
MODULE *TwoPhase*

This specification is discussed in “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 (*RM*s) to implement the Transaction *Commit* specification of module *TCommit*. In this specification, *RM*s spontaneously issue *Prepared* messages. We ignore the Prepare messages that the *TM* can send to the *RM*s.

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 resource managers

VARIABLES

rmState, *rmState*[*r*] is the state of resource manager *r*.
tmState, The state of the transaction manager.
tmPrepared, The set of *RM*s from which the *TM* has received “Prepared”
 messages.

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.

Messages \triangleq

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 *RM*s. The set *msgs* contains just a single copy of such a message.

$[type : \{ \text{“Prepared”} \}, rm : RM] \cup [type : \{ \text{“Commit”}, \text{“Abort”} \}]$

TPTYPEOK \triangleq

The type-correctness invariant

$\wedge rmState \in [RM \rightarrow \{ \text{“working”}, \text{“prepared”}, \text{“committed”}, \text{“aborted”} \}]$
 $\wedge tmState \in \{ \text{“init”}, \text{“done”} \}$
 $\wedge tmPrepared \subseteq RM$
 $\wedge msgs \subseteq Messages$

TPInit \triangleq

The initial predicate.

$\wedge rmState = [r \in RM \mapsto \text{“working”}]$
 $\wedge tmState = \text{“init”}$
 $\wedge tmPrepared = \{ \}$
 $\wedge msgs = \{ \}$

We now define the actions that may be performed by the processes, first the *TM*’s actions, then the *RM*s’ actions.

TMRCvPrepared(*r*) \triangleq

The *TM* receives a “Prepared” message from resource manager *r*. We could add the additional enabling condition $r \notin tmPrepared$, which disables the action if the *TM* has already received this message. But there is no need, because in that case the action has no effect; it leaves the state unchanged.

$$\begin{aligned} & \wedge tmState = \text{“init”} \\ & \wedge [type \mapsto \text{“Prepared”}, rm \mapsto r] \in msgs \\ & \wedge tmPrepared' = tmPrepared \cup \{r\} \\ & \wedge \text{UNCHANGED } \langle rmState, tmState, msgs \rangle \end{aligned}$$

$TMCommit \triangleq$

The *TM* commits the transaction; enabled iff the *TM* is in its initial state and every *RM* has sent a “Prepared” message.

$$\begin{aligned} & \wedge tmState = \text{“init”} \\ & \wedge tmPrepared = RM \\ & \wedge tmState' = \text{“done”} \\ & \wedge msgs' = msgs \cup \{[type \mapsto \text{“Commit”}]\} \\ & \wedge \text{UNCHANGED } \langle rmState, tmPrepared \rangle \end{aligned}$$

$TMAbort \triangleq$

The *TM* spontaneously aborts the transaction.

$$\begin{aligned} & \wedge tmState = \text{“init”} \\ & \wedge tmState' = \text{“done”} \\ & \wedge msgs' = msgs \cup \{[type \mapsto \text{“Abort”}]\} \\ & \wedge \text{UNCHANGED } \langle rmState, tmPrepared \rangle \end{aligned}$$

$RMPprepare(r) \triangleq$

Resource manager *r* prepares.

$$\begin{aligned} & \wedge rmState[r] = \text{“working”} \\ & \wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{“prepared”}] \\ & \wedge msgs' = msgs \cup \{[type \mapsto \text{“Prepared”}, rm \mapsto r]\} \\ & \wedge \text{UNCHANGED } \langle tmState, tmPrepared \rangle \end{aligned}$$

$RMChooseToAbort(r) \triangleq$

Resource manager *r* spontaneously decides to abort. As noted above, *r* does not send any message in our simplified spec.

$$\begin{aligned} & \wedge rmState[r] = \text{“working”} \\ & \wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{“aborted”}] \\ & \wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle \end{aligned}$$

$RMRecvCommitMsg(r) \triangleq$

Resource manager *r* is told by the *TM* to commit.

$$\begin{aligned} & \wedge [type \mapsto \text{“Commit”}] \in msgs \\ & \wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{“committed”}] \\ & \wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle \end{aligned}$$

$RMRecvAbortMsg(r) \triangleq$

Resource manager r is told by the TM to abort.

$\wedge [type \mapsto \text{"Abort"}] \in msgs$
 $\wedge rmState' = [rmState \text{ EXCEPT } ![r] = \text{"aborted"}]$
 $\wedge \text{UNCHANGED } \langle tmState, tmPrepared, msgs \rangle$

$TPNext \triangleq$
 $\vee TMCommit \vee TMAbort$
 $\vee \exists r \in RM :$
 $TMRcvPrepared(r) \vee RMPPrepare(r) \vee RMChooseToAbort(r)$
 $\vee RMRcvCommitMsg(r) \vee RMRcvAbortMsg(r)$

The material below this point is not discussed in Video Lecture 6. It will be explained in Video Lecture 8.

$TPSpec \triangleq TPInit \wedge \Box [TPNext]_{\langle rmState, tmState, tmPrepared, msgs \rangle}$

The complete spec of the Two-Phase *Commit* protocol.

THEOREM $TPSpec \Rightarrow \Box TPTYPEOK$

This theorem asserts that the type-correctness predicate $TPTYPEOK$ is an invariant of the specification.

We now assert that the Two-Phase *Commit* protocol implements the Transaction *Commit* protocol of module $TCommit$. The following statement imports all the definitions from module $TCommit$ into the current module.

INSTANCE $TCommit$

THEOREM $TPSpec \Rightarrow TCSpec$

This theorem asserts that the specification $TPSpec$ of the Two-Phase Commit protocol implements the specification $TCSpec$ of the Transaction *Commit* protocol.

The two theorems in this module have been checked with TLC for six RM s, a configuration with 50816 reachable states, in a little over a minute on a 1 GHz PC .

* Modification History
* Last modified Wed Nov 13 20:33:39 CET 2019 by martin
* Last modified Tue Jul 17 11:38:20 CEST 2018 by max
* Created Tue Jul 17 11:37:58 CEST 2018 by max