MODULE *OneUpdateMeta*

EXTENDS *Integers*, *FiniteSets*

CONSTANTS *CORES*,
MAX_WRITES,
WRITE_TO_UPDATE *I.e.*, number of write on which we will trigger the Update

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,
Cache/core variables
cState,
cRcvAcks,
data is a monotonically increasing int to check correctness invariants
cData,
mData Memory data

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

Helper Definitions
$EMPTY\_OWNER \triangleq 0$

Assumptions
ASSUME $Cardinality(CORES) > 0$ assume atleast 1 cache
ASSUME $EMPTY\_OWNER \notin CORES$ id used for *EMPTY_ONWER* should not be used to identify a CORE

Useful Unchanged shortcuts
$unchanged\_g \triangleq$ UNCHANGED $\langle gBcstMsg, gBcstMsgRcvers\rangle$
$unchanged\_m \triangleq$ UNCHANGED $\langle mData\rangle$
$unchanged\_c \triangleq$ UNCHANGED $\langle cState, cRcvAcks, cData\rangle$
$unchanged\_d \triangleq$ UNCHANGED $\langle dOwner, dSharers, dReqPending, dState\rangle$
$unchanged\_dm \triangleq unchanged\_d \land unchanged\_m$
$unchanged\_cm \triangleq unchanged\_c \land unchanged\_m$
$unchanged\_cd \triangleq unchanged\_c \land unchanged\_d$
$unchanged\_mcd \triangleq unchanged\_c \land unchanged\_d \land unchanged\_m$

$$
\begin{array}{ll}
unchanged\_gm & \triangleq\ unchanged\_g \land unchanged\_m \\
unchanged\_gmc & \triangleq\ unchanged\_c \land unchanged\_gm \\
unchanged\_gmd & \triangleq\ unchanged\_d \land unchanged\_gm
\end{array}
$$

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

$$
\begin{array}{ll}
unchanged\_gMsgs & \triangleq\ unchanged\_g \land unchanged\_Msgs \\
unchanged\_gmMsgs & \triangleq\ unchanged\_g \land unchanged\_mMsgs \\
unchanged\_gcMsgs & \triangleq\ unchanged\_g \land unchanged\_cMsgs \\
unchanged\_gdMsgs & \triangleq\ unchanged\_g \land unchanged\_dMsgs \\
unchanged\_gdmMsgs & \triangleq\ unchanged\_g \land unchanged\_dmMsgs \\
unchanged\_gcmMsgs & \triangleq\ unchanged\_g \land unchanged\_cmMsgs \\
unchanged\_gcdMsgs & \triangleq\ unchanged\_g \land unchanged\_cdMsgs \\
unchanged\_gmcdMsgs & \triangleq\ unchanged\_g \land unchanged\_mcdMsgs
\end{array}
$$

Type definitions

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

Msgs send by requester

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

$$
\begin{array}{lll}
Type\_uMsg\ \triangleq & & \\
\quad [type : \{\text{“Upd”}\}, & data & : Type\_Data, \\
& sender & : CORES, \\
& receiver & : CORES]
\end{array}
$$

Msgs send by directory

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

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

Msgs send by sharer

$$
\begin{array}{l}
Type\_sMsg \triangleq \\
\quad [type : \{\text{“SAck, UAck”}\}, \quad sender \ : CORES, \\
\qquad\qquad\qquad\qquad\quad receiver \ : CORES] \\
\quad \cup \\
\quad [type : \{\text{“SData”}, \\
\qquad\quad \text{“SAckData”}\}, \qquad data : Type\_Data, \\
\qquad\qquad\qquad\qquad\quad sender \ : CORES, \\
\qquad\qquad\qquad\qquad\quad receiver \ : CORES]
\end{array}
$$

$$
\begin{array}{l}
Type\_bcastMsg \triangleq Type\_uMsg \cup Type\_iMsg \\
Type\_msg \triangleq Type\_sMsg \\
\qquad \cup Type\_rMsg \\
\qquad \cup Type\_uMsg \\
\qquad \cup Type\_iMsg \\
\qquad \cup Type\_dMsg \\
\qquad \cup Type\_sMsg
\end{array}
$$

Type check and initialization

$$
\begin{array}{l}
ATypeOK \triangleq \quad \text{The type correctness invariant} \\
\qquad\qquad \text{GLOBAL variables} \\
\qquad\quad \land Msgs \quad\quad\quad \subseteq Type\_msg \\
\qquad\quad \land gBcstMsg \quad\quad \in \ Type\_bcastMsg \\
\qquad\quad \land gBcstMsgRcvers \subseteq CORES \\
\qquad\qquad \text{Directory/memory variables} \\
\qquad\quad \land dOwner \quad\quad\ \in CORES \\
\qquad\quad \land dSharers \quad\quad \subseteq CORES \\
\qquad\quad \land dReqPending \quad \in Type\_binary \\
\qquad\quad \land dState \quad\quad\ \in Type\_State \\
\qquad\quad \land cState \quad\quad\ \in Type\_State \\
\qquad\quad \land mData \quad\quad\ \in Type\_Data \\
\qquad\qquad \text{Core/cache variables} \\
\qquad\quad \land \forall\, n \in CORES : cData[n] \quad\quad \in Type\_Data \\
\qquad\quad \land \forall\, n \in CORES : cState[n] \quad\quad \in Type\_State \\
\qquad\quad \land \forall\, n \in CORES : cRcvAcks[n] \quad \subseteq (CORES \setminus \{n\})
\end{array}
$$

$$
\begin{array}{l}
AInit \triangleq \quad \text{The initial predicate} \\
\qquad\qquad \text{GLOBAL variables} \\
\qquad\quad \land Msgs \quad\quad\quad = \{\} \\
\qquad\quad \land gBcstMsg \quad\quad = \{\} \\
\qquad\quad \land gBcstMsgRcvers = \{\} \\
\qquad\qquad \text{Directory/memory variables} \\
\qquad\quad \land mData \quad\quad = 0 \\
\qquad\quad \land dState \quad\quad = \text{“I”} \\
\qquad\quad \land dOwner \quad\quad = EMPTY\_OWNER
\end{array}
$$

$\land\ dSharers = \{\}$
$\land\ dReqPending = 0$

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)

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

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

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

$\_resp\_Msg(m, new\_m) \triangleq Msgs' = (Msgs \setminus \{m\}) \cup \{new\_m\}$
$\_resp\_Msg\_simple(m, \_type) \triangleq$
  $\_resp\_Msg(m, [type \mapsto \_type, sender \mapsto m.receiver, receiver \mapsto m.sender])$
$\_resp\_Msg\_with\_data(m, \_type) \triangleq$
  $\_resp\_Msg(m, [type \mapsto \_type, data \mapsto cData[m.receiver], sender \mapsto m.receiver, receiver \mapsto m.sender])$

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

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

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

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

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

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

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

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

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

Helper functions

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

$$
\begin{array}{ll}
rcved\_acks\_from\_set(n,\ set) & \triangleq\ set \subseteq cRcvAcks[n] \\
rcved\_all\_sharer\_acks(n) & \triangleq\ (dSharers \setminus \{n\}) \subseteq cRcvAcks[n]
\end{array}
$$

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

todo check the correctness of the following

$$
\begin{array}{lcl}
is\_sharer(n) & \triangleq & \neg is\_I(n) \\
is\_exclusive(n) & \triangleq & is\_M(n) \lor is\_E(n) \\
is\_owner(n) & \triangleq & is\_O(n) \lor is\_M(n)
\end{array}
$$

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

$$
\begin{array}{l}
Min(S) \triangleq \text{CHOOSE } x \in S : \\
\qquad \forall\, y \in S \setminus \{x\} : \\
\qquad\quad y > x
\end{array}
$$

Protocol *Invariants*:

memory data consistency invariant

$$
\begin{array}{l}
MEM\_DATA\_CONSISTENT \triangleq \\
\qquad \lor \exists\, n \in CORES : is\_owner(n) \\
\qquad \lor \forall\, n \in CORES : cData[n] \leq mData
\end{array}
$$

All valid core/cache data are consistent

$$
\begin{array}{l}
CORE\_DATA\_CONSISTENT \triangleq \\
\quad \forall\, o, k \in CORES : \lor \neg is\_I(o) \\
\qquad\qquad\qquad\qquad \lor \land cData[o] \geq mData \\
\qquad\qquad\qquad\qquad\quad \land cData[o] \geq cData[k]
\end{array}
$$

There is always at most one owner

$$
\begin{array}{l}
AT\_MOST\_ONE\_OWNER \triangleq \\
\quad \forall\, n, m \in CORES : \lor m = n \\
\qquad\qquad\qquad\qquad \lor \neg is\_owner(n) \\
\qquad\qquad\qquad\qquad \lor \neg is\_owner(m)
\end{array}
$$

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

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

Directory correctly indicates owner and sharers

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

$$CONSISTENT\_DIRECTORY\_SHARERS \triangleq$$
$$\forall\, k \in CORES : \lor\, is\_I(k)$$
$$\lor\, k \in dSharers$$

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

$$INVARIANTS \triangleq$$
$$\land\, MEM\_DATA\_CONSISTENT$$
$$\land\, CORE\_DATA\_CONSISTENT$$
$$\land\, AT\_MOST\_ONE\_OWNER$$
$$\land\, IF\_EXLUSIVE\_REST\_INV$$
$$\land\, CONSISTENT\_OWNER$$
$$\land\, CONSISTENT\_DIRECTORY\_OWNER$$
$$\land\, CONSISTENT\_DIRECTORY\_SHARERS$$