MODULE *OneUpdateMeta*

This module is a formal specification of 1-Update,
a hybrid invalidate and update cache coherence protocol that appears in *PACT'*21.
This spec actually includes two variants of 1-Update one in which acks for updates
are gathered directly by the directory (the main variant discussed in the paper);
and another variant where acks for updates are gathered by the writer itself.
Setting the constant *ENABLE_DIR_ACKS* TRUE or FALSE verifies either variant accordingly.

EXTENDS *Integers*, *FiniteSets*

CONSTANTS *CORES*,
MAX_WRITES,
WRITE_TO_UPDATE, *I.e.*, number of write on which we will trigger the Update
ENABLE_DIR_ACKS - If TRUE: update acks are gathered by the directory.
- If FALSE: update acks are gathered by the writer.

VARIABLES variable prefixes -> *g*:global, d: directory, c: cache/core | *VECTORS* indexed by cache/*core_id*
GLOBAL variables
Msgs,
gBcstMsg,
gBcstMsgRcvers,
Dir variables
dOwner,
dSharers, No sharers/owner: $.readers = \{\}$ / $.owner = 0$
dReqPending,
dState,
dRcvAcks,
Cache/core variables
cState,
cRcvAcks,
data is a monotonically increasing int to check correctness invariants
cData,
mData Memory data

$vars \triangleq \langle dOwner,\ dSharers,\ dReqPending,\ dState,\ dRcvAcks,$
$cState,\ cRcvAcks,\ cData,$
$mData,\ Msgs,\ gBcstMsg,\ gBcstMsgRcvers\rangle$

Helper Definitions
$EMPTY\_OWNER \triangleq 0$

Assumptions
ASSUME $Cardinality(CORES) > 0$ assume atleast 1 cache
ASSUME $MAX\_WRITES > WRITE\_TO\_UPDATE$ ensure we always have enough writes to trigger an update
ASSUME $EMPTY\_OWNER \notin CORES$ id used for *EMPTY_ONWER* should not be used to identify a CORE
ASSUME $ENABLE\_DIR\_ACKS \in \{\text{TRUE, FALSE}\}$

Useful Unchanged shortcuts

$unchanged\_g \triangleq \text{UNCHANGED } \langle gBcstMsg, gBcstMsgRcvers \rangle$
$unchanged\_m \triangleq \text{UNCHANGED } \langle mData \rangle$
$unchanged\_c \triangleq \text{UNCHANGED } \langle cState, cRcvAcks, cData \rangle$
$unchanged\_d \triangleq \text{UNCHANGED } \langle dOwner, dSharers, dReqPending, dState, dRcvAcks \rangle$
$unchanged\_dm \triangleq unchanged\_d \land unchanged\_m$
$unchanged\_cm \triangleq unchanged\_c \land unchanged\_m$
$unchanged\_cd \triangleq unchanged\_c \land unchanged\_d$
$unchanged\_mcd \triangleq unchanged\_c \land unchanged\_d \land unchanged\_m$

$unchanged\_gm \triangleq unchanged\_g \land unchanged\_m$
$unchanged\_gmc \triangleq unchanged\_c \land unchanged\_gm$
$unchanged\_gmd \triangleq unchanged\_d \land unchanged\_gm$

$unchanged\_Msgs \triangleq \text{UNCHANGED } \langle Msgs \rangle$
$unchanged\_mMsgs \triangleq unchanged\_Msgs \land unchanged\_m$
$unchanged\_cMsgs \triangleq unchanged\_Msgs \land unchanged\_c$
$unchanged\_dMsgs \triangleq unchanged\_Msgs \land unchanged\_d$
$unchanged\_dmMsgs \triangleq unchanged\_Msgs \land unchanged\_dm$
$unchanged\_cmMsgs \triangleq unchanged\_Msgs \land unchanged\_cm$
$unchanged\_cdMsgs \triangleq unchanged\_Msgs \land unchanged\_cd$
$unchanged\_mcdMsgs \triangleq unchanged\_Msgs \land unchanged\_mcd$

$unchanged\_gMsgs \triangleq unchanged\_g \land unchanged\_Msgs$
$unchanged\_gmMsgs \triangleq unchanged\_g \land unchanged\_mMsgs$
$unchanged\_gcMsgs \triangleq unchanged\_g \land unchanged\_cMsgs$
$unchanged\_gdMsgs \triangleq unchanged\_g \land unchanged\_dMsgs$
$unchanged\_gdmMsgs \triangleq unchanged\_g \land unchanged\_dmMsgs$
$unchanged\_gcmMsgs \triangleq unchanged\_g \land unchanged\_cmMsgs$
$unchanged\_gcdMsgs \triangleq unchanged\_g \land unchanged\_cdMsgs$
$unchanged\_gmcdMsgs \triangleq unchanged\_g \land unchanged\_mcdMsgs$

Type definitions

$Type\_binary \triangleq 0 \,..\, 1$
$Type\_Data \triangleq 0 \,..\, MAX\_WRITES$
$Type\_State \triangleq \{\text{“M”}, \text{“O”}, \text{“E”}, \text{“S”}, \text{“I”}\}$ all nodes start from I

Msgs send by requester

$Type\_rMsg \triangleq$
$\quad [type : \{\text{“GetS”}, \text{“GetM”}\}, sender : CORES]$

$Type\_uMsg \triangleq$
$\quad [type : \{\text{“Upd”}\}, \quad data : Type\_Data,$
$\qquad\qquad\qquad\quad sender \;\; : CORES,$
$\qquad\qquad\qquad\quad receiver \; : CORES]$

Msgs send by directory

$Type\_iMsg \triangleq$
$[type : \{\text{“DInv”}\}, \quad sender : CORES, \quad receiver : CORES]$ initial sender (*i.e.*, requester)

$Type\_dMsg \triangleq Type\_iMsg \cup$
$[type : \{\text{“Fwd-GetM”}, \text{“Fwd-GetS”}\}, \quad sender : CORES, \quad receiver : CORES]$ initial sender (*i.e.*, requester)

Msgs send by sharer
$Type\_sMsg \triangleq$
$[type : \{\text{“SAck, UAck”}\}, \quad sender : CORES, \quad receiver : CORES]$
$\cup$
$[type : \{\text{“SData”}, \text{“SAckData”}\}, \quad data : Type\_Data, \quad sender : CORES, \quad receiver : CORES]$

$Type\_bcastMsg \triangleq Type\_uMsg \cup Type\_iMsg$
$Type\_msg \triangleq Type\_sMsg$
$\cup Type\_rMsg$
$\cup Type\_uMsg$
$\cup Type\_iMsg$
$\cup Type\_dMsg$
$\cup Type\_sMsg$

Type check and initialization

$ATypeOK \triangleq$ The type correctness invariant
GLOBAL variables
$\land Msgs \subseteq Type\_msg$
$\land gBcstMsg \in Type\_bcastMsg$
$\land gBcstMsgRcvers \subseteq CORES$
Directory/memory variables
$\land dOwner \in CORES$
$\land dSharers \subseteq CORES$
$\land dRcvAcks \subseteq CORES$
$\land dReqPending \in Type\_binary$
$\land dState \in Type\_State$
$\land cState \in Type\_State$
$\land mData \in Type\_Data$
Core/cache variables
$\land \forall n \in CORES : cData[n] \in Type\_Data$
$\land \forall n \in CORES : cState[n] \in Type\_State$

$$\land \forall\, n \in CORES : cRcvAcks[n] \quad \subseteq (CORES \setminus \{n\})$$

$AInit \triangleq$ The initial predicate

GLOBAL variables

$\land\ Msgs = \{\}$
$\land\ gBcstMsg = \{\}$
$\land\ gBcstMsgRcvers = \{\}$

Directory/memory variables

$\land\ mData = 0$
$\land\ dState = \text{“I”}$
$\land\ dOwner = EMPTY\_OWNER$
$\land\ dSharers = \{\}$
$\land\ dRcvAcks = \{\}$
$\land\ dReqPending = 0$

Core/cache variables

$\land\ cData = [n \in CORES \mapsto 0]$
$\land\ cRcvAcks = [n \in CORES \mapsto \{\}]$
$\land\ cState = [n \in CORES \mapsto \text{“I”}]$

TODO may add sanity check that m already exists in the set before delivering it

$deliver\_Msg(m) \triangleq Msgs' = Msgs \setminus \{m\}$

TODO may add all messages to one set from which we do not remove msgs for debugging (w/ a global counter)

$\_send\_Msg(m) \triangleq Msgs' = Msgs \cup \{m\}$

$\_send\_Msg\_with\_data(\_type, \_sender, \_receiver, \_data) \triangleq$
$\quad \_send\_Msg([type \mapsto \_type, data \mapsto \_data, sender \mapsto \_sender, receiver \mapsto \_receiver])$

$\_send\_Msg\_simple(\_type, \_requester, \_receiver) \triangleq$
$\quad \_send\_Msg([type \mapsto \_type, sender \mapsto \_requester, receiver \mapsto \_receiver])$

$\_resp\_Msg(m, new\_m) \triangleq Msgs' = (Msgs \setminus \{m\}) \cup \{new\_m\}$

$\_resp\_Msg\_simple(m, \_type) \triangleq$
$\quad \_resp\_Msg(m, [type \mapsto \_type, sender \mapsto m.receiver, receiver \mapsto m.sender])$

$\_resp\_Msg\_with\_data(m, \_type) \triangleq$
$\quad \_resp\_Msg(m, [type \mapsto \_type, data \mapsto cData[m.receiver], sender \mapsto m.receiver,$

$$receiver \;\mapsto m.sender])$$

$$\begin{array}{ll} resp\_UAck(m) & \triangleq \; \_resp\_Msg\_simple(m,\ \text{“UAck”}) \\ resp\_SAck(m) & \triangleq \; \_resp\_Msg\_simple(m,\ \text{“SAck”}) \\ resp\_SData(m) & \triangleq \; \_resp\_Msg\_with\_data(m,\ \text{“SData”}) \\ resp\_SDataAck(m) & \triangleq \; \_resp\_Msg\_with\_data(m,\ \text{“SDataAck”}) \end{array}$$

$$\begin{array}{l} ucst\_FwdGetM(\_requester,\ \_receiver) \triangleq \\ \qquad \_send\_Msg\_simple(\text{“Fwd-GetM”},\ \_requester,\ \_receiver) \\ ucst\_FwdGetS(\_requester,\ \_receiver) \triangleq \\ \qquad \_send\_Msg\_simple(\text{“Fwd-GetS”},\ \_requester,\ \_receiver) \end{array}$$

$$\begin{array}{l} \_bcst\_msg(\_requester,\ \_receivers,\ \_msg) \triangleq \\ \qquad \text{LET } rcver \triangleq \text{CHOOSE } x \in \_receivers : \text{TRUE IN} \\ \qquad\qquad \land\ gBcstMsgRcvers' = \_receivers \setminus \{rcver\} \\ \qquad\qquad \land\ gBcstMsg' = \{[\_msg \text{ EXCEPT } !.receiver = rcver]\} \end{array}$$

$$\begin{array}{l} bcst\_DInv(\_requester,\ \_receivers) \triangleq \\ \qquad \_bcst\_msg(\_requester,\ \_receivers, \\ \qquad\qquad [type \;\mapsto \text{“DInv”}, \\ \qquad\qquad\ sender \;\mapsto \_requester, \\ \qquad\qquad\ receiver \mapsto 0]) \end{array}$$

$$\begin{array}{l} bcst\_Upd(\_requester,\ \_receivers,\ \_data) \triangleq \\ \qquad \_bcst\_msg(\_requester,\ \_receivers, \\ \qquad\qquad [type \;\mapsto \text{“Upd”}, \\ \qquad\qquad\ data \;\mapsto \_data, \\ \qquad\qquad\ sender \;\mapsto \_requester, \\ \qquad\qquad\ receiver \mapsto 0]) \end{array}$$

$$\begin{array}{l} rcv\_unicast(m,\ receiver,\ type) \triangleq \\ \qquad \land\ m.type = type \\ \qquad \land\ m.receiver = receiver \end{array}$$

$$\begin{array}{ll} rcv\_UAck(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“UAck”}) \\ rcv\_SAck(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“SAck”}) \\ rcv\_SData(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“SData”}) \\ rcv\_SDataAck(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“SDataAck”}) \end{array}$$

$$\begin{array}{ll} rcv\_Upd(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“Upd”}) \\ rcv\_DInv(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“DInv”}) \end{array}$$

$$\begin{array}{ll} rcv\_FwdGetM(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“Fwd-GetM”}) \\ rcv\_FwdGetS(m,\ receiver) & \triangleq \; rcv\_unicast(m,\ receiver,\ \text{“Fwd-GetS”}) \end{array}$$

Helper functions

$is\_M(n) \triangleq cState[n] = \text{“M”}$
$is\_O(n) \triangleq cState[n] = \text{“O”}$
$is\_E(n) \triangleq cState[n] = \text{“E”}$
$is\_S(n) \triangleq cState[n] = \text{“S”}$
$is\_I(n) \triangleq cState[n] = \text{“I”}$

$dir\_rcved\_acks\_from\_set(set) \triangleq set \subseteq dRcvAcks$
$rcved\_acks\_from\_set(n, set) \triangleq set \subseteq cRcvAcks[n]$
$rcved\_all\_sharer\_acks(n) \triangleq (dSharers \setminus \{n\}) \subseteq cRcvAcks[n]$

$has\_valid\_data(n) \triangleq \neg is\_I(n)$
$set\_next\_data\_value(n) \triangleq cData' = [cData \text{ EXCEPT } ![n] = cData[n] + 1]$
$has\_not\_reached\_final\_value \triangleq \forall\, n \in CORES : cData[n] < MAX\_WRITES + 1$

todo check the correctness of the following

$is\_sharer(n) \triangleq \neg is\_I(n)$
$is\_exclusive(n) \triangleq is\_M(n) \lor is\_E(n)$
$is\_owner(n) \triangleq is\_O(n) \lor is\_M(n)$

$upd\_core\_data(n, \_data) \triangleq cData' = [cData \text{ EXCEPT } ![n] = \_data]$
$rd\_mem\_data(n) \triangleq upd\_core\_data(n, mData)$
$upd\_mem\_data(n) \triangleq mData' = cData[n]$

$$Min(S) \triangleq \text{CHOOSE } x \in S : \forall\, y \in S \setminus \{x\} : y > x$$

Protocol Invariants:

memory data consistency invariant

$$MEM\_DATA\_CONSISTENT \triangleq \lor \exists\, n \in CORES : is\_owner(n) \lor \forall\, n \in CORES : cData[n] \leq mData$$

All valid core/cache data are consistent

$$CORE\_DATA\_CONSISTENT \triangleq \forall\, o, k \in CORES : \lor \neg is\_I(o) \lor \land cData[o] \geq mData \land cData[o] \geq cData[k]$$

There is always at most one owner

$$AT\_MOST\_ONE\_OWNER \triangleq \forall\, n, m \in CORES : \lor m = n \lor \neg is\_owner(n) \lor \neg is\_owner(m)$$

$$
\begin{array}{l}
IF\_EXLUSIVE\_REST\_INV \triangleq \\
\quad \lor \neg \exists\, n \in CORES : is\_exclusive(n) \\
\quad \lor \ \forall\, n \ \in CORES : \lor is\_I(n) \\
\qquad\qquad\qquad\qquad\ \lor is\_exclusive(n)
\end{array}
$$

$$
\begin{array}{l}
CONSISTENT\_OWNER \triangleq \\
\quad \forall\, n \in CORES : \lor dOwner = EMPTY\_OWNER \\
\qquad\qquad\qquad\ \lor dReqPending = 1 \\
\qquad\qquad\qquad\ \lor cState[dOwner] = dState
\end{array}
$$

Directory correctly indicates owner and sharers

$$
\begin{array}{l}
CONSISTENT\_DIRECTORY\_OWNER \triangleq \\
\quad \forall\, n \in CORES : \lor dOwner = n \\
\qquad\qquad\qquad\ \lor \neg is\_owner(n)
\end{array}
$$

$$
\begin{array}{l}
CONSISTENT\_DIRECTORY\_SHARERS \triangleq \\
\quad \forall\, k \in CORES : \lor is\_I(k) \\
\qquad\qquad\qquad\ \lor k \in dSharers
\end{array}
$$

The owner and readers are always correctly reflected by any valid sharing vectors

$$
\begin{array}{l}
INVARIANTS \triangleq \\
\quad \land MEM\_DATA\_CONSISTENT \\
\quad \land CORE\_DATA\_CONSISTENT \\
\quad \land AT\_MOST\_ONE\_OWNER \\
\quad \land IF\_EXLUSIVE\_REST\_INV \\
\quad \land CONSISTENT\_OWNER \\
\quad \land CONSISTENT\_DIRECTORY\_OWNER \\
\quad \land CONSISTENT\_DIRECTORY\_SHARERS
\end{array}
$$