MSI: \{M, S, I\}
- M: Modifiable, Exclusive copy, cannot be present in any other cache
- S: Shared copy, can also be present in other caches
- I: Not present in this cache, can be present in other caches

M > S > I

Cache states:
c.state(a): sibling info – M|S|I
c.child[c_k](a): child c_k info – M|S|I
c.waitp(a): Denotes if cache c is waiting for a response from its parent. If so, what type of response
- Nothing means not waiting
- Valid (M|S|I) means waiting for response to go to M or S or I as the case may be
c.waitc[c_k](a): Denotes if cache c is waiting for a response from its child c_k. If so, what type of response
- Nothing | Valid (M|S|I)

IsCompatible:
The states x, y of two sibling caches are compatible iff IsCompatible(x, y) is True where
IsCompatible(M, M) = False
IsCompatible(M, S) = False
IsCompatible(S, M) = False
All other cases = True

Messages:
Parent to Child:
\(<c, m, M2C_{Req}, a, y>\): Parent m requesting a child c to downgrade the state of location a to y
\(<c, m, M2C_{Rep}, a, x, y, data>\): Parent m sending a notification to child c to upgrade the state of location a from x to y

Child to Parent:
\(<m, c, C2M_{Req}, a, y>\): Child c requesting the parent m to upgrade the state of location a to y
\(<m, c, C2M_{Rep}, a, x, y, data>\): Child c sending a notification to parent m saying it has downgraded the state of location a from x to y
**DataTransfer:**

*Child to Parent data transfer:* Downgrade of a child from M to S and M to I requires that the dirty data for that location also gets transferred to the parent

\[
\text{DataTransfer}(M, S) = \text{True} \\
\text{DataTransfer}(M, I) = \text{True} \\
\text{DataTransfer}(S, I) = \text{False}
\]

*Parent to child data transfer:* Upgrade initiated by the parent usually requires a data transfer except in the case of S to M because the child already has the data

\[
\text{DataTransfer}(I, S) = \text{True} \\
\text{DataTransfer}(I, M) = \text{True} \\
\text{DataTransfer}(S, M) = \text{False}
\]

**Processor rules:**

**Load-hit rule**

\[\text{inst is (Load a) \& c.state(a) is S or M}\]  
\[\rightarrow \text{p2m.deq; m2p.enq(c.data(a));}\]

**Store-hit rule**

\[\text{inst is (Store a v) \& c.state(a) is M}\]  
\[\rightarrow \text{p2m.deq; m2p.enq(Ack); c.data(a):=v;}\]

**Types of Actions:**

A protocol specifies cache actions corresponding to each of these 8 requests and responses
### Sending Requests

1. **Child sending Upgrade-to-y req**  
   \[(c.\text{state}(a)<y) \& (c.\text{waitp}(a)==\text{Nothing})\]  
   \[\rightarrow\ c.\text{waitp}(a):=\text{Valid y};\]  
   \[\text{c2m.enq(<m, c, C2M_Req, a, y>);}\]

2. **Parent sending downgrade to y req**  
   \[(m.\text{child}[i](a)>y) \& (m.\text{waitc}[i](a)==\text{Nothing})\]  
   \[\rightarrow m.\text{waitc}[i](a):=\text{Valid y};\]  
   \[\text{m2c.enq(<i, m, M2C_Req, a, y>);}\]

### Dequeueing Requests

3. **Child dequeuing Downgrade-to-y req**  
   \[(\text{m2c.msg}=<c, m, M2C_Req, a, y>) \& (c.\text{state}(a)\leq y)\]  
   \[\rightarrow m2c.deq;\]

4. **Parent dequeuing Upgrade-to-y req**  
   \[(c.\text{msg}=<m, c, C2M_Req, a, y>) \& (m.\text{child}[c](a)\geq y)\]  
   \[\rightarrow c2m.deq;\]

### Sending Responses

5. **Child sending Downgrade-from-x-to-y rep**  
   \[(c.\text{state}(a)=x) \& (y<x) \& (\forall i, c.\text{child}[i](a)\leq y)\]  
   \[\rightarrow c.\text{state}(a):=y;\]  
   \[\text{c2m.enq(<m, c, C2M_Rep, a, x, y,}\]  
   \[\text{if DataTransfer(x,y) then c.data(a)}\]  
   \[\text{else _>);}\]

6. **Parent sending Upgrade-from-x-to-y rep**  
   \[(m.\text{child}[c](a)=x) \& (y>x) \& (m.\text{state}(a)\geq y)\]  
   \[\rightarrow m.\text{child}[c](a):=y;\]  
   \[\text{m2c.enq(<c, m, M2C_Rep, a, x, y,}\]  
   \[\text{if DataTransfer(x,y) then m.data(a)}\]  
   \[\text{else _>);}\]

### Receiving Responses

7. **Child receiving Upgrade-from-x-to-y rep**  
   \[\text{m2c.msg}=<c, m, M2C_Rep, a, x, y, data>\]  
   \[\rightarrow m2c.deq;\]  
   \[\text{if c.\text{state}(a)==x then}\]  
   \[\text{c.\text{state}(a):=y;}\]  
   \[\text{if DataTransfer(x,y) then c.data(a):=data;}\]  
   \[\text{if c.\text{waitp}(a) \leq (Valid y)\]  
   \[\text{then c.\text{waitp}(a):=\text{Nothing};}\}

8. **Parent receiving Downgrade-from-x-to-y rep**  
   \[\text{c2m.msg}=<m, c, C2M_Rep, a, x, y, data>\]  
   \[\rightarrow c2m.deq;\]  
   \[\text{m.\text{child}[c](a):=y;}\]  
   \[\text{if DataTransfer(x,y) then m.data(a):=data;}\]  
   \[\text{if m.\text{waitc}[c](a) \geq (Valid y)\]  
   \[\text{then m.waitc}[c](a):=\text{Nothing;}\]