MODULE *TwoPhase*

This specification is based on "Two-Phase *Commit*", Lecture 6 of the TLA+ Video Course. It describes the Two-Phase *Commit* protocol, in which a transaction manager (*TM*) coordinates the resource managers (*RMs*) to implement the Transaction *Commit* specification of module *TCommit*. In this specification, *RMs* spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *TM* can send to the *RMs*.

For simplicity, we also eliminate *Abort* messages sent by an *RM* when it decides to abort. Such a message would cause the *TM* to abort the transaction, an event represented here by the *TM* spontaneously deciding to abort.

CONSTANT RM The set of all Resource Managers (*e.g.* {"*r*1", "*r*2", "*r*3"}).

VARIABLES
- $rmState$, — *rmState*[*rm*] is the state of the Resource Manager *rm*.
- $tmState$, — Transaction Manager state: "init" or "done".
- $tmPrepared$, — The set of resource managers the transaction manager knows are prepared.
- $msgs$

In the protocol, processes communicate with one another by sending messages. For simplicity, we represent message passing with the variable *msgs* whose value is the set of all messages that have been sent. A message is sent by adding it to the set *msgs*. An action that, in an implementation, would be enabled by the receipt of a certain message is here enabled by the presence of that message in *msgs*. For simplicity, messages are never removed from *msgs*. This allows a single message to be received by multiple receivers. Receipt of the same message twice is therefore allowed; but in this particular protocol, that's not a problem.

The set of all possible messages. Messages of type "Prepared" are sent from the *RM* indicated by the message's *rm* field to the *TM*. Messages of type "Commit" and "Abort" are broadcast by the *TM*, to be received by all *RMs*. The set *msgs* contains just a single copy of such message.

$$Messages \triangleq [type : \{\text{"Prepared"}\},\ rm : RM] \cup [type : \{\text{"Commit"}, \text{"Abort"}\}]$$

$$\begin{aligned} TPTypeOK \triangleq\ & \land\ rmState \in [RM \rightarrow \{\text{"working"}, \text{"prepared"}, \text{"committed"}, \text{"aborted"}\}] \\ & \land\ tmState \in \{\text{"init"}, \text{"done"}\} \\ & \land\ tmPrepared \subseteq RM \\ & \land\ msgs \subseteq Messages \end{aligned}$$

$$\begin{aligned} TPInit \triangleq\ & \land\ rmState = [r \in RM \mapsto \text{"working"}] \\ & \land\ tmState = \text{"init"} \\ & \land\ tmPrepared = \{\} \\ & \land\ msgs = \{\} \end{aligned}$$

$TMRecvPrepared(r) \triangleq$ The *TM* receives a "Prepared" message from resource manager *r*.

$$\begin{aligned} & \land\ tmState = \text{"init"} \\ & \land\ [type \mapsto \text{"Prepared"}, rm \mapsto r] \in msgs \quad \text{The message received exists} \\ & \land\ tmPrepared' = tmPrepared \cup \{r\} \\ & \land\ \text{UNCHANGED}\ \langle rmState, tmState, msgs \rangle \end{aligned}$$

```
$TMCommit \triangleq$    The $TM$ commits the transaction
  $\land\ tmState = \text{“init”}$
  $\land\ tmState' = \text{“done”}$                      $TM$ state transitions
  $\land\ tmPrepared = RM$                               $TM$ sees all $RMs$ as prepared
  $\land\ rmState = [r \in RM \mapsto \text{“prepared”}]$     All $RMs$ are prepared (nececessary?)
  $\land\ msgs' = msgs \cup \{[type \mapsto \text{“Commit”}]\}$   Send the $Commit$ message
  $\land\ \text{UNCHANGED}\ \langle rmState,\ tmPrepared \rangle$

$TMAbort \triangleq$    The $TM$ spontaneously aborts the transaction
  $\land\ tmState = \text{“init”} \land tmState' = \text{“done”}$    $TM$ state transitions
  $\land\ msgs' = msgs \cup \{[type \mapsto \text{“Abort”}]\}$     Send the $Abort$ message
  $\land\ \text{UNCHANGED}\ \langle rmState,\ tmPrepared \rangle$
```

```
$RMPrepare(r) \triangleq$  $\land\ rmState[r] = \text{“working”}$
                  $\land\ rmState' = [rmState\ \text{EXCEPT}\ ![r] = \text{“prepared”}]$
                  $\land\ msgs' = msgs \cup \{[type \mapsto \text{“Prepared”},\ rm \mapsto r]\}$
                  $\land\ \text{UNCHANGED}\ \langle tmState,\ tmPrepared \rangle$

$RM$ spontaneously chooses to abort. No message sent in our simplified model.
$RMChooseToAbort(r) \triangleq$  $\land\ rmState[r] = \text{“aborted”}$
                        $\land\ rmState' = [rmState\ \text{EXCEPT}\ ![r] = \text{“aborted”}]$
                        $\land\ \text{UNCHANGED}\ \langle tmState,\ tmPrepared,\ msgs \rangle$

$RMRecvCommitMsg(r) \triangleq$    $RM\ r$ is told to commit
  $\land\ [type \mapsto \text{“Commit”}] \in msgs$       A $Commit$ message should have been sent
  $\land\ rmState' = [rmState\ \text{EXCEPT}\ ![r] = \text{“committed”}]$
  $\land\ \text{UNCHANGED}\ \langle tmState,\ tmPrepared,\ msgs \rangle$

$RMRecvAbortMsg(r) \triangleq$    $RM$ is told to abort
  $\land\ [type \mapsto \text{“Abort”}] \in msgs$        An $Abort$ message should have been sent
  $\land\ rmState' = [rmState\ \text{EXCEPT}\ ![r] = \text{“aborted”}]$
  $\land\ \text{UNCHANGED}\ \langle tmState,\ tmPrepared,\ msgs \rangle$

$TPNext \triangleq$  $\lor\ TMCommit$             The transaction is committed by the $TM$.
          $\lor\ TMAbort$              The transaction is aborted by the $TM$.
          $\lor\ \exists\, r \in RM :$          A message is sent or received by a $RM$ or the $TM$.
              $\lor\ TMRecvPrepared(r)$
              $\lor\ RMPrepare(r)$
              $\lor\ RMChooseToAbort(r)$
              $\lor\ RMRecvCommitMsg(r)$
              $\lor\ RMRecvAbortMsg(r)$
```

TODO on Lecture 8

* Modification History
* Last modified Sun *Apr* 04 10:00:30 *BRT* 2021 by *felipec*
* Created Sat *Apr* 03 15:07:14 *BRT* 2021 by *felipec*