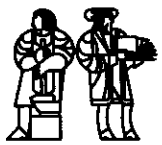
LABORATORY FOR
COMPUTER SCIENCE



MASSACHUSETTS
INSTITUTE OF
TECHNOLOGY

# An Adaptive Cache Coherence Protocol That Implements Sequential Consistency for DSM Systems With Multi-level Caches

Computation Structures Group Memo 404
April 1, 1998

**Xiaowei Shen and Arvind**

This paper describes research done at the Laboratory for Computer Science of the Massachusetts Institute of Technology. Funding for this work is provided in part by the Advanced Research Projects Agency of the Department of Defense under the Office of Naval Research contract N00014-92-J-1310 and Ft Huachuca contract DABT63-95-C-0150.

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

# An Adaptive Cache Coherence Protocol That Implements Sequential Consistency for DSM Systems With Multi-level Caches

Xiaowei Shen and Arvind

April 1, 1998

# 1 HCN-opt: An Optimized Protocol

The HCN-opt protocol is an optimized protocol based on HCN-base. While most of HCN-base rules remain unchanged in HCN-opt, several HCN-base rules are slightly modified, and a number of new rules are added. HCN-opt incorporates three new protocol messages, Upgrade-rep, Pushout-rep and Pushout-req.

Upgrade-rep is Ex-rep without data. It can be used to upgrade a shared cell to an exclusive cell. A common scenario is that when a memory receives an Ex-req from a child memory, the directory shows that the child has a shared copy. In HCN-base, the parent memory first invalidates all shared copies cached by its children, and then sends an Ex-rep message to the requesting child memory. An HCN-opt, the parent memory only invalidates, if any, other shared copies cached by its other child memories, and then sends an Upgrade-rep message to the requesting child memory.

Pushout-rep is a combination of Wb-rep and Inv-rep. In HCN-base, when a memory receives an Ex-req, if currently a child memory has the most up-to-date data and the ownership, the parent first sends a Wb-req message to the child to enforce the data to be written back, and then sends an Inv-req message to the child to have the shared copy left be invalidated. In HCN-opt, the parent simply sends a Pushout-req message.

In HCN-opt, the six reply messages form three pairs, Sh-rep/InvRep, Ex-rep/Pushout-rep and Upgrade-rep/Wb-rep, where the two messages in each pair can cause inverse actions each other. The following table gives the initiator (suspended instruction or message) and the expected reply when a request message is issued.

| Request Issued | Initiator | Expected Reply |
|---|---|---|
| Sh-req | Load<br>Sh-req | Sh-rep |
| Ex-req | Store<br>Ex-req | Ex-rep / Upgrade-rep |
| Wb-req | Sh-req<br>Wb-req | Wb-rep |
| Pushout-req | Ex-req<br>Pushout-req | Pushout-rep |
| Inv-req | Ex-req<br>Pushout-req<br>Inv-req | Inv-rep |

## 1.1 Memory Access Rules

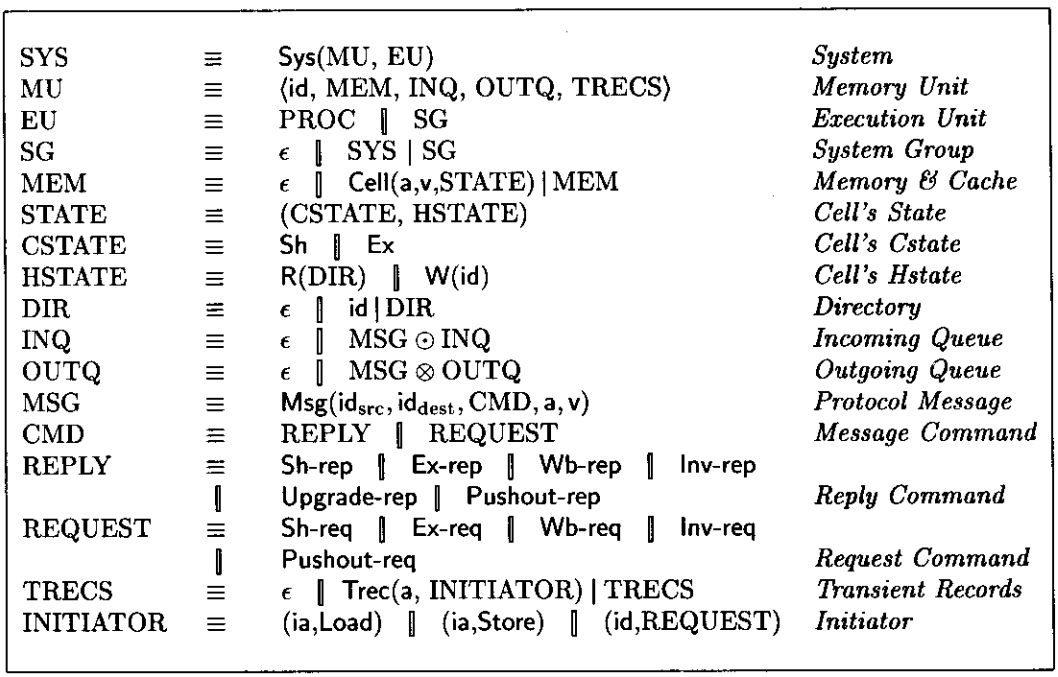
### 1.1.1 Cache-Hit Rules

*Read-Cache-Hit Rule*

$$\begin{aligned} &\mathsf{Sys}(\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{cs},\mathsf{R}(\epsilon))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog})) \\ &\qquad \textit{if}\ \ \mathsf{prog}[\mathsf{ia}] = \mathsf{r} := \mathsf{Load}(\mathsf{r}_1)\ \ \textit{and}\ \ \mathsf{a} = \mathsf{rf}[\mathsf{r}_1] \\ \longrightarrow\ &\mathsf{Sys}(\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{cs},\mathsf{R}(\epsilon))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}+1, \mathsf{rf}[\mathsf{r} := \mathsf{v}], \mathsf{prog})) \end{aligned}$$

*Write-Cache-Hit Rule*

$$\begin{aligned} &\mathsf{Sys}(\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{u},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog})) \\ &\qquad \textit{if}\ \ \mathsf{prog}[\mathsf{ia}] = \mathsf{Store}(\mathsf{r}_1, \mathsf{r}_2)\ \ \textit{and}\ \ \mathsf{a} = \mathsf{rf}[\mathsf{r}_1] \\ \longrightarrow\ &\mathsf{Sys}(\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}+1, \mathsf{rf}, \mathsf{prog}))\quad \textit{where}\ \ \mathsf{v} = \mathsf{rf}[\mathsf{r}_2] \end{aligned}$$



| | | | |
|---|---|---|---|
| SYS | $\equiv$ | Sys(MU, EU) | *System* |
| MU | $\equiv$ | ⟨id, MEM, INQ, OUTQ, TRECS⟩ | *Memory Unit* |
| EU | $\equiv$ | PROC ‖ SG | *Execution Unit* |
| SG | $\equiv$ | $\epsilon$ ‖ SYS \| SG | *System Group* |
| MEM | $\equiv$ | $\epsilon$ ‖ Cell(a,v,STATE) \| MEM | *Memory & Cache* |
| STATE | $\equiv$ | (CSTATE, HSTATE) | *Cell's State* |
| CSTATE | $\equiv$ | Sh ‖ Ex | *Cell's Cstate* |
| HSTATE | $\equiv$ | R(DIR) ‖ W(id) | *Cell's Hstate* |
| DIR | $\equiv$ | $\epsilon$ ‖ id \| DIR | *Directory* |
| INQ | $\equiv$ | $\epsilon$ ‖ MSG $\odot$ INQ | *Incoming Queue* |
| OUTQ | $\equiv$ | $\epsilon$ ‖ MSG $\otimes$ OUTQ | *Outgoing Queue* |
| MSG | $\equiv$ | $\mathsf{Msg}(\mathsf{id}_{\mathrm{src}}, \mathsf{id}_{\mathrm{dest}}, \mathrm{CMD}, \mathsf{a}, \mathsf{v})$ | *Protocol Message* |
| CMD | $\equiv$ | REPLY ‖ REQUEST | *Message Command* |
| REPLY | $\equiv$ | Sh-rep ‖ Ex-rep ‖ Wb-rep ‖ Inv-rep | |
| | ‖ | Upgrade-rep ‖ Pushout-rep | *Reply Command* |
| REQUEST | $\equiv$ | Sh-req ‖ Ex-req ‖ Wb-req ‖ Inv-req | |
| | ‖ | Pushout-req | *Request Command* |
| TRECS | $\equiv$ | $\epsilon$ ‖ Trec(a, INITIATOR) \| TRECS | *Transient Records* |
| INITIATOR | $\equiv$ | (ia,Load) ‖ (ia,Store) ‖ (id,REQUEST) | *Initiator* |

Figure 1: The HCN-opt Model (Initially, all non-outermost memories, all incoming and outgoing message queues, and all transient records are empty; the outermost memory contains a cell in the (Ex,R($\epsilon$)) state for each address)

### 1.1.2 Cache-Miss Rules

*Read-Cache-Miss Rule*

$\mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
*if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{r} := \mathsf{Load}(\mathsf{r}_1)$ *and* $\mathsf{rf}[\mathsf{r}_1] \notin \mathsf{m}$ *and* $\mathsf{rf}[\mathsf{r}_1] \notin \mathsf{trecs}$
$\longrightarrow$ $\mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_p}, \text{Sh-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Load})) \mid \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
*where* $\mathsf{id_p} = \mathsf{parent}(\mathsf{id})$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r}_1]$

*Write-Cache-Miss Rule*

$\mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
*if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{Store}(\mathsf{r}_1, \mathsf{r}_2)$ *and* $\mathsf{Cell}(\mathsf{rf}[\mathsf{r}_1], \text{-}, (\mathsf{Ex}, \text{-})) \notin \mathsf{m}$ *and* $\mathsf{rf}[\mathsf{r}_1] \notin \mathsf{trecs}$
$\longrightarrow$ $\mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_p}, \text{Ex-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Store})) \mid \mathsf{trecs}\rangle,\ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
*where* $\mathsf{id_p} = \mathsf{parent}(\mathsf{id})$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r}_1]$

## 1.2 Child-to-Parent Request Processing Rules

### 1.2.1 Sh-Request Rules

*Receive-Sh-Req-And-Send-Sh-Rep Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{cs},\mathsf{R}(\mathsf{dir}))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Sh-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$
*if* $\mathsf{id_k} \notin \mathsf{dir}$ *and* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{cs},\mathsf{R}(\mathsf{id_k}|\mathsf{dir}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}), \mathsf{trecs}\rangle$

*Receive-Sh-Req-And-Send-Wb-Req Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_j}))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Sh-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$
*if* $\mathsf{id_k} \neq \mathsf{id_j}$ *and* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_j}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_j}, \text{Wb-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Sh-req})) \mid \mathsf{trecs}\rangle$

*Receive-Sh-Req-And-Send-Sh-Req Rule*

$\langle \mathsf{id}, \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Sh-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$ *if* $\mathsf{a} \notin \mathsf{m}$ *and* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_p}, \text{Sh-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Sh-req})) \mid \mathsf{trecs}\rangle$
*where* $\mathsf{id_p} = \mathsf{parent}(\mathsf{id})$

### 1.2.2 Ex-Request Rules

*Receive-Ex-Req-And-Send-Ex-Rep Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Ex-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$ *if* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}), \mathsf{trecs}\rangle$

*Receive-Ex-Req-And-Send-Upgrade-Rep Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\mathsf{id_k}))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Ex-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$ *if* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Upgrade-rep}, \mathsf{a}, \mathsf{v}), \mathsf{trecs}\rangle$

*Receive-Ex-Req-And-Multicast-Inv-Req Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\mathsf{dir}))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Ex-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$
*if* $\mathsf{dir}\text{-}\{\mathsf{id_k}\} \neq \epsilon$ *and* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\mathsf{dir}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{multicast}(\mathsf{id}, \mathsf{dir}\text{-}\{\mathsf{id_k}\}, \text{Inv-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs}\rangle$

*Receive-Ex-Req-And-Send-Pushout-Req Rule*

$\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_j}))) \mid \mathsf{m}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Ex-req}, \mathsf{a}, \bot) \odot \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle$
*if* $\mathsf{id_k} \neq \mathsf{id_j}$ *and* $\mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow$ $\langle \mathsf{id}, \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_j}))) \mid \mathsf{m}, \mathsf{in}, \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_j}, \text{Pushout-req}, \mathsf{a}, \bot), \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs}\rangle$

*Receive-Ex-Req-And-Send-Ex-Req Rule*

$\langle$id, m, Msg($\mathrm{id_k}$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$
*if* Cell(a,-,(Ex,-)) $\notin$ m *and* a $\notin$ trecs
$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Ex-req, a, $\perp$), Trec(a, ($\mathrm{id_k}$,Ex-req)) | trecs$\rangle$
*where* $\mathrm{id_p}$ = parent(id)

## 1.3 Parent-to-Child Request Processing Rules

### 1.3.1 Wb-Request Rules

*Receive-Wb-Req-And-Send-Wb-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, Msg($\mathrm{id_p}$, id, Wb-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs
$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Wb-rep, a, v), trecs$\rangle$

*Receive-Wb-Req-And-Send-Wb-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_p}$, id, Wb-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs
$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_k}$, Wb-req, a, $\perp$), Trec(a, ($\mathrm{id_p}$,Wb-req)) | trecs$\rangle$

### 1.3.2 Pushout-Request Rules

*Receive-Pushout-Req-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, Msg($\mathrm{id_p}$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs
$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Pushout-rep, a, v), trecs$\rangle$

*Receive-Pushout-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, Msg($\mathrm{id_p}$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$
*if* dir $\neq \epsilon$ *and* a $\notin$ trecs
$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out $\otimes$ multicast(id, dir, Inv-req, a, $\perp$), Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

*Receive-Pushout-Req-And-Send-Pushout-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_p}$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs
$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_k}$, Pushout-req, a, $\perp$), Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

### 1.3.3 Inv-Request Rules

*Receive-Inv-Req-And-Send-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, Msg($\mathrm{id_p}$, id, Inv-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$
$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Inv-rep, a, $\perp$), trecs$\rangle$

*Receive-Inv-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Sh,R(dir))) | m, Msg($\mathrm{id_p}$, id, Inv-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* dir $\neq \epsilon$
$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ multicast(id, dir, Inv-req, a, $\perp$), Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

## 1.4 Parent-to-Child Reply Processing Rules

### 1.4.1 Sh-Reply Rules

*Receive-Sh-Rep-And-Execute-Load Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \text{Load})) \mid \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
$\quad\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{r} := \mathsf{Load}(\mathsf{r_1})$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r_1}]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}+1,\ \mathsf{rf}[\mathsf{r} := \mathsf{v}],\ \mathsf{prog}))$

*Receive-Sh-Rep-And-Send-Sh-Rep Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Sh-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Sh},\mathsf{R}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

### 1.4.2 Ex-Reply Rules

*Receive-Ex-Rep-And-Execute-Store Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{u}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \text{Store})) \mid \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))$
$\quad\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{Store}(\mathsf{r_1}, \mathsf{r_2})$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r_1}]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}+1,\ \mathsf{rf},\ \mathsf{prog}))\quad$ *where* $\mathsf{v} = \mathsf{rf}[\mathsf{r_2}]$

*Receive-Ex-Rep-And-Send-Ex-Rep Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

### 1.4.3 Upgrade-Reply Rules

*Receive-Upgrade-Rep-And-Execute-Store Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{u},(\text{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Upgrade-rep}, \mathsf{a}, \perp) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \text{Store})) \mid \mathsf{trecs} \rangle,$
$\quad\quad \mathsf{Proc}(\mathsf{ia}, \mathsf{rf}, \mathsf{prog}))\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{Store}(\mathsf{r_1}, \mathsf{r_2})$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r_1}]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}+1,\ \mathsf{rf},\ \mathsf{prog}))\quad$ *where* $\mathsf{v} = \mathsf{rf}[\mathsf{r_2}]$

*Receive-Upgrade-Rep-And-Send-Ex-Rep Rule*

$\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Upgrade-rep}, \mathsf{a}, \perp) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

*Receive-Upgrade-Rep-And-Send-Upgrade-Rep Rule*

$\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Sh},\mathsf{R}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Upgrade-rep}, \mathsf{a}, \perp) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Upgrade-rep}, \mathsf{a}, \perp),\ \mathsf{trecs} \rangle$

*Receive-Upgrade-Rep-And-Multicast-Inv-Req Rule*

$\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Sh},\mathsf{R}(\text{dir}))) \mid \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Upgrade-rep}, \mathsf{a}, \perp) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\quad\quad$ *if* $\text{dir-}\{\mathsf{id_k}\} \neq \epsilon$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{R}(\text{dir}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{multicast}(\mathsf{id}, \text{dir-}\{\mathsf{id_k}\}, \text{Inv-req}, \mathsf{a}, \perp),\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$

## 1.5 Child-to-Parent Reply Processing Rules

### 1.5.1 Wb-Reply Rules

*Receive-Wb-Rep-And-Send-Sh-Rep Rule*

$\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{u},(\text{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \text{Wb-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_j}, \text{Sh-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\text{Ex},\mathsf{R}(\mathsf{id_k} | \mathsf{id_j}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_j}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

*Receive-Wb-Rep-And-Send-Wb-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Wb-rep, a, v) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Wb-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R($\mathrm{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Wb-rep, a, v), trecs$\rangle$

### 1.5.2 Pushout-Reply Rules

*Receive-Pushout-Rep-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Ex-rep, a, v), trecs$\rangle$

*Receive-Pushout-Rep-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Pushout-rep, a, v), trecs$\rangle$

### 1.5.3 Inv-Reply Rules

*Receive-Inv-Rep-With-Ex-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

*if* dir - $\{\mathrm{id_j}\} \neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

*Receive-Inv-Rep-With-Pushout-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

*if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

*Receive-Inv-Rep-With-Inv-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Sh,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

*if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

*Receive-Inv-Rep-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Ex-rep, a, v), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Upgrade-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|$\mathrm{id_j}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Upgrade-rep, a, $\perp$), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Pushout-rep, a, v), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Inv-rep, a, $\perp$), trecs$\rangle$

## 1.6 Message Passing Rules

*Message-Passing-To-Child Rule*

$$\begin{aligned} & \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{in},\ \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}) \otimes \mathsf{out},\ \mathsf{trecs}\rangle,\ \ \mathsf{Sys}(\langle \mathsf{id_k},\ \mathsf{m_k},\ \mathsf{in_k},\ \mathsf{out_k},\ \mathsf{trecs_k}\rangle,\ \mathsf{eu_k}) \mid \mathsf{sg}) \\ \longrightarrow\ & \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs}\rangle,\ \ \mathsf{Sys}(\langle \mathsf{id_k},\ \mathsf{m_k},\ \mathsf{in_k} \odot \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}),\ \mathsf{out_k},\ \mathsf{trecs_k}\rangle,\ \mathsf{eu_k}) \mid \mathsf{sg}) \end{aligned}$$

*Message-Passing-To-Parent Rule*

$$\begin{aligned} & \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs}\rangle,\ \ \mathsf{Sys}(\langle \mathsf{id_k},\ \mathsf{m_k},\ \mathsf{in_k},\ \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}) \otimes \mathsf{out_k},\ \mathsf{trecs_k}\rangle,\ \mathsf{eu_k}) \mid \mathsf{sg}) \\ \longrightarrow\ & \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{in} \odot \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}),\ \mathsf{out},\ \mathsf{trecs}\rangle,\ \ \mathsf{Sys}(\langle \mathsf{id_k},\ \mathsf{m_k},\ \mathsf{in_k},\ \mathsf{out_k},\ \mathsf{trecs_k}\rangle,\ \mathsf{eu_k}) \mid \mathsf{sg}) \end{aligned}$$

# 2 HCN-adpt: An Adapative Protocol

The HCN-adpt protocol is an adaptive protocol based on HCN-opt. While all HCN-opt rules remain unchanged, new rules are added to incorporate adaptivity, which allows protocol messages to be issued voluntarily, *i.e.*, without initiator. It is worth noting that voluntary Pushout-rep and Inv-rep effectively model cache line replacements due to capacity or associativity conflict, while voluntary Sh-req and Ex-req behave like data prefetch.

The newly added rules fall into four categories: voluntary reply issue rules, voluntary request issue rules, unexpected reply processing rules, and unexpected request processing rules.

## 2.1 Memory Access Rules

### 2.1.1 Cache-Hit Rules

*Read-Cache-Hit Rule*

Sys(⟨id, Cell(a,v,(cs,R($\epsilon$))) | m, in, out, trecs⟩, Proc(ia, rf, prog))
*if* prog[ia] = r:=Load($r_1$) *and* a = rf[$r_1$]
$\longrightarrow$ Sys(⟨id, Cell(a,v,(cs,R($\epsilon$))) | m, in, out, trecs⟩, Proc(ia+1, rf[r:=v], prog))

*Write-Cache-Hit Rule*

Sys(⟨id, Cell(a,u,(Ex,R($\epsilon$))) | m, in, out, trecs⟩, Proc(ia, rf, prog))
*if* prog[ia] = Store($r_1$, $r_2$) *and* a = rf[$r_1$]
$\longrightarrow$ Sys(⟨id, Cell(a,v,(Ex,R($\epsilon$))) | m, in, out, trecs⟩, Proc(ia+1, rf, prog)) *where* v = rf[$r_2$]

### 2.1.2 Cache-Miss Rules

*Read-Cache-Miss Rule*

Sys(⟨id, m, in, out, trecs⟩, Proc(ia, rf, prog))
*if* prog[ia] = r:=Load($r_1$) *and* rf[$r_1$] $\notin$ m *and* rf[$r_1$] $\notin$ trecs
$\longrightarrow$ Sys(⟨id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Sh-req, a, $\perp$), Trec(a, (ia,Load)) | trecs⟩, Proc(ia, rf, prog))
*where* $\mathrm{id_p}$ = parent(id) *and* a = rf[$r_1$]

*Write-Cache-Miss Rule*

Sys(⟨id, m, in, out, trecs⟩, Proc(ia, rf, prog))
*if* prog[ia] = Store($r_1$, $r_2$) *and* Cell(rf[$r_1$],-,(Ex,-)) $\notin$ m *and* rf[$r_1$] $\notin$ trecs
$\longrightarrow$ Sys(⟨id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Ex-req, a, $\perp$), Trec(a, (ia,Store)) | trecs⟩, Proc(ia, rf, prog))
*where* $\mathrm{id_p}$ = parent(id) *and* a = rf[$r_1$]

## 2.2 Voluntary Reply Issue Rules

*Send-Voluntary-Sh-Rep Rule*

⟨id, Cell(a,v,(cs,R(dir))) | m, in, out, trecs⟩
*if* $\mathrm{id_k}$ $\in$ children(id) *and* $\mathrm{id_k}$ $\notin$ dir *and* a $\notin$ trecs
$\longrightarrow$ ⟨id, Cell(a,v,(cs,R($\mathrm{id_k}$|dir))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_k}$, Sh-rep, a, v), trecs⟩

*Send-Voluntary-Ex-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathsf{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_k}$, Ex-rep, a, v), trecs$\rangle$

*where* $\mathsf{id_k} \in$ children(id)

*Send-Voluntary-Wb-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_p}$, Wb-rep, a, v), trecs$\rangle$

*where* $\mathsf{id_p}$ = parent(id)

*Send-Voluntary-Pushout-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathsf{id_p}$, Pushout-rep, a, v), trecs$\rangle$ *where* $\mathsf{id_p}$ = parent(id)

*Send-Voluntary-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathsf{id_p}$, Inv-rep, a, $\perp$), trecs$\rangle$ *where* $\mathsf{id_p}$ = parent(id)

## 2.3 Voluntary Request Issue Rules

*Send-Voluntary-Sh-Req Rule*

$\langle$id, m, in, out, trecs$\rangle$ *if* a $\notin$ m *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathsf{id_p}$, Sh-req, a, $\perp$), trecs$\rangle$ *where* $\mathsf{id_p}$ = parent(id)

*Send-Voluntary-Ex-Req Rule*

$\langle$id, m, in, out, trecs$\rangle$ *if* Cell(a,-,(Ex,-)) $\notin$ m *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathsf{id_p}$, Ex-req, a, $\perp$), trecs$\rangle$ *where* $\mathsf{id_p}$ = parent(id)

*Send-Voluntary-Wb-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($\mathsf{id_k}$))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathsf{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_k}$, Wb-req, a, $\perp$), trecs$\rangle$

*Send-Voluntary-Pushout-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($\mathsf{id_k}$))) | m, in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathsf{id_k}$))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_k}$, Pushout-req, a, $\perp$), trecs$\rangle$

*Send-Voluntary-Inv-Req Rule*

$\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out, trecs$\rangle$ *if* dir $\neq \epsilon$ *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_k}$, Inv-req, a, $\perp$), trecs$\rangle$ *where* $\mathsf{id_k} \in$ dir

## 2.4 Child-to-Parent Request Processing Rules

### 2.4.1 Sh-Request Rules

*Receive-Sh-Req-And-Send-Sh-Rep Rule*

$\langle$id, Cell(a,v,(cs,R(dir))) | m, Msg($\mathsf{id_k}$, id, Sh-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

*if* $\mathsf{id_k} \notin$ dir *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(cs,R($\mathsf{id_k}$|dir))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_k}$, Sh-rep, a, v), trecs$\rangle$

*Receive-Sh-Req-And-Send-Wb-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($\mathsf{id_j}$))) | m, Msg($\mathsf{id_k}$, id, Sh-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

*if* $\mathsf{id_k} \neq \mathsf{id_j}$ *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathsf{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathsf{id_j}$, Wb-req, a, $\perp$), Trec(a, ($\mathsf{id_k}$,Sh-req)) | trecs$\rangle$

*Receive-Sh-Req-And-Send-Sh-Req Rule*

$\langle$id, m, Msg(id$_k$, id, Sh-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ m *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, id$_p$, Sh-req, a, $\perp$), Trec(a, (id$_k$,Sh-req)) | trecs$\rangle$

*where* id$_p$ = parent(id)

*Discard-Redundant-Sh-Req-Due-To-Shared-Copy Rule*

$\langle$id, Cell(a,v,(cs,R(id$_k$|dir))) | m, Msg(id$_k$, id, Sh-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(cs,R(id$_k$|dir))) | m, in, out, trecs$\rangle$

*Discard-Redundant-Sh-Req-Due-To-Exclusive-Copy Rule*

$\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, Msg(id$_k$, id, Sh-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, in, out, trecs$\rangle$

### 2.4.2 Ex-Request Rules

*Receive-Ex-Req-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, in, out $\otimes$ Msg(id, id$_k$, Ex-rep, a, v), trecs$\rangle$

*Receive-Ex-Req-And-Send-Upgrade-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R(id$_k$))) | m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, in, out $\otimes$ Msg(id, id$_k$, Upgrade-rep, a, v), trecs$\rangle$

*Receive-Ex-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

*if* dir-{id$_k$} $\neq \epsilon$ *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out $\otimes$ multicast(id, dir-{id$_k$}, Inv-req, a, $\perp$), Trec(a, (id$_k$,Ex-req)) | trecs$\rangle$

*Receive-Ex-Req-And-Send-Pushout-Req Rule*

$\langle$id, Cell(a,v,(Ex,W(id$_j$))) | m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

*if* id$_k \neq$ id$_j$ *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W(id$_j$))) | m, in, out $\otimes$ Msg(id, id$_j$, Pushout-req, a, $\perp$), Trec(a, (id$_k$,Ex-req)) | trecs$\rangle$

*Receive-Ex-Req-And-Send-Ex-Req Rule*

$\langle$id, m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

*if* Cell(a,-,(Ex,-)) $\notin$ m *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, id$_p$, Ex-req, a, $\perp$), Trec(a, (id$_k$,Ex-req)) | trecs$\rangle$

*where* id$_p$ = parent(id)

*Discard-Redundant-Ex-Req Rule*

$\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, Msg(id$_k$, id, Ex-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W(id$_k$))) | m, in, out, trecs$\rangle$

## 2.5 Parent-to-Child Request Processing Rules

### 2.5.1 Wb-Request Rules

*Receive-Wb-Req-And-Send-Wb-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, Msg(id$_p$, id, Wb-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ Msg(id, id$_p$, Wb-rep, a, v), trecs$\rangle$

*Receive-Wb-Req-And-Send-Wb-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, Msg($id_p$, id, Wb-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, in, out $\otimes$ Msg(id, $id_k$, Wb-req, a, $\perp$), Trec(a, ($id_p$,Wb-req)) | trecs$\rangle$

*Discard-Redundant-Wb-Req Rule*

$\langle$id, m, Msg($id_p$, id, Wb-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* Cell(a,-,(Ex,-)) $\notin$ m

$\longrightarrow$ $\langle$id, m, in, out, trecs$\rangle$

### 2.5.2 Pushout-Request Rules

*Receive-Pushout-Req-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $id_p$, Pushout-rep, a, v), trecs$\rangle$

*Receive-Pushout-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Ex,R(dir))) | m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$
*if* dir $\neq \epsilon$ *and* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out $\otimes$ multicast(id, dir, Inv-req, a, $\perp$), Trec(a, ($id_p$,Pushout-req)) | trecs$\rangle$

*Receive-Pushout-Req-And-Send-Pushout-Req Rule*

$\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, in, out $\otimes$ Msg(id, $id_k$, Pushout-req, a, $\perp$), Trec(a, ($id_p$,Pushout-req)) | trecs$\rangle$

*Receive-Unexpected-Pushout-Req-And-Send-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $id_p$, Inv-rep, a, $\perp$), trecs$\rangle$

*Receive-Unexpected-Pushout-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Sh,R(dir))) | m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ multicast(id, dir, Inv-req, a, $\perp$), Trec(a, ($id_p$,Inv-req)) | trecs$\rangle$

*Discard-Redundant-Pushout-Req Rule*

$\langle$id, m, Msg($id_p$, id, Pushout-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ m

$\longrightarrow$ $\langle$id, m, in, out, trecs$\rangle$

### 2.5.3 Inv-Request Rules

*Receive-Inv-Req-And-Send-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, Msg($id_p$, id, Inv-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $id_p$, Inv-rep, a, $\perp$), trecs$\rangle$

*Receive-Inv-Req-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Sh,R(dir))) | m, Msg($id_p$, id, Inv-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out $\otimes$ multicast(id, dir, Inv-req, a, $\perp$), Trec(a, ($id_p$,Inv-req)) | trecs$\rangle$

*Discard-Redundant-Inv-Req Rule*

$\langle$id, m, Msg($id_p$, id, Inv-req, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* Cell(a,-,(Sh,-)) $\notin$ m

$\longrightarrow$ $\langle$id, m, in, out, trecs$\rangle$

## 2.6 Parent-to-Child Reply Processing Rules

### 2.6.1 Sh-Reply Rules

*Receive-Sh-Rep-And-Execute-Load Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Load})) \mid \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia},\ \mathsf{rf},\ \mathsf{prog}))$
$\quad\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{r} := \mathsf{Load}(\mathsf{r}_1)$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r}_1]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}{+}1,\ \mathsf{rf}[\mathsf{r} := \mathsf{v}],\ \mathsf{prog}))$

*Receive-Sh-Rep-And-Send-Sh-Rep Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Sh-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Sh},\mathsf{R}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

*Receive-Unexpected-Sh-Rep-With-Suspended-Store Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Store})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Store})) \mid \mathsf{trecs} \rangle$

*Receive-Unexpected-Sh-Rep-With-Suspended-Ex-Req Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$

*Receive-Unexpected-Sh-Rep-Without-Initiator Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle \quad \textit{if}\ \ \mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Sh},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle$

### 2.6.2 Ex-Reply Rules

*Receive-Ex-Rep-And-Execute-Store Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{u}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Store})) \mid \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia},\ \mathsf{rf},\ \mathsf{prog}))$
$\quad\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{Store}(\mathsf{r}_1, \mathsf{r}_2)$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r}_1]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}{+}1,\ \mathsf{rf},\ \mathsf{prog}))$ *where* $\mathsf{v} = \mathsf{rf}[\mathsf{r}_2]$

*Receive-Ex-Rep-And-Send-Ex-Rep Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Ex-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{W}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

*Receive-Unexpected-Ex-Rep-With-Suspended-Load Rule*

$\mathsf{Sys}(\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{ia}, \mathsf{Load})) \mid \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia},\ \mathsf{rf},\ \mathsf{prog}))$
$\quad\quad$ *if* $\mathsf{prog}[\mathsf{ia}] = \mathsf{r} := \mathsf{Load}(\mathsf{r}_1)$ *and* $\mathsf{a} = \mathsf{rf}[\mathsf{r}_1]$
$\longrightarrow\ \ \mathsf{Sys}(\langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle,\ \ \mathsf{Proc}(\mathsf{ia}{+}1,\ \mathsf{rf}[\mathsf{r} := \mathsf{v}],\ \mathsf{prog}))$

*Receive-Unexpected-Ex-Rep-With-Suspended-Sh-Req Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{Trec}(\mathsf{a}, (\mathsf{id_k}, \text{Sh-req})) \mid \mathsf{trecs} \rangle$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\mathsf{id_k}))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out} \otimes \mathsf{Msg}(\mathsf{id}, \mathsf{id_k}, \text{Sh-rep}, \mathsf{a}, \mathsf{v}),\ \mathsf{trecs} \rangle$

*Receive-Unexpected-Ex-Rep-Without-Initiator Rule*

$\langle \mathsf{id},\ \mathsf{m},\ \mathsf{Msg}(\mathsf{id_p}, \mathsf{id}, \text{Ex-rep}, \mathsf{a}, \mathsf{v}) \odot \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle \quad \textit{if}\ \ \mathsf{a} \notin \mathsf{trecs}$
$\longrightarrow\ \ \langle \mathsf{id},\ \mathsf{Cell}(\mathsf{a},\mathsf{v},(\mathsf{Ex},\mathsf{R}(\epsilon))) \mid \mathsf{m},\ \mathsf{in},\ \mathsf{out},\ \mathsf{trecs} \rangle$

### 2.6.3 Upgrade-Reply Rules

*Receive-Upgrade-Rep-And-Execute-Store Rule*

$\langle$id, Cell(a,u,(Sh,R($\epsilon$))) | m, Msg($id_p$, id, Upgrade-rep, a, $\perp$) $\odot$ in, out, Trec(a, (ia,Store)) | trecs$\rangle$,
Proc(ia, rf, prog)  *if* prog[ia] = Store($r_1$, $r_2$) *and* a = rf[$r_1$]

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, in, out, trecs$\rangle$, Proc(ia+1, rf, prog)  *where* v = rf[$r_2$]

*Receive-Upgrade-Rep-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, Msg($id_p$, id, Upgrade-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($id_k$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, in, out $\otimes$ Msg(id, $id_k$, Ex-rep, a, v), trecs$\rangle$

*Receive-Upgrade-Rep-And-Send-Upgrade-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($id_k$))) | m, Msg($id_p$, id, Upgrade-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($id_k$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($id_k$))) | m, in, out $\otimes$ Msg(id, $id_k$, Upgrade-rep, a, $\perp$), trecs$\rangle$

*Receive-Upgrade-Rep-And-Multicast-Inv-Req Rule*

$\langle$id, Cell(a,v,(Sh,R(dir))) | m, Msg($id_p$, id, Upgrade-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($id_k$,Ex-req)) | trecs$\rangle$
*if* dir-{$id_k$} $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out $\otimes$ multicast(id, dir-{$id_k$}, Inv-req, a, $\perp$), Trec(a, ($id_k$,Ex-req)) | trecs$\rangle$

## 2.7 Child-to-Parent Reply Processing Rules

### 2.7.1 Wb-Reply Rules

*Receive-Wb-Rep-And-Send-Sh-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Wb-rep, a, v) $\odot$ in, out, Trec(a, ($id_j$,Sh-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($id_k$|$id_j$))) | m, in, out $\otimes$ Msg(id, $id_j$, Sh-rep, a, v), trecs$\rangle$

*Receive-Wb-Rep-And-Send-Wb-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Wb-rep, a, v) $\odot$ in, out, Trec(a, ($id_p$,Wb-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R($id_k$))) | m, in, out $\otimes$ Msg(id, $id_p$, Wb-rep, a, v), trecs$\rangle$

*Receive-Unexpected-Wb-Rep-With-Suspended-Ex-Req Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Wb-rep, a, v) $\odot$ in, out, Trec(a, ($id_j$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($id_k$))) | m, in, out, Trec(a, ($id_j$,Ex-req)) | trecs$\rangle$

*Receive-Unexpected-Wb-Rep-With-Suspended-Pushout-Req Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Wb-rep, a, v) $\odot$ in, out, Trec(a, ($id_j$,Pushout-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($id_k$))) | m, in, out, Trec(a, ($id_j$,Pushout-req)) | trecs$\rangle$

*Receive-Unexpected-Wb-Rep-Without-Initiator Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Wb-rep, a, v) $\odot$ in, out, trecs$\rangle$  *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($id_k$))) | m, in, out, trecs$\rangle$

### 2.7.2 Pushout-Reply Rules

*Receive-Pushout-Rep-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($id_j$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($id_j$))) | m, in, out $\otimes$ Msg(id, $id_j$, Ex-rep, a, v), trecs$\rangle$

*Receive-Pushout-Rep-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,u,(Ex,W($id_k$))) | m, Msg($id_k$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($id_p$,Pushout-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $id_p$, Pushout-rep, a, v), trecs$\rangle$

*Receive-Unexpected-Pushout-Rep-With-Suspended-Sh-Req Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Sh-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Sh-rep, a, v), trecs$\rangle$

*Receive-Unexpected-Pushout-Rep-With-Suspended-Wb-Req Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Pushout-rep, a, v) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Wb-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R($\epsilon$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Wb-rep, a, v), trecs$\rangle$

*Receive-Unexpected-Pushout-Rep-Without-Initiator Rule*

$\langle$id, Cell(a,u,(Ex,W($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Pushout-rep, a, v) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R($\epsilon$))) | m, in, out, trecs$\rangle$

### 2.7.3 Inv-Reply Rules

*Receive-Inv-Rep-With-Ex-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

*if* dir - {$\mathrm{id_j}$} $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

*Receive-Inv-Rep-With-Inv-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Sh,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

*if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Sh,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

*Receive-Inv-Rep-With-Pushout-Req-Suspended Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

*if* dir $\neq \epsilon$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,R(dir))) | m, in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

*Receive-Inv-Rep-And-Send-Ex-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Ex-rep, a, v), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Upgrade-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$|$\mathrm{id_j}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_j}$,Ex-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, Cell(a,v,(Ex,W($\mathrm{id_j}$))) | m, in, out $\otimes$ Msg(id, $\mathrm{id_j}$, Upgrade-rep, a, $\perp$), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Inv-Rep Rule*

$\langle$id, Cell(a,v,(Sh,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Inv-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Inv-rep, a, $\perp$), trecs$\rangle$

*Receive-Inv-Rep-And-Send-Pushout-Rep Rule*

$\langle$id, Cell(a,v,(Ex,R($\mathrm{id_k}$))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, Trec(a, ($\mathrm{id_p}$,Pushout-req)) | trecs$\rangle$

$\longrightarrow$ $\langle$id, m, in, out $\otimes$ Msg(id, $\mathrm{id_p}$, Pushout-rep, a, v), trecs$\rangle$

*Receive-Unexpected-Inv-Rep-Without-Initiator Rule*

$\langle$id, Cell(a,v,(cs,R($\mathrm{id_k}$|dir))) | m, Msg($\mathrm{id_k}$, id, Inv-rep, a, $\perp$) $\odot$ in, out, trecs$\rangle$ *if* a $\notin$ trecs

$\longrightarrow$ $\langle$id, Cell(a,v,(cs,R(dir))) | m, in, out, trecs$\rangle$

## 2.8 Message Passing Rules

*Message-Passing-To-Child Rule*

Sys($\langle$id, m, in, Msg(id, $\mathrm{id_k}$, cmd, a, v) $\otimes$ out, trecs$\rangle$, Sys($\langle$$\mathrm{id_k}$, $\mathrm{m_k}$, $\mathrm{in_k}$, $\mathrm{out_k}$, $\mathrm{trecs_k}\rangle$, $\mathrm{eu_k}$) | sg)

$\longrightarrow$ Sys($\langle$id, m, in, out, trecs$\rangle$, Sys($\langle$$\mathrm{id_k}$, $\mathrm{m_k}$, $\mathrm{in_k}$ $\odot$ Msg(id, $\mathrm{id_k}$, cmd, a, v), $\mathrm{out_k}$, $\mathrm{trecs_k}\rangle$, $\mathrm{eu_k}$) | sg)

*Message-Passing-To-Parent Rule*

$\mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in}, \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Sys}(\langle \mathsf{id_k}, \mathsf{m_k}, \mathsf{in_k}, \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}) \otimes \mathsf{out_k}, \mathsf{trecs_k}\rangle, \mathsf{eu_k}) \mid \mathsf{sg})$

$\longrightarrow\ \mathsf{Sys}(\langle \mathsf{id}, \mathsf{m}, \mathsf{in} \odot \mathsf{Msg}(\mathsf{id_k}, \mathsf{id}, \mathsf{cmd}, \mathsf{a}, \mathsf{v}), \mathsf{out}, \mathsf{trecs}\rangle,\ \mathsf{Sys}(\langle \mathsf{id_k}, \mathsf{m_k}, \mathsf{in_k}, \mathsf{out_k}, \mathsf{trecs_k}\rangle, \mathsf{eu_k}) \mid \mathsf{sg})$

## 2.9 Buffer Management

## 2.10 Discussion

**Voluntary Request Issue Rules:** In theory, the same voluntary request message can be issued repeatedly. In practice, when a voluntary request message is issued, we can encode this information in Trecs to prevent the same request messages being issued multiple times. Moreover, we can distinguish between voluntary request messages and normal request messages by encoding a bit information in the message format. An advantage of this is that voluntary request messages can be discarded without affecting the correctness of the protocol. We may want to do this due to control flow of message passing, or the voluntary request message is deemed inappropriate upon arrival. The protocol liveness liveness is guaranteed regardless of how voluntary messages are issued.

## 2.11 FIFO Message Passing

In HCN-adpt, there are two scenarios that reply messages can be reordered due to non-FIFO message passing: Inv-rep overtakes Wb-rep, and Upgrade-rep overtakes ShRep. When this happens, the reply message that arrives first (*i.e.*, Inv-rep and Upgrade-rep) cannot be processed before the other reply message (*i.e.*, Wb-rep and Sh-rep) is processed (unless some other rules are added, which will inevitably increase the number of states). In HCN-adpt, we deal with this issue by imposing circular buffer management for reply messages.

With FIFO message passing, this reordeing cannot happen. Thus, the circular buffer management is no longer needed.

$\mathsf{msg_1} \otimes \mathsf{msg_2} \equiv \mathsf{msg_2} \otimes \mathsf{msg_1}$
*if* $\mathsf{addr}(\mathsf{msg_1}) \neq \mathsf{addr}(\mathsf{msg_2})$ *or* $\mathsf{dest}(\mathsf{msg_1}) \neq \mathsf{dest}(\mathsf{msg_1})$

$\mathsf{msg_1} \odot \mathsf{msg_2} \equiv \mathsf{msg_2} \odot \mathsf{msg_1}$
*if* $\mathsf{addr}(\mathsf{msg_1}) \neq \mathsf{addr}(\mathsf{msg_2})$ *or* $\mathsf{src}(\mathsf{msg_1}) \neq \mathsf{src}(\mathsf{msg_1})$

**FIFO Message Passing:** We assume message passing is FIFO in the sense that messages that have the same source and destination are received in the same order as they are issued if they are regarding the same memory location.