is a TLA+ specification of the two-phase commit protocol used for distributed data bases. It models only one instance of the protocol, *i.e.* for a single transaction.

CONSTANT *Node*   set of nodes other than the coordinator

VARIABLES

cState,	the state of the coordinator
nState,	the state of the non-coordinator nodes
committed,	nodes that the coordinator knows are *OK* for committing
msgs	messages sent during the protocol

$$vars \triangleq \langle cState, nState, committed, msgs \rangle$$

Possible states of coordinator.

$$CState \triangleq \{\textsf{``preparing''}, \textsf{``committed''}, \textsf{``aborted''}\}$$

Possible states of non-coordinator participants.

$$NState \triangleq \{\textsf{``preparing''}, \textsf{``readyCommit''}, \textsf{``readyAbort''}, \textsf{``committed''}, \textsf{``aborted''}\}$$

Messages sent during the protocol.

$$Message \triangleq$$

node informs coordinator about its decision

$$[kind : \{\textsf{``commit''}, \textsf{``abort''}\}, node : Node] \cup$$

coordinator tells nodes whether to commit or abort

$$[kind : \{\textsf{``doCommit''}, \textsf{``doAbort''}\}]$$

$$\begin{aligned}
commit(n) &\triangleq [kind \mapsto \textsf{``commit''}, node \mapsto n] \\
abort(n) &\triangleq [kind \mapsto \textsf{``abort''}, node \mapsto n] \\
doCommit &\triangleq [kind \mapsto \textsf{``doCommit''}] \\
doAbort &\triangleq [kind \mapsto \textsf{``doAbort''}]
\end{aligned}$$

The following predicate specifies what values the variables can take during an execution of the protocol.

$$\begin{aligned}
TypeOK \triangleq \\
&\land cState \in CState \\
&\land nState \in [Node \rightarrow NState] \\
&\land committed \subseteq Node \\
&\land msgs \subseteq Message
\end{aligned}$$

The initial state of the protocol.

$$\begin{aligned}
Init \triangleq \\
&\land cState = \textsf{``preparing''} \\
&\land nState = [n \in Node \mapsto \textsf{``preparing''}] \\
&\land committed = \{\} \\
&\land msgs = \{\}
\end{aligned}$$

The following action formulas describe the possible transitions of the nodes.

A participant decides and informs the coordinator of its decision.

$$
\begin{array}{l}
\mathit{Decide}(n) \triangleq \\
\quad \land\ \mathit{nState}[n] = \text{“}\textsf{preparing}\text{”} \\
\quad \land\ \lor\ \land\ \mathit{nState}' = [\mathit{nState}\ \textsc{except}\ ![n] = \text{“}\textsf{readyCommit}\text{”}] \\
\quad \land\ \mathit{msgs}' = \mathit{msgs} \cup \{\mathit{commit}(n)\} \\
\quad \lor\ \land\ \mathit{nState}' = [\mathit{nState}\ \textsc{except}\ ![n] = \text{“}\textsf{readyAbort}\text{”}] \\
\quad \land\ \mathit{msgs}' = \mathit{msgs} \cup \{\mathit{abort}(n)\} \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{cState},\ \mathit{committed} \rangle
\end{array}
$$

The coordinator receives a new commit decision for some participant. If all participants wish to commit, it sends an order to commit.

$$
\begin{array}{l}
\mathit{RcvCommit}(n) \triangleq \\
\quad \land\ n \notin \mathit{committed} \land \mathit{commit}(n) \in \mathit{msgs} \\
\quad \land\ \mathit{committed}' = \mathit{committed} \cup \{n\} \\
\quad \land\ \textsc{if}\ \ \mathit{committed}' = \mathit{Node} \\
\quad \ \ \textsc{then}\ \ \land\ \mathit{cState}' = \text{“}\textsf{committed}\text{”} \\
\quad \phantom{\land\ \ \ \textsc{then}\ \ }\land\ \mathit{msgs}' = \mathit{msgs} \cup \{\mathit{doCommit}\} \\
\quad \ \ \textsc{else}\ \ \textsc{unchanged}\ \langle \mathit{cState},\ \mathit{msgs} \rangle \\
\quad \land\ \mathit{nState}' = \mathit{nState}
\end{array}
$$

The coordinator receives an abort decision and sends an order to abort.

$$
\begin{array}{l}
\mathit{RcvAbort}(n) \triangleq \\
\quad \land\ \mathit{abort}(n) \in \mathit{msgs} \\
\quad \land\ \mathit{cState}' = \text{“}\textsf{aborted}\text{”} \\
\quad \land\ \mathit{msgs}' = \mathit{msgs} \cup \{\mathit{doAbort}\} \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{nState},\ \mathit{committed} \rangle
\end{array}
$$

A participant receives a commit or abort order from the coordinator.

$$
\begin{array}{l}
\mathit{Execute}(n) \triangleq \\
\quad \land\ \lor\ \mathit{doCommit} \in \mathit{msgs} \land \mathit{nState}' = [\mathit{nState}\ \textsc{except}\ ![n] = \text{“}\textsf{committed}\text{”}] \\
\quad \lor\ \mathit{doAbort} \in \mathit{msgs} \land \mathit{nState}' = [\mathit{nState}\ \textsc{except}\ ![n] = \text{“}\textsf{aborted}\text{”}] \\
\quad \land\ \textsc{unchanged}\ \langle \mathit{cState},\ \mathit{committed},\ \mathit{msgs} \rangle
\end{array}
$$

The overall next-state relation is the disjunction of the action formulas defined previously.

$$
\begin{array}{l}
\mathit{Next} \triangleq \\
\quad \exists\, n \in \mathit{Node} : \mathit{Decide}(n) \lor \mathit{RcvCommit}(n) \lor \mathit{RcvAbort}(n) \lor \mathit{Execute}(n)
\end{array}
$$

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

$$
\begin{array}{l}
\mathit{NoConflictingOrders} \triangleq \\
\quad \mathit{doCommit} \in \mathit{msgs} \Rightarrow \\
\quad\quad \land\ \neg(\mathit{doAbort} \in \mathit{msgs}) \\
\quad\quad \land\ \forall\, q \in \mathit{Node} : \mathit{nState}[q] \in \{\text{“}\textsf{readyCommit}\text{”},\ \text{“}\textsf{committed}\text{”}\}
\end{array}
$$

Correctness properties.

The coordinator never sends both a *doCommit* and a *doAbort* message.

$$CommitOrAbort \triangleq \neg(doCommit \in msgs \land doAbort \in msgs)$$

The coordinator may commit only if all participants wish to commit and no participant wishes to abort.

$$\begin{array}{l} AbortWins \triangleq \\ \quad doCommit \in msgs \Rightarrow \\ \quad\quad \forall\, n \in Node : \\ \quad\quad\quad \land\ commit(n) \in msgs \land nState[n] \in \{\text{“readyCommit”},\ \text{“committed”}\} \\ \quad\quad\quad \land\ abort(n) \notin msgs \end{array}$$

$$\begin{array}{l} Terminal \triangleq \\ \quad \lor\ cState = \text{“aborted”} \land \forall\, n \in Node : nState[n] = \text{“aborted”} \\ \quad \lor\ cState = \text{“committed”} \land \forall\, n \in Node : nState[n] = \text{“committed”} \end{array}$$

The protocol may only terminate in a terminal state.

$$CorrectTermination \triangleq (\neg \text{ENABLED}\ \langle Next \rangle_{vars}) \Rightarrow Terminal$$

$$Liveness \triangleq \Diamond Terminal$$

Two-phase commitment implements distributed commitment.

$$DC \triangleq \text{INSTANCE}\ DistributedCommit$$

$$\text{THEOREM}\ Spec \Rightarrow DC!Spec$$