A Formal Model for Secure Multiparty Computations

ANONYMOUS AUTHOR(S)

Contents

XXXX-XXXX/2022/3-ART $15.00

https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SMC^2 SEMANTIC RULES

1.1 Grammar

$$
\begin{array}{lcl}
ty & ::= & a\ bty \mid a\ bty* \mid bty \mid bty* \mid tyL \to ty \\
bty & ::= & \text{int} \mid \text{float} \mid \text{void} \\
a & ::= & \text{private} \mid \text{public} \\
tyL & ::= & [\,] \mid ty :: tyL \\
 & & \\
s & ::= & var = e \mid *x = e \mid s_1;\ s_2 \mid decl \mid \text{while } (e)\ s \\
 & & \mid ty\ x(P)\ \{s\} \mid \text{if } (e)\ s_1 \text{ else } s_2 \mid \{s\} \mid e \\
e & ::= & e\ bop\ e \mid uop\ x \mid var \mid x(E) \mid prim \mid (ty)\ e \\
 & & \mid (e) \mid v \\
decl & ::= & ty\ var \mid ty\ x(P) \\
var & ::= & x \mid x[e] \\
v & ::= & n \mid (l, \mu) \mid V \mid ptr \mid \text{NULL} \mid \text{skip} \\
V & ::= & [\,] \mid v :: V \\
ptr & ::= & [\alpha, L, J, i] \\
 & & \\
prim & ::= & \text{malloc}(e) \mid \text{pmalloc}(e,\ ty) \mid \text{sizeof}(ty) \\
 & & \mid \text{free}(e) \mid \text{pfree}(e) \\
 & & \mid \text{smcinput}(var,\ e) \mid \text{smcoutput}(var,\ e) \\
bop & ::= & - \mid + \mid \cdot \mid \div \mid == \mid\ != \mid < \\
uop & ::= & \& \mid * \mid ++ \\
E & ::= & E,\ e \mid e \mid \text{void} \\
P & ::= & P,\ ty\ var \mid ty\ var \mid \text{void}
\end{array}
$$

Fig. 1. Combined Vanilla C/SMC^2 Grammar. The color red denotes terms specific to programs written in SMC^2.

$$
\begin{array}{rcl}
C & ::= & \epsilon \mid (\text{p}, \gamma, \sigma, \Delta, \text{acc}, s) \parallel C \\
 & & \\
\gamma & ::= & [\,] \mid \gamma[x \to (l,\ ty)] \\
\sigma & ::= & [\,] \mid \sigma[l \to (\omega,\ ty,\ \alpha,\ \text{PermL})] \\
\text{PermL} & ::= & [\,] \mid [(0,\ a_0,\ p_0), \ldots, (\kappa, a_\kappa, p_\kappa)] \\
p & ::= & \text{Freeable} \mid \text{None} \\
\kappa & ::= & \tau(ty) \cdot \alpha - 1 \\
\Delta & ::= & [\,] \mid \delta :: \Delta \\
\delta & ::= & [\,] \mid ((l, \mu) \to (v_1, v_2, j, ty)) :: \delta \\
 & & \\
\mathcal{L} & ::= & \epsilon \mid (\text{p}, L) \parallel \mathcal{L} \\
L & ::= & [\,] \mid (l, \mu) :: L \\
\mathcal{D} & ::= & \epsilon \mid (\text{p}, D) \parallel \mathcal{D} \\
D & ::= & [\,] \mid d :: D \\
 & & \\
n, m, i, l, \mu, \alpha & \in & \mathbb{N} \\
\text{p, q} & \in & \mathbb{N} \\
j & ::= & 0 \mid 1 \\
\omega & ::= & \{0 \mid 1\}^+ \\
d & ::= & \{a \ldots z \mid 0 \ldots 9\}^+
\end{array}
$$

Fig. 2. Configuration: party identifier p, environment γ, memory σ, location map Δ, accumulator acc, and statement s.

1.2 Multiparty Computation Rules

The number of locations that a pointer will refer to and the level of indirection of a pointer is based on the program itself, and therefore must be the same across all parties. Proving that the level of indirection is consistent across all parties is done by induction over all rules, showing that it is assigned when a pointer is declared and never changed in any other rules. Proving that the number of locations a pointer will refer to can be done by evaluating the following:

- Private If Else rules change the number of locations based on the statements from both branches

- Private Free changes the number of locations based on how many locations the pointer that is being freed had
- Private Pointer Write and Dereference Write assign a new number of locations to a pointer based on the pointer that is being read from.
- All other rules do not modify the number of locations that a pointer refers to.

Given that CheckFreeable returns 1, by the definition of CheckFreeable we can assume that all offsets must be 0.

$$
\begin{array}{l}
\text{Private Free Multiple Locations}\\
\quad \{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}},\ \text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad acc = 0 \qquad (bty = \text{int}) \vee (bty = \text{float})\\
\quad \{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private } bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\alpha > 1\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\quad \{[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ i] = \text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\quad \text{if}(i > 1)\{ty = \text{private } bty*\} \text{ else } \{ty = \text{private } bty\}\\
\quad \{\text{CheckFreeable}(\gamma^{\mathrm{p}}, L^{\mathrm{p}}, J^{\mathrm{p}}, \sigma^{\mathrm{p}}) = 1\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\quad \{\forall (l_m^{\mathrm{p}}, 0) \in L^{\mathrm{p}}.\quad \sigma^{\mathrm{p}}(l_m^{\mathrm{p}}) = (\omega_m^{\mathrm{p}}, ty, \alpha_m, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha_m))\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\quad \text{MPC}_{free}([[\omega_0^1, \ldots, \omega_{\alpha-1}^1], \ldots, [\omega_0^{\mathrm{q}}, \ldots, \omega_{\alpha-1}^{\mathrm{q}}]], [J^1, \ldots J^{\mathrm{q}}])\\
\qquad\qquad = ([[\omega_0'^1, \ldots, \omega_{\alpha-1}'^1], \ldots, [\omega_0'^{\mathrm{q}}, \ldots, \omega_{\alpha-1}'^{\mathrm{q}}]], [J'^1, \ldots, J'^{\mathrm{q}}])\\
\quad \{\text{UpdateBytesFree}(\sigma^{\mathrm{p}}, L^{\mathrm{p}}, [\omega_0'^{\mathrm{p}}, \ldots, \omega_{\alpha-1}'^{\mathrm{p}}]) = \sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\quad \{(\sigma_2^{\mathrm{p}}, L_1^{\mathrm{p}}) = \text{UpdatePointerLocations}(\sigma_1^{\mathrm{p}}, L^{\mathrm{p}}[1 : \alpha-1],\ J^{\mathrm{p}}[1 : \alpha-1], L^{\mathrm{p}}[0], J^{\mathrm{p}}[0])\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\hline
((1, \gamma^1, \sigma^1, \Delta^1, acc, \text{pfree}(x)) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, acc, \text{pfree}(x))) \Downarrow_{(\text{ALL}, [mpfre])}^{(1, [(l^1, 0)] :: L^1 :: L_1^1) \parallel \ldots \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)] :: L^{\mathrm{q}} :: L_1^{\mathrm{q}})}\\
((1, \gamma^1, \sigma_2^1, \Delta^1, acc, \text{skip}) \qquad \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta^{\mathrm{q}}, acc, \text{skip}))
\end{array}
$$

Fig. 3. Rule for Secure Multiparty Computation: private free with multiple locations.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\text{Multiparty Pre-Increment Private Float Variable}}\\
\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \text{private float})\}_{\mathrm{p}=1}^{\mathrm{q}} & \{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private float}, 1, \text{PermL}(\text{Freeable}, \text{private float}, \text{private}, 1))\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\{(x) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} & \{\text{DecodeVal}(\text{private float}, \omega^{\mathrm{p}}) = n_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\text{MPC}_u(++, n_1^1, \ldots, n_1^{\mathrm{q}}) = (n_2^1, \ldots, n_2^{\mathrm{q}}) & \{\text{UpdateVal}(\sigma^{\mathrm{p}}, l^{\mathrm{p}}, n_2^{\mathrm{p}}, \text{private float}) = \sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\
\hline
\multicolumn{2}{c}{((1, \gamma^1, \sigma^1, \Delta^1, acc, ++\, x) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, acc, ++\, x)) \Downarrow_{(\text{ALL}, [mppin])}^{(1, [(l^1, 0)]) \parallel \ldots \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)])}}\\
\multicolumn{2}{c}{((1, \gamma^1, \sigma_1^1, \Delta^1, acc, n_2^1) \qquad \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta^{\mathrm{q}}, acc, n_2^{\mathrm{q}}))}
\end{array}
$$

Fig. 4. Rules for Secure Multiparty Computation of the Pre-Increment Operation over Private Float Values

Multiparty Array Read Private Index

$$\frac{\begin{array}{l}\{(e)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{(n^{\mathrm{p}})\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\ \text{const } a\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e))\Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},i^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},i^{\mathrm{q}}))\\ \{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ a\text{ const } bty*,1,\mathrm{PermL\_Ptr}(\mathrm{Freeable},a\text{ const } bty*,a,1))\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\mathrm{DecodePtr}(a\text{ const } bty*,\ 1,\ \omega^{\mathrm{p}})=[1,\ [(l_1^{\mathrm{p}},0)],\ [1],\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\sigma_1^{\mathrm{p}}(l_1^{\mathrm{p}})=(\omega_1^{\mathrm{p}},a\ bty,\alpha,\mathrm{PermL}(\mathrm{Freeable},a\ bty,a,\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\forall j\in\{0\ldots\alpha-1\}\quad \mathrm{DecodeArr}(a\ bty,j,\omega_1^{\mathrm{p}})=n_j^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \mathrm{MPC}_{ar}((i^1,[n_0^1,\ldots,n_{\alpha-1}^1]),\ldots,(i^{\mathrm{q}},[n_0^{\mathrm{q}},\ldots,n_{\alpha-1}^{\mathrm{q}}]))=(n^1,\ldots,n^{\mathrm{q}})\\ \mathcal{L}_2=(1,[(l^1,0),(l_1^1,0),\ldots,(l_1^1,\alpha-1)])\parallel\ldots\parallel(\mathrm{q},[(l^{\mathrm{q}},0),(l_1^{\mathrm{q}},0),\ldots,(l_1^{\mathrm{q}},\alpha-1)])\end{array}}{\begin{array}{c}((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},x[e])\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},x[e]))\Downarrow_{\mathcal{D}_1::(\mathrm{ALL},[mpra])}^{\mathcal{L}_1::\mathcal{L}_2}\\ ((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},n^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},n^{\mathrm{q}}))\end{array}}$$

Multiparty Array Write Private Index

$$\frac{\begin{array}{l}\{(e_1)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\ \text{private const } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e_1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e_1))\Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},i^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},i^{\mathrm{q}}))\\ ((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}}\mathrm{acc},e_2))\Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2}((1,\gamma^1,\sigma_2^1,\Delta_2^1,\mathrm{acc},n^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},n^{\mathrm{q}}))\\ \{\sigma_2^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ \text{private const } bty*,1,\mathrm{PermL\_Ptr}(\mathrm{Freeable},\text{private const } bty*,\text{private},1))\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega^{\mathrm{p}})=[1,\ [(l_1^{\mathrm{p}},0)],\ [1],\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\sigma_2^{\mathrm{p}}(l_1^{\mathrm{p}})=(\omega_1^{\mathrm{p}},\text{private } bty,\alpha,\mathrm{PermL}(\mathrm{Freeable},\text{private } bty,\text{private},\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\forall j\in\{0\ldots\alpha-1\}\quad \mathrm{DecodeArr}(\text{private } bty,j,\omega_1^{\mathrm{p}})=n_j^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \mathrm{MPC}_{aw}((i^1,n^1,[n_0^1,\ldots,n_{\alpha-1}^1]),\ldots,(i^{\mathrm{q}},n^{\mathrm{q}},[n_0^{\mathrm{q}},\ldots,n_{\alpha-1}^{\mathrm{q}}]))=([n_0'^1,\ldots,n_{\alpha-1}'^1],\ldots,[n_0'^{\mathrm{q}},\ldots,n_{\alpha-1}'^{\mathrm{q}}])\\ \{\forall j\in\{0\ldots\alpha-1\}\quad \mathrm{UpdateArr}(\sigma_{2+j}^{\mathrm{p}},\ (l_1^{\mathrm{p}},j),\ n_j'^{\mathrm{p}},\ \text{private } bty)=\sigma_{3+j}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \mathcal{L}_3=(1,[(l^{\mathrm{p}},0),(l_1^{\mathrm{p}},0),\ldots,(l_1^{\mathrm{p}},\alpha-1)])\parallel\ldots\parallel(\mathrm{q},[(l^{\mathrm{p}},0),(l_1^{\mathrm{p}},0),\ldots,(l_1^{\mathrm{p}},\alpha-1)])\end{array}}{\begin{array}{c}((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},x[e_1]=e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},x[e_1]=e_2))\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{ALL},[mpwa])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}\\ ((1,\gamma^1,\sigma_{3+\alpha-1}^1,\Delta_2^1,\mathrm{acc},\mathrm{skip})\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_{3+\alpha-1}^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},\mathrm{skip}))\end{array}}$$

Multiparty Binary Operation

$$\frac{\begin{array}{l}\{(e_1,e_2)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad bop\in\{\cdot,+,-,\div\}\\ \quad((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e_1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e_1))\\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},n_1^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},n_1^{\mathrm{q}}))\\ \quad((1,\gamma^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},e_2))\\ \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2}((1,\gamma^1,\sigma_2^1,\Delta_2^1,\mathrm{acc},n_2^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},n_2^{\mathrm{q}}))\\ \mathrm{MPC}_b(bop,[n_1^1,\ldots,n_1^{\mathrm{q}}],[n_2^1,\ldots,n_2^{\mathrm{q}}])=(n_3^1,\ldots,n_3^{\mathrm{q}})\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e_1\ bop\ e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e_1\ bop\ e_2))\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{ALL},[mpb])}^{\mathcal{L}_1::\mathcal{L}_2}\\ ((1,\gamma_2^1,\sigma_2^1,\Delta_2^1,\mathrm{acc},n_3^1)\qquad\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},n_3^{\mathrm{q}}))\end{array}}$$

Multiparty Comparison Operation

$$\frac{\begin{array}{l}\{(e_1,e_2)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad bop\in\{==,\,!=,\,<\}\\ \quad((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e_1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e_1))\\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma_1^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},n_1^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},n_1^{\mathrm{q}}))\\ \quad((1,\gamma_1^1,\sigma_1^1,\Delta_1^1,\mathrm{acc},e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\mathrm{acc},e_2))\\ \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2}((1,\gamma^1,\sigma_2^1,\Delta_2^1,\mathrm{acc},n_2^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},n_2^{\mathrm{q}}))\\ \mathrm{MPC}_{cmp}(bop,[n_1^1,\ldots,n_1^{\mathrm{q}}],[n_2^1,\ldots,n_2^{\mathrm{q}}])=(n_3^1,\ldots,n_3^{\mathrm{q}})\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\mathrm{acc},e_1\ bop\ e_2)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\mathrm{acc},e_1\ bop\ e_2))\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{ALL},[mpcmp])}^{\mathcal{L}_1::\mathcal{L}_2}\\ ((1,\gamma^1,\sigma_2^1,\Delta_2^1,\mathrm{acc},n_3^1)\qquad\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\mathrm{acc},n_3^{\mathrm{q}}))\end{array}}$$

Fig. 5. Rules for Secure Multiparty Computation when reading from or writing to a private index of an array and binary operations involving private data.

Multiparty Private Pointer Dereference Single Level Indirection

$$\frac{\begin{array}{l}\{(x)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\sigma^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable, private } bty*,\text{private},\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha>1\\ \{\text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}})=[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\text{Retrieve\_vals}(\alpha,L^{\mathrm{p}},\text{private } bty,\sigma^{\mathrm{p}})=([n_0^{\mathrm{p}},\ldots n_{\alpha-1}^{\mathrm{p}}],1)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \text{MPC}_{dv}([[n_0^1,\ldots,n_{\alpha-1}^1],\ldots,[n_0^{\mathrm{q}},\ldots,n_{\alpha-1}^{\mathrm{q}}]],[J^1,\ldots,J^{\mathrm{q}}])=(n^1,\ldots,n^{\mathrm{q}})\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},*x)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},*x))\Downarrow_{(\text{ALL},[mprdp])}^{(1,(l^1,0)::L^1)\parallel\ldots\parallel(\mathrm{q},(l^{\mathrm{q}},0)::L^{\mathrm{q}})}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},n^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},n^{\mathrm{q}}))\end{array}}$$

Multiparty Private Pointer Dereference Higher Level Indirection

$$\frac{\begin{array}{l}\{(x)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\sigma^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable, private } bty*,\text{private},\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha>1\\ \{\text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}})=[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ i]\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad i>1\\ \{\text{Retrieve\_vals}(\alpha,\ L^{\mathrm{p}},\ \text{private } bty*,\ \sigma^{\mathrm{p}})=([[\alpha_0,\ L_0^{\mathrm{p}},\ J_0^{\mathrm{p}},\ i-1],\ldots,[\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{p}},\ J_{\alpha-1}^{\mathrm{p}},\ i-1]],1)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \text{MPC}_{dp}([[[\alpha_0,\ L_0^1,\ J_0^1],\ldots,[\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1]],\ldots,[[\alpha_0,\ L_0^{\mathrm{q}},\ J_0^{\mathrm{q}}],\ldots,[\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{q}},\ J_{\alpha-1}^{\mathrm{q}}]]],[J^1,\ldots,J^{\mathrm{q}}])\\ \qquad =([[\alpha_\alpha,\ L_\alpha^1,\ J_\alpha^1],\ldots,[\alpha_\alpha,\ L_\alpha^{\mathrm{q}},\ J_\alpha^{\mathrm{q}}]])\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},*x)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},*x))\Downarrow_{(\text{ALL},[mprdp1])}^{(1,(l^1,0)::L^1)\parallel\ldots\parallel(\mathrm{q},(l^{\mathrm{q}},0)::L^{\mathrm{q}})}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},[\alpha_\alpha,L_\alpha^1,J_\alpha^1,i-1])\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},[\alpha_\alpha,L_\alpha^{\mathrm{q}},J_\alpha^{\mathrm{q}},i-1]))\end{array}}$$

Fig. 6. Multiparty SMC[2] semantic rules for private pointer dereference read with multiple locations

Multiparty Private Pointer Dereference Write Private Value

$$\frac{\begin{array}{l}\{(e)\vdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},e)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},e))\Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma^1,\sigma_1^1,\Delta_1^1,\text{acc},n^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\text{acc},n^{\mathrm{q}}))\\ \{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable, private } bty*,\text{private},\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha>1\\ \{\text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}})=[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\text{DynamicUpdate}(\Delta_1^{\mathrm{p}},\sigma_1^{\mathrm{p}},L^{\mathrm{p}},\ \text{acc},\text{private } bty)=(\Delta_2^{\mathrm{p}},L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\text{Retrieve\_vals}(\alpha,L^{\mathrm{p}},\text{private } bty,\sigma_1^{\mathrm{p}})=([n_0^{\mathrm{p}},\ldots n_{\alpha-1}^{\mathrm{p}}],1)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \text{MPC}_{wdv}([[n_0^1,\ldots,n_{\alpha-1}^1],\ldots,[n_0^{\mathrm{q}},\ldots,n_{\alpha-1}^{\mathrm{q}}]],[n^1,\ldots,n^{\mathrm{q}}],[J^1,\ldots,J^{\mathrm{q}}])=([n_0'^1,\ldots,n_{\alpha-1}'^1],\ldots,[n_0'^{\mathrm{q}},\ldots,n_{\alpha-1}'^{\mathrm{q}}])\\ \{\text{UpdateDerefVals}(\alpha,L^{\mathrm{p}},[n_0'^{\mathrm{p}},\ldots,n_{\alpha-1}'^{\mathrm{p}}],\text{private } bty,\sigma_1^{\mathrm{p}})=\sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},*x=e)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},*x=e))\Downarrow_{\mathcal{D}_1::(\text{ALL},[mpwdp3])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1)\parallel\ldots\parallel(\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})}\\ ((1,\gamma^1,\sigma_2^1,\Delta_2^1,\text{acc},\text{skip})\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\text{acc},\text{skip}))\end{array}}$$

Multiparty Private Pointer Dereference Write Public Value

$$\frac{\begin{array}{l}\{(e)\nvdash\gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \{\gamma^{\mathrm{p}}(x)=(l^{\mathrm{p}},\text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ ((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},e)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},e))\Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}((1,\gamma^1,\sigma_1^1,\Delta_1^1,\text{acc},n^1)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_1^{\mathrm{q}},\Delta_1^{\mathrm{q}},\text{acc},n^{\mathrm{q}}))\\ \{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}})=(\omega^{\mathrm{p}},\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable, private } bty*,\text{private},\alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha>1\\ \{\text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}})=[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\text{DynamicUpdate}(\Delta_1^{\mathrm{p}},\sigma_1^{\mathrm{p}},L^{\mathrm{p}},\ \text{acc},\text{private } bty)=(\Delta_2^{\mathrm{p}},L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \{\text{Retrieve\_vals}(\alpha,L^{\mathrm{p}},\text{private } bty,\sigma_1^{\mathrm{p}})=([n_0^{\mathrm{p}},\ldots n_{\alpha-1}^{\mathrm{p}}],1)\}_{\mathrm{p}=1}^{\mathrm{q}}\\ \text{MPC}_{wdv}([[n_0^1,\ldots,n_{\alpha-1}^1],\ldots,[n_0^{\mathrm{q}},\ldots,n_{\alpha-1}^{\mathrm{q}}]],[\text{encrypt}(n^1),\ldots,\text{encrypt}(n^{\mathrm{q}})],[J^1,\ldots,J^{\mathrm{q}}])\\ \qquad =([n_0'^1,\ldots,n_{\alpha-1}'^1],\ldots,[n_0'^{\mathrm{q}},\ldots,n_{\alpha-1}'^{\mathrm{q}}])\\ \{\text{UpdateDerefVals}(\alpha,L^{\mathrm{p}},[n_0'^{\mathrm{p}},\ldots,n_{\alpha-1}'^{\mathrm{p}}],\text{private } bty,\sigma_1^{\mathrm{p}})=\sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\end{array}}{\begin{array}{l}((1,\gamma^1,\sigma^1,\Delta^1,\text{acc},*x=e)\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma^{\mathrm{q}},\Delta^{\mathrm{q}},\text{acc},*x=e))\Downarrow_{\mathcal{D}_1::(\text{ALL},[mpwdp])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1)\parallel\ldots\parallel(\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})}\\ ((1,\gamma^1,\sigma_2^1,\Delta_2^1,\text{acc},\text{skip})\parallel\ldots\parallel(\mathrm{q},\gamma^{\mathrm{q}},\sigma_2^{\mathrm{q}},\Delta_2^{\mathrm{q}},\text{acc},\text{skip}))\end{array}}$$

Fig. 7. Multiparty SMC[2] semantic rules for dereference writing a public value to a private pointer.

Multiparty Private Pointer Dereference Write Value Higher Level Indirection

$$\frac{\begin{array}{l}\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \quad ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, e)) \\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, (l_e^1, \mu_e^1)) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \text{acc}, (l_e^{\mathrm{q}}, \mu_e^{\mathrm{q}}))) \\ \{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private } bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha > 1 \\ \{\text{DecodePtr}(\text{private } bty*, \alpha, \omega^{\mathrm{p}}) = [\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad i > 1 \\ \{\text{DynamicUpdate}(\Delta_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, L^{\mathrm{p}}, \text{acc}, \text{private } bty*) = (\Delta_2^{\mathrm{p}}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \{\text{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \text{private } bty*, \sigma_1^{\mathrm{p}}) = ([[\alpha_0, L_0^{\mathrm{p}}, J_0^{\mathrm{p}}, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{p}}, J_{\alpha-1}^{\mathrm{p}}, i-1]], 1)\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \text{MPC}_{wdp}([[[1, [(l_e^1, \mu_e^1)], [1], i-1], [\alpha_0, L_0^1, J_0^1, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^1, J_{\alpha-1}^1, i-1]], \ldots, \\ \qquad [[1, [(l_e^{\mathrm{q}}, \mu_e^{\mathrm{q}})], [1], i-1], [\alpha_0, L_0^{\mathrm{q}}, J_0^{\mathrm{q}}, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{q}}, J_{\alpha-1}^{\mathrm{q}}, i-1]]], [J^1, \ldots, J^{\mathrm{q}}]) \\ \quad = [[[\alpha_0', L_0'^1, J_0'^1, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^1, J_{\alpha-1}'^1, i-1]], \ldots, \\ \qquad [[\alpha_0', L_0'^{\mathrm{q}}, J_0'^{\mathrm{q}}, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^{\mathrm{q}}, J_{\alpha-1}'^{\mathrm{q}}, i-1]]] \\ \{\text{UpdateDerefVals}(\alpha, L^{\mathrm{p}}, [[\alpha_0', L_0'^{\mathrm{p}}, J_0'^{\mathrm{p}}, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^{\mathrm{p}}, J_{\alpha-1}'^{\mathrm{p}}, i-1]], \text{private } bty*, \sigma_1^{\mathrm{p}}) = \sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\end{array}}{\begin{array}{l}((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpwdp2])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} \\ ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \text{acc}, \text{skip}))\end{array}}$$

Multiparty Private Pointer Dereference Write Multiple Locations Higher Level Indirection

$$\frac{\begin{array}{l}\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \quad ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel .. \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, e)) \\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, [\alpha_e, L_e^1, J_e^1, i-1]) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \text{acc}, [\alpha_e, L_e^{\mathrm{q}}, J_e^{\mathrm{q}}, i-1])) \qquad \alpha_e > 1 \\ \{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private } bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad \alpha > 1 \\ \{\text{DecodePtr}(\text{private } bty*, \alpha, \omega^{\mathrm{p}}) = [\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}} \qquad i > 1 \\ \{\text{DynamicUpdate}(\Delta_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, L^{\mathrm{p}}, \text{acc}, \text{private } bty*) = (\Delta_2^{\mathrm{p}}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \{\text{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \text{private } bty*, \sigma_1^{\mathrm{p}}) = ([[\alpha_0, L_0^{\mathrm{p}}, J_0^{\mathrm{p}}, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{p}}, J_{\alpha-1}^{\mathrm{p}}, i-1]], 1)\}_{\mathrm{p}=1}^{\mathrm{q}} \\ \text{MPC}_{wdp}([[[\alpha_e, L_e^1, J_e^1, i-1], [\alpha_0, L_0^1, J_0^1, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^1, J_{\alpha-1}^1, i-1]], \ldots, \\ \qquad [[\alpha_e, L_e^{\mathrm{q}}, J_e^{\mathrm{q}}, i-1], [\alpha_0, L_0^{\mathrm{q}}, J_0^{\mathrm{q}}, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{q}}, J_{\alpha-1}^{\mathrm{q}}, i-1]]], [J^1, \ldots, J^{\mathrm{q}}]) \\ \quad = [[[\alpha_0', L_0'^1, J_0'^1, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^1, J_{\alpha-1}'^1, i-1]], \ldots, \\ \qquad [[\alpha_0', L_0'^{\mathrm{q}}, J_0'^{\mathrm{q}}, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^{\mathrm{q}}, J_{\alpha-1}'^{\mathrm{q}}, i-1]]] \\ \{\text{UpdateDerefVals}(\alpha, L^{\mathrm{p}}, [[\alpha_0', L_0'^{\mathrm{p}}, J_0'^{\mathrm{p}}, i-1], \ldots, [\alpha_{\alpha-1}', L_{\alpha-1}'^{\mathrm{p}}, J_{\alpha-1}'^{\mathrm{p}}, i-1]], \text{private } bty*, \sigma_1^{\mathrm{p}}) = \sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}\end{array}}{\begin{array}{l}((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpwdp1])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} \\ ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \text{acc}, \text{skip}))\end{array}}$$

Fig. 8. Multiparty SMC^2 semantic rules for dereference writing multiple location to a private pointer of a higher level of indirection

1.3 Branching and Loop Rules

Private If Else (Variable Tracking)

$$
\frac{
\begin{array}{l}
\quad ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \\
\Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \mathrm{acc}, n^1) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_1, \Delta^{\mathrm{q}}_1, \mathrm{acc}, n^{\mathrm{q}})) \qquad \{(e) \vdash \gamma^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\{\mathrm{Extract}(s_1, \ s_2, \gamma^{\mathrm{p}}) = (x_{list}, 0)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\{\mathrm{InitializeVariables}(x_{list}, \ \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}_1, n^{\mathrm{p}}, \ \mathrm{acc}+1) = (\gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_2, L^{\mathrm{p}}_2)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\quad ((1, \gamma^1_1, \sigma^1_2, \Delta^1_1, \mathrm{acc}+1, s_1) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_1, \sigma^{\mathrm{q}}_2, \Delta^{\mathrm{q}}_1, \mathrm{acc}+1, s_1)) \\
\Downarrow^{\mathcal{L}_3}_{\mathcal{D}_2} ((1, \gamma^1_2, \sigma^1_3, \Delta^1_2, \mathrm{acc}+1, \mathrm{skip}) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_2, \sigma^{\mathrm{q}}_3, \Delta^{\mathrm{q}}_2, \mathrm{acc}+1, \mathrm{skip})) \\
\{\mathrm{RestoreVariables}(x_{list}, \ \gamma^{\mathrm{p}}_1, \ \sigma^{\mathrm{p}}_3, \ \mathrm{acc}+1) = (\sigma^{\mathrm{p}}_4, L^{\mathrm{p}}_4)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\quad ((1, \gamma^1_1, \sigma^1_4, \Delta^1_2, \mathrm{acc}+1, s_2) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_1, \sigma^{\mathrm{q}}_4, \Delta^{\mathrm{q}}_2, \mathrm{acc}+1, s_2)) \\
\Downarrow^{\mathcal{L}_5}_{\mathcal{D}_3} ((1, \gamma^1_3, \sigma^1_5, \Delta^1_3, \mathrm{acc}+1, \mathrm{skip}) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_3, \sigma^{\mathrm{q}}_5, \Delta^{\mathrm{q}}_3, \mathrm{acc}+1, \mathrm{skip})) \\
\{\mathrm{ResolveVariables\_Retrieve}(x_{list}, \ \mathrm{acc}+1, \gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_5) = ([(v^{\mathrm{p}}_{t1}, v^{\mathrm{p}}_{e1}), \dots, (v^{\mathrm{p}}_{tm}, v^{\mathrm{p}}_{em})], n^{\mathrm{p}}, L^{\mathrm{p}}_6)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\mathrm{MPC}_{resolve}([n^1, \dots, n^{\mathrm{q}}], [[(v^1_{t1}, v^1_{e1}), \dots, (v^1_{tm}, v^1_{em})], \dots, [(v^{\mathrm{q}}_{t1}, v^{\mathrm{q}}_{e1}), \dots, (v^{\mathrm{q}}_{tm}, v^{\mathrm{q}}_{em})]]) \\
\quad = [[v^1_1, \dots, v^1_m], \dots, [v^{\mathrm{q}}_1, \dots, v^{\mathrm{q}}_m]] \\
\{\mathrm{ResolveVariables\_Store}(x_{list}, \ \mathrm{acc}+1, \gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_5, [v^{\mathrm{p}}_1, \dots, v^{\mathrm{p}}_m]) = (\sigma^{\mathrm{p}}_6, L^{\mathrm{p}}_7)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\mathcal{L}_2 = (1, L^1_2) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_2) \qquad \mathcal{L}_4 = (1, L^1_4) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_4) \\
\mathcal{L}_6 = (1, L^1_6) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_6) \qquad \mathcal{L}_7 = (1, L^1_7) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_7)
\end{array}
}{
\begin{array}{l}
((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p}, [iep])} \\
((1, \gamma^1, \sigma^1_6, \Delta^1_3, \mathrm{acc}, \mathrm{skip}) \qquad \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_6, \Delta^{\mathrm{q}}_3, \mathrm{acc}, \mathrm{skip}))
\end{array}
}
$$

Private If Else (Location Tracking)

$$
\frac{
\begin{array}{l}
\quad ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \\
\Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \mathrm{acc}, n^1) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_1, \Delta^{\mathrm{q}}_1, \mathrm{acc}, n^{\mathrm{q}})) \qquad \{(e) \vdash \gamma^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\{\mathrm{Extract}(s_1, \ s_2, \gamma^{\mathrm{p}}) = (x_{list}, 1)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\{\mathrm{Initialize}(\Delta^{\mathrm{p}}_1, x_{list}, \ \gamma^{\mathrm{p}}, \ \sigma^{\mathrm{p}}_1, \ n^{\mathrm{p}}, \mathrm{acc}+1) = (\gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_2, \Delta^{\mathrm{p}}_2, L^{\mathrm{p}}_2)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\quad ((1, \gamma^1_1, \sigma^1_2, \Delta^1_2, \mathrm{acc}+1, s_1) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_1, \sigma^{\mathrm{q}}_2, \Delta^{\mathrm{q}}_2, \mathrm{acc}+1, s_1)) \\
\Downarrow^{\mathcal{L}_3}_{\mathcal{D}_2} ((1, \gamma^1_2, \sigma^1_3, \Delta^1_3, \mathrm{acc}+1, \mathrm{skip}) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_2, \sigma^{\mathrm{q}}_3, \Delta^{\mathrm{q}}_3, \mathrm{acc}+1, \mathrm{skip})) \\
\{\mathrm{Restore}(\sigma^{\mathrm{p}}_3, \Delta^{\mathrm{p}}_3, \mathrm{acc}+1) = (\sigma^{\mathrm{p}}_4, \Delta^{\mathrm{p}}_4, L^{\mathrm{p}}_4)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\quad ((1, \gamma^1_1, \sigma^1_4, \Delta^1_4, \mathrm{acc}+1, s_2) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_1, \sigma^{\mathrm{q}}_4, \Delta^{\mathrm{q}}_4, \mathrm{acc}+1, s_2)) \\
\Downarrow^{\mathcal{L}_5}_{\mathcal{D}_3} ((1, \gamma^1_3, \sigma^1_5, \Delta^1_5, \mathrm{acc}+1, \mathrm{skip}) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}_3, \sigma^{\mathrm{q}}_5, \Delta^{\mathrm{q}}_5, \mathrm{acc}+1, \mathrm{skip})) \\
\{\mathrm{Resolve\_Retrieve}(\gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_5, \Delta^{\mathrm{p}}_5, \mathrm{acc}+1) = ([(v^{\mathrm{p}}_{t1}, v^{\mathrm{p}}_{e1}), \dots, (v^{\mathrm{p}}_{tm}, v^{\mathrm{p}}_{em})], n^{\mathrm{p}}, L^{\mathrm{p}}_6)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\mathrm{MPC}_{resolve}([n^1, \dots, n^{\mathrm{q}}], [[(v^1_{t1}, v^1_{e1}), \dots, (v^1_{tm}, v^1_{em})], \dots, [(v^{\mathrm{q}}_{t1}, v^{\mathrm{q}}_{e1}), \dots, (v^{\mathrm{q}}_{tm}, v^{\mathrm{q}}_{em})]]) \\
\quad = [[v^1_1, \dots, v^1_m], \dots [v^{\mathrm{q}}_1, \dots, v^{\mathrm{q}}_m]] \\
\{\mathrm{Resolve\_Store}(\Delta^{\mathrm{p}}_5, \sigma^{\mathrm{p}}_5, \mathrm{acc}+1, [v^{\mathrm{p}}_1, \dots, v^{\mathrm{p}}_m]) = (\sigma^{\mathrm{p}}_6, \Delta^{\mathrm{p}}_6, L^{\mathrm{p}}_7)\}^{\mathrm{q}}_{\mathrm{p}=1} \\
\mathcal{L}_2 = (1, L^1_2) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_2) \qquad \mathcal{L}_4 = (1, L^1_4) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_4) \\
\mathcal{L}_6 = (1, L^1_6) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_6) \qquad \mathcal{L}_7 = (1, L^1_7) \ \| \dots \| \ (\mathrm{q}, L^{\mathrm{q}}_7)
\end{array}
}{
\begin{array}{l}
((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \ \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{ALL}, [iepd])} \\
((1, \gamma^1, \sigma^1_6, \Delta^1_6, \mathrm{acc}, \mathrm{skip}) \qquad \| \dots \| \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_6, \Delta^{\mathrm{q}}_6, \mathrm{acc}, \mathrm{skip}))
\end{array}
}
$$

Fig. 9. The SMC^2 semantic rule for Private-Conditioned If Else - Multiparty Execution.

Public If Else True

$$\frac{(e) \nvdash \gamma \quad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \quad n \neq 0 \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, s_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)}$$

Public If Else False

$$\frac{(e) \nvdash \gamma \quad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \quad n = 0 \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, s_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [ief])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)}$$

While End

$$\frac{(e) \nvdash \gamma \quad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \quad n = 0}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}::(\mathrm{p}, [wle])}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)}$$

While Continue

$$\frac{(e) \nvdash \gamma \quad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \quad n \neq 0 \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [wlc])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C_2)}$$

Fig. 10. Additional SMC^2 semantic rules for branching and loops

1.4 Pointer Rules

Reading from a private pointer that has multiple locations and assigning multiple locations to a pointer are local operations. This is because we are simply reading from or writing to memory - we do not need to know the true location for the pointer in these operations.

Public Pointer Declaration

$$\frac{\begin{array}{lll} (ty = \text{public } bty*) & \text{acc} = 0 & l = \phi() \\ \text{GetIndirection}(*) = i & \multicolumn{2}{l}{\omega = \text{EncodePtr}(\text{public } bty*, [1, [(l_{default}, 0)], [1], i])} \\ \gamma_1 = \gamma[x \rightarrow (l, \ \text{public } bty*)] & \multicolumn{2}{l}{\sigma_1 = \sigma[l \rightarrow (\omega, \ \text{public } bty*, \ 1, \ \text{PermL\_Ptr}(\text{Freeable}, \text{public } bty*, \text{public}, 1))]} \end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ ty \ x) \parallel C) \Downarrow_{(\text{p}, [dp])}^{(\text{p}, [(l, 0)])} ((\text{p}, \gamma_1, \ \sigma_1, \ \Delta, \ \text{acc}, \ \text{skip}) \parallel C)}$$

Private Pointer Declaration

$$\frac{\begin{array}{ll} l = \phi() & ((ty = bty*) \vee (ty = \text{private } bty*)) \wedge ((bty = \text{int}) \vee (bty = \text{float})) \\ \text{GetIndirection}(*) = i & \omega = \text{EncodePtr}(\text{private } bty*, [1, [(l_{default}, 0)], [1], i]) \\ \gamma_1 = \gamma[x \rightarrow (l, \text{private } bty*)] & \sigma_1 = \sigma[l \rightarrow (\omega, \text{private } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, 1))] \end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ ty \ x) \parallel C) \Downarrow_{(\text{p}, [dp1])}^{(\text{p}, [(l, 0)])} ((\text{p}, \gamma_1, \ \sigma_1, \ \Delta, \ \text{acc}, \ \text{skip}) \parallel C)}$$

Public Pointer Write

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \text{acc}, \ (l_e, \mu_e)) \parallel C_1) \\ \gamma(x) = (l, \ \text{public } bty*) & \sigma_1(l) = (\omega, \ \text{public } bty*, \ 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public } bty*, \text{public}, 1)) \\ \text{acc} = 0 & \text{DecodePtr}(\text{public } bty*, \ 1, \ \omega) = [1, \ [(l_1, \mu_1)], \ [1], \ i] \\ & \text{UpdatePtr}(\sigma_1, \ (l, 0), \ [1, \ [(l_e, \mu_e)], \ [1], \ i], \ \text{public } bty*) = (\sigma_2, \ 1) \end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wp])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_1, \ \text{acc}, \ \text{skip}) \parallel C_1)}$$

Private Pointer Write

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \text{acc}, \ (l_e, \mu_e)) \parallel C_1) \\ \gamma(x) = (l, \ \text{private } bty*) & \sigma_1(l) = (\omega, \ \text{private } bty*, \ \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) \\ & \text{DecodePtr}(\text{private } bty*, \ \alpha, \ \omega) = [\alpha, \ L, \ J, \ i] \\ & \text{UpdatePtr}(\sigma_1, \ (l, 0), \ [1, \ [(l_e, \mu_e)], \ [1], \ i], \ \text{private } bty*) = (\sigma_2, \ 1) \end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wp1])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_1, \ \text{acc}, \ \text{skip}) \parallel C_1)}$$

Private Pointer Write Multiple Locations

$$\frac{\begin{array}{ll} (bty = \text{int}) \vee (bty = \text{float}) & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, [\alpha_e, L_e, J_e, i]) \parallel C_1) \\ \gamma(x) = (l, \text{private } bty*) & \sigma_1(l) = (\omega, \ \text{private } bty*, \ \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) \\ & \text{DecodePtr}(\text{private } bty*, \ \alpha, \ \omega) = [\alpha, \ L, \ J, \ i] \\ & \text{UpdatePtr}(\sigma_1, (l, 0), [\alpha_e, L_e, J_e, i], \text{private } bty*) = (\sigma_2, 1) \end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wp2])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_1, \ \text{acc}, \ \text{skip}) \parallel C_1)}$$

Fig. 11. Additional SMC^2 semantic rules for pointer declarations, reading, and writing.

All dereference operations over private pointers with single locations are executed locally, as we easily read and write at the publicly known location that the private pointer refers to. These operations have multiparty counterparts for when the private pointers refer to multiple locations, as when we execute those versions we must have communication between parties to privately evaluate what location's data we are truly reading from or writing to.

Pointer Read Single Location

$$\frac{\gamma(x) = (l,\ a\ bty*) \quad \sigma(l) = (\omega,\ a\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1)) \quad \text{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i]}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x) \parallel C) \Downarrow^{(\text{p},[(l,0)])}_{(\text{p},[rp])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_1, \mu_1)) \parallel C)}$$

Private Pointer Read Multiple Locations

$$\frac{\gamma(x) = (l,\ \text{private}\ bty*) \quad \sigma(l) = (\omega,\ \text{private}\ bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, \alpha)) \quad (bty = \text{int}) \vee (bty = \text{float}) \quad \text{DecodePtr}(\text{private}\ bty*,\ \alpha,\ \omega) = [\alpha,\ L,\ J,\ i]}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x) \parallel C) \Downarrow^{(\text{p},[(l,0)])}_{(\text{p},[rp1])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, [\alpha, L, J, i]) \parallel C)}$$

Pointer Dereference Single Location

$$\frac{\gamma(x) = (l,\ a\ bty*) \quad \sigma(l) = (\omega,\ a\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1)) \quad \text{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1] \quad \text{DerefPtr}(\sigma, a\ bty, (l_1, \mu_1)) = (n, 1)}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow^{(\text{p},[(l,0),(l_1,\mu_1)])}_{(\text{p},[rdp])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, n) \parallel C)}$$

Pointer Dereference Single Location Higher Level Indirection

$$\frac{\gamma(x) = (l,\ a\ bty*) \quad \sigma(l) = (\omega_1,\ a\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1)) \quad i > 1 \quad \text{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i] \quad \text{DerefPtrHLI}(\sigma, a\ bty*, (l_1, \mu_1)) = ([1,\ [(l_2, \mu_2)],\ [1],\ i-1], 1)}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow^{(\text{p},[(l,0),(l_1,\mu_1)])}_{(\text{p},[rdp1])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_2, \mu_2)) \parallel C)}$$

Private Pointer Dereference Single Location Higher Level Indirection

$$\frac{\gamma(x) = (l,\ \text{private}\ bty*) \quad \sigma(l) = (\omega_1,\ \text{private}\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, 1)) \quad i > 1 \quad \text{DecodePtr}(\text{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i] \quad \text{DerefPtrHLI}(\sigma, \text{private}\ bty*, (l_1, \mu_1)) = ([\alpha,\ L,\ J,\ i-1], 1)}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow^{(\text{p},[(l,0),(l_1,\mu_1)])}_{(\text{p},[rdp2])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, [\alpha, L, J, i-1]) \parallel C)}$$

Fig. 12. Additional SMC^2 semantic rules for pointer dereference read at a single location.

Public Pointer Dereference Write Public Value

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{public}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{public}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public}\ bty*, \mathrm{public}, 1)) \\ \mathrm{acc} = 0 & \mathrm{DecodePtr}(\mathrm{public}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ \mathrm{public}\ bty,\ 1] \\ & \mathrm{UpdateOffset}(\sigma_1,\ (l_1, \mu_1),\ n, \mathrm{public}\ bty) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Private Pointer Dereference Write Single Location Private Value

$$\frac{\begin{array}{ll} (e) \vdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{private}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{private}\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, 1)) \\ (bty = \mathrm{int}) \vee (bty = \mathrm{float}) & \mathrm{DecodePtr}(\mathrm{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1] \\ & \mathrm{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \mathrm{acc}, \mathrm{private}\ bty) = (\Delta_2, L_1) \\ & \mathrm{UpdateOffset}(\sigma_1,\ (l_1, \mu_1),\ n, \mathrm{private}\ bty) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp3])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Private Pointer Dereference Write Single Location Public Value

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{private}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{private}\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, 1)) \\ (bty = \mathrm{int}) \vee (bty = \mathrm{float}) & \mathrm{DecodePtr}(\mathrm{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1] \\ & \mathrm{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \mathrm{acc}, \mathrm{private}\ bty) = (\Delta_2, L_1) \\ & \mathrm{UpdateOffset}(\sigma_1,\ (l_1, \mu_1),\ \mathrm{encrypt}(n), \mathrm{private}\ bty) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp4])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Public Pointer Dereference Write Higher Level Indirection

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ (l_e, \mu_e)) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{public}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{public}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public}\ bty*, \mathrm{public}, 1)) \\ \mathrm{acc} = 0 & \mathrm{DecodePtr}(\mathrm{public}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i] \\ i > 1 & \mathrm{UpdatePtr}(\sigma_1,\ (l_1, \mu_1),\ [1,\ [(l_e, \mu_e)],\ [1],\ i-1], \mathrm{public}\ bty*) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp1])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::[(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Private Pointer Dereference Write to Single Location Higher Level Indirection

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ (l_e, \mu_e)) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{private}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{private}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, 1)) \\ & \mathrm{DecodePtr}(\mathrm{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i] \\ i > 1 & \mathrm{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \mathrm{acc}, \mathrm{private}\ bty*) = (\Delta_2, L_1) \\ & \mathrm{UpdatePtr}(\sigma_1,\ (l_1, \mu_1),\ [1,\ [(l_e, \mu_e)],\ [1],\ i-1], \mathrm{private}\ bty*) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp5])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Private Pointer Dereference Write Multiple Locations to Single Location Higher Level Indirection

$$\frac{\begin{array}{ll} & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ [\alpha, L_e, J_e, i-1]) \parallel C_1) \\ \gamma(x) = (l,\ \mathrm{private}\ bty*) & \sigma_1(l) = (\omega,\ \mathrm{private}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, 1)) \\ & \mathrm{DecodePtr}(\mathrm{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i] \\ i > 1 & \mathrm{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \mathrm{acc}, \mathrm{private}\ bty*) = (\Delta_2, L_1) \\ & \mathrm{UpdatePtr}(\sigma_1,\ (l_1, \mu_1),\ [\alpha, L_e, J_e, i-1], \mathrm{private}\ bty*) = (\sigma_2, 1) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[wdp2])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

Fig. 13. SMC^2 semantic rules for pointer dereference write.

1.5 Array Rules

Aside from reading and writing to a private index, all array operations will occur locally. This is because we are simply accessing data at or copying data to a known position in our local memory - we do not need to know anything further about the data during these operations, therefore no communication between parties is needed in these rules.

Public Array Declaration

$$\frac{\begin{array}{ll} \text{acc} = 0 & (ty = \text{public } bty) \\ (e) \nvdash \gamma & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, \alpha) \parallel C_1) \\ \alpha > 0 & \omega = \text{EncodePtr}(\text{public const } bty*, [1, [(l_1, 0)], [1], 1]) \\ l = \phi() & \omega_1 = \text{EncodeArr}(\text{public } bty, \alpha, \text{NULL}) \\ l_1 = \phi() & \gamma_1 = \gamma[x \to (l, \text{public const } bty*)] \\ & \sigma_2 = \sigma_1[l \to (\omega, \text{public const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const } bty*, \text{public}, 1))] \\ & \sigma_3 = \sigma_2[l_1 \to (\omega_1, \text{public } bty, \alpha, \text{PermL}(\text{Freeable}, \text{public } bty, \text{public}, \alpha))] \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1, \sigma_3, \Delta, \text{acc}, \text{skip}) \parallel C_1)}$$

Private Array Declaration

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((ty = \text{private } bty) \vee (ty = bty)) \wedge ((bty = \text{int}) \vee (bty = \text{float})) \\ & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, \alpha) \parallel C_1) \\ \alpha > 0 & \omega = \text{EncodePtr}(\text{private const } bty*, [1, [(l_1, 0)], [1], 1]) \\ l = \phi() & \omega_1 = \text{EncodeArr}(\text{private } bty, \alpha, \text{NULL}) \\ l_1 = \phi() & \gamma_1 = \gamma[x \to (l, \text{private const } bty*)] \\ & \sigma_2 = \sigma_1[l \to (\omega, \text{private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1))] \\ & \sigma_3 = \sigma_2[l_1 \to (\omega_1, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha))] \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1, \sigma_3, \Delta, \text{acc}, \text{skip}) \parallel C_1)}$$

Array Declaration Assignment

$$\frac{\begin{array}{l} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x[e_1]) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1) \\ ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, x = e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_1, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2) \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[das])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma_1, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)}$$

Public Array Read Public Index

$$\frac{\begin{array}{ll} (e) \nvdash \gamma & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, i) \parallel C_1) \\ \gamma(x) = (l, \text{public const } bty*) & \sigma_1(l) = (\omega, \text{public const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const } bty*, \text{public}, 1)) \\ & \text{DecodePtr}(\text{public const } bty*, 1, \omega) = [1, [(l_1, 0)], [1], 1] \\ & \sigma_1(l_1) = (\omega_1, \text{public } bty, \alpha, \text{PermL}(\text{Freeable}, \text{public } bty, \text{public}, \alpha)) \\ 0 \le i \le \alpha - 1 & \text{DecodeArr}(\text{public } bty, i, \omega_1) = n_i \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[ra])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,i)])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n_i) \parallel C_1)}$$

Private Array Read Public Index

$$\frac{\begin{array}{ll} \gamma(x) = (l, \text{private const } bty*) & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, i) \parallel C_1) \\ (e) \nvdash \gamma & \sigma_1(l) = (\omega, \text{private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1)) \\ & \text{DecodePtr}(\text{private const } bty*, 1, \omega) = [1, [(l_1, 0)], [1], 1] \\ 0 \le i \le \alpha - 1 & \sigma_1(l_1) = (\omega_1, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha)) \\ & \text{DecodeArr}(\text{private } bty, i, \omega_1) = n_i \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[ra1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,i)])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n_i) \parallel C_1)}$$

Fig. 14. SMC^2 semantic rules for array declarations and reading from a public index.

Public Array Write Public Value Public Index

$$\dfrac{\begin{array}{ll} \mathrm{acc} = 0 & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C)\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e_1,\ e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ \gamma(x) = (l,\ \mathrm{public\ const}\ bty*) & \sigma_2(l) = (\omega,\ \mathrm{public\ const}\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{public\ const}\ bty*,\ \mathrm{public},\ 1)) \\ & \mathrm{DecodePtr}(\mathrm{public\ const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1,\ 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \mathrm{public}\ bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \mathrm{public}\ bty,\ \mathrm{public},\ \alpha)) \\ 0 \leq i \leq \alpha - 1 & \mathrm{UpdateArr}(\sigma_2,\ (l_1,\ i),\ n,\ \mathrm{public}\ bty) = \sigma_3 \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C)\ \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wa])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,i)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Private Array Write Private Value Public Index

$$\dfrac{\begin{array}{ll} \gamma(x) = (l,\ \mathrm{private\ const}\ bty*) & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C)\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e_1) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ (e_2) \vdash \gamma & \sigma_2(l) = (\omega,\ \mathrm{private\ const}\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{private\ const}\ bty*,\ \mathrm{private},\ 1)) \\ & \mathrm{DecodePtr}(\mathrm{private\ const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1,\ 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \mathrm{private}\ bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \mathrm{private}\ bty,\ \mathrm{private},\ \alpha)) \\ 0 \leq i \leq \alpha - 1 & \mathrm{DynamicUpdate}(\Delta_2,\ \sigma_2,\ [(l_1,\ i)],\ \mathrm{acc},\ \mathrm{private}\ bty) = \Delta_3 \\ & \mathrm{UpdateArr}(\sigma_2,\ (l_1,\ i),\ n,\ \mathrm{private}\ bty) = \sigma_3 \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C)\ \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wa2])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,i)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Private Array Write Public Value Public Index

$$\dfrac{\begin{array}{ll} \gamma(x) = (l,\ \mathrm{private\ const}\ bty*) & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C)\ \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e_1,\ e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ & \sigma_2(l) = (\omega,\ \mathrm{private\ const}\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{private\ const}\ bty*,\ \mathrm{private},\ 1)) \\ & \mathrm{DecodePtr}(\mathrm{private\ const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1,\ 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \mathrm{private}\ bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \mathrm{private}\ bty,\ \mathrm{private},\ \alpha)) \\ 0 \leq i \leq \alpha - 1 & \mathrm{DynamicUpdate}(\Delta_2,\ \sigma_2,\ [(l_1,\ i)],\ \mathrm{acc},\ \mathrm{private}\ bty) = \Delta_3 \\ & \mathrm{UpdateArr}(\sigma_2,\ (l_1,\ i),\ \mathrm{encrypt}(n),\ \mathrm{private}\ bty) = \sigma_3 \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C)\ \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wa1])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,i)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Fig. 15. SMC^2 semantic rules for writing to an array.

Read Entire Array

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ a \text{ const } bty*) & \sigma(l) = (\omega,\ a \text{ const } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable},\ a \text{ const } bty*,\ a,\ 1)) \\ & \text{DecodePtr}(a \text{ const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma(l_1) = (\omega_1,\ a\ bty,\ \alpha,\ \text{PermL}(\text{Freeable},\ a\ bty,\ a,\ \alpha)) \\ & \forall i \in \{0...\alpha-1\} \quad \text{DecodeArr}(a\ bty,\ i,\ \omega_1) = n_i \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[rea])}^{(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ [n_0,\ ...,\ n_{\alpha-1}]) \parallel C)}$$

Write Entire Public Array

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{public const } bty*) & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ [n_0,\ ...,\ n_{\alpha_e-1}]) \parallel C_1) \\ \text{acc} = 0 & \sigma_1(l) = (\omega,\ \text{public const } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable},\ \text{public const } bty*,\ \text{public},\ 1)) \\ (e) \nvdash \gamma & \text{DecodePtr}(\text{public const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_1(l_1) = (\omega_1,\ \text{public } bty,\ \alpha,\ \text{PermL}(\text{Freeable},\ \text{public } bty,\ \text{public},\ \alpha)) \\ \alpha_e = \alpha & \forall i \in \{0...\alpha-1\} \quad \text{UpdateArr}(\sigma_{1+i},\ (l_1,\ i),\ n_i,\ \text{public } bty) = \sigma_{2+i} \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wea])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p},\ \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Write Entire Private Array

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{private const } bty*) & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ [n_0,\ ...,\ n_{\alpha_e-1}]) \parallel C_1) \\ (e) \vdash \gamma & \sigma_1(l) = (\omega,\ \text{private const } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable},\ \text{private const } bty*,\ \text{private},\ 1)) \\ & \text{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_1(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \text{PermL}(\text{Freeable},\ \text{private } bty,\ \text{private},\ \alpha)) \\ \alpha_e = \alpha & \forall i \in \{0...\alpha-1\} \quad \text{UpdateArr}(\sigma_{1+i},\ (l_1,\ i),\ n_i,\ \text{private } bty) = \sigma_{2+i} \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wea1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p},\ \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Private Array Write Entire Public Array

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{private const } bty*) & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ [n_0,\ ...,\ n_{\alpha_e-1}]) \parallel C_1) \\ (e) \nvdash \gamma & \sigma_1(l) = (\omega,\ \text{private const } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable},\ \text{private const } bty*,\ \text{private},\ 1)) \\ & \text{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_1(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \text{PermL}(\text{Freeable},\ \text{private } bty,\ \text{private},\ \alpha)) \\ \alpha_e = \alpha & \forall i \in \{0...\alpha-1\} \quad \text{UpdateArr}(\sigma_{1+i},\ (l_1,\ i),\ \text{encrypt}(n_i),\ \text{private } bty) = \sigma_{2+i} \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wea2])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p},\ \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Fig. 16. SMC^2 semantic rules for reading and writing an entire array.

Public Array Read Out of Bounds Public Index

$$\frac{\begin{array}{ll} \gamma(x) = (l,\ \text{public const } bty*) & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e) \nvdash \gamma & \sigma_1(l) = (\omega,\ \text{public const } bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \text{public const } bty*,\ \mathrm{public},\ 1)) \\ & \mathrm{DecodePtr}(\text{public const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_1(l_1) = (\omega_1,\ \text{public } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \text{public } bty,\ \mathrm{public},\ \alpha)) \\ (i < 0) \lor (i \geq \alpha) & \mathrm{ReadOOB}(i,\ \alpha,\ l_1,\ \text{public } bty,\ \sigma_1) = (n,\ 1,\ (l_2,\ \mu)) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[rao])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)}$$

Private Array Read Out of Bounds Public Index

$$\frac{\begin{array}{ll} \gamma(x) = (l,\ \text{private const } bty*) & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e) \nvdash \gamma & \sigma_1(l) = (\omega,\ \text{private const } bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \text{private const } bty*,\ \mathrm{private},\ 1)) \\ & \mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_1(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \text{private } bty,\ \mathrm{private},\ \alpha)) \\ (i < 0) \lor (i \geq \alpha) & \mathrm{ReadOOB}(i,\ \alpha,\ l_1,\ \text{private } bty,\ \sigma_1) = (n,\ 1,\ (l_2,\ \mu)) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[rao1])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)}$$

Public Array Write Out of Bounds Public Index Public Value

$$\frac{\begin{array}{ll} (e_1,\ e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ \mathrm{acc} = 0 & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ \gamma(x) = (l,\ \text{public const } bty*) & \sigma_2(l) = (\omega,\ \text{public const } bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \text{public const } bty*,\ \mathrm{public},\ 1)) \\ & \mathrm{DecodePtr}(\text{public const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \text{public } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \text{public } bty,\ \mathrm{public},\ \alpha)) \\ (i < 0) \lor (i \geq \alpha) & \mathrm{WriteOOB}(n,\ i,\ \alpha,\ l_1,\ \text{public } bty,\ \sigma_2,\ \mathrm{acc}) = (\sigma_3,\ 1,\ (l_2,\ \mu)) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Private Array Write Out of Bounds Public Index Private Value

$$\frac{\begin{array}{ll} \gamma(x) = (l,\ \text{private const } bty*) & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ (e_1) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ (e_2) \vdash \gamma & \sigma_2(l) = (\omega,\ \text{private const } bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \text{private const } bty*,\ \mathrm{private},\ 1)) \\ & \mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \text{private } bty,\ \mathrm{private},\ \alpha)) \\ (i < 0) \lor (i \geq \alpha) & \mathrm{WriteOOB}(n,\ i,\ \alpha,\ l_1,\ \text{private } bty,\ \sigma_2,\ \Delta_2,\ \mathrm{acc}) = (\sigma_3,\ \Delta_3,\ 1,\ (l_2,\ \mu)) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao2])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Private Array Write Public Value Out of Bounds Public Index

$$\frac{\begin{array}{ll} (e_1,\ e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \\ \gamma(x) = (l,\ \text{private const } bty*) & ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2) \\ & \sigma_2(l) = (\omega,\ \text{private const } bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \text{private const } bty*,\ \mathrm{private},\ 1)) \\ & \mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable},\ \text{private } bty,\ \mathrm{private},\ \alpha)) \\ (i < 0) \lor (i \geq \alpha) & \mathrm{WriteOOB}(\mathrm{encrypt}(n),\ i,\ \alpha,\ l_1,\ \text{private } bty,\ \sigma_2,\ \Delta_2,\ \mathrm{acc}) = (\sigma_3,\ \Delta_3,\ 1,\ (l_2,\ \mu)) \end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao1])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p},\ \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)}$$

Fig. 17. SMC[2] semantic rules for reading and writing out of bounds for arrays.

1.6 Pre-Increment Rules

Incrementing a private int value occurs locally. Incrementing the locations of pointers (public and private) will always be local, as all locations pointed to by the pointer will be incremented by the appropriate amount, regardless of which is the true location. This does not modify which is the true location, nor require knowing which is the true location.

Pre-Increment Private Int Variable

$$\frac{\begin{array}{ll}\gamma(x) = (l, \text{private int}) & \sigma(l) = (\omega, \text{private int}, 1, \text{PermL}(\text{Freeable}, \text{private int}, \text{private}, 1)) \\ & \text{DecodeVal}(\text{private int}, \omega) = n_1 \\ n_2 = n_1 + \text{encrypt}(1) & \text{UpdateVal}(\sigma, l, n_2, \text{private int}) = \sigma_1\end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, {++}\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin3])} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n_2) \parallel C)}$$

Pre-Increment Private Pointer Multiple Locations

$$\frac{\begin{array}{ll}\gamma(x) = (l, \text{ private } bty*) & \sigma(l) = (\omega, \text{ private } bty*, \ \alpha, \ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) \\ & \text{DecodePtr}(\text{private } bty*, \ \alpha, \ \omega) = [\alpha, \ L, \ J, \ 1] \\ & \text{IncrementList}(L, \ \tau(\text{private } bty), \sigma) = (L_1, 1) \\ & \text{UpdatePtr}(\sigma, \ (l, 0), \ [\alpha, \ L_1, \ J, \ 1], \ \text{private } bty*) = (\sigma_1, \ 1)\end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ {++}\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin4])} ((\text{p}, \gamma, \ \sigma_1, \ \Delta, \ \text{acc}, \ [n, \ L_1, \ J, \ 1]) \parallel C)}$$

Pre-Increment Private Pointer Higher Level Indirection Multiple Locations

$$\frac{\begin{array}{ll}\gamma(x) = (l, \text{ private } bty*) & \sigma(l) = (\omega, \text{ private } bty*, \ \alpha, \ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) \\ & \text{DecodePtr}(\text{private } bty*, \ \alpha, \ \omega) = [\alpha, \ L, \ J, \ i] \\ & \text{IncrementList}(L, \ \tau(\text{private } bty*), \sigma) = (L_1, 1) \\ & \text{UpdatePtr}(\sigma, \ (l, 0), \ [\alpha, \ L_1, \ J, \ i], \ \text{private } bty*) = (\sigma_1, \ 1)\end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ {++}\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin5])} ((\text{p}, \gamma, \ \sigma_1, \ \Delta, \ \text{acc}, \ [\alpha, \ L_1, \ J, \ i]) \parallel C)}$$

Pre-Increment Private Pointer Single Location

$$\frac{\begin{array}{ll}\gamma(x) = (l, \text{ private } bty*) & \sigma(l) = (\omega, \text{ private } bty*, \ 1, \ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, 1)) \\ & \text{DecodePtr}(\text{private } bty*, \ 1, \ \omega) = [1, \ [(l_1, \mu_1)], \ [1], \ 1] \\ & ((l_2, \mu_2), 1) = \text{GetLocation}((l_1, \mu_1), \tau(\text{private } bty), \sigma) \\ & \text{UpdatePtr}(\sigma, \ (l, 0), \ [1, \ [(l_2, \mu_2)], \ [1], \ 1], \ \text{private } bty*) = (\sigma_1, \ 1)\end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ {++}\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin6])} ((\text{p}, \gamma, \ \sigma_1, \ \Delta, \ \text{acc}, \ (l_2, \mu_2)) \parallel C)}$$

Pre-Increment Private Pointer Higher Level Indirection Single Location

$$\frac{\begin{array}{ll}\gamma(x) = (l, \text{ private } bty*) & \sigma(l) = (\omega, \text{ private } bty*, \ 1, \ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, 1)) \\ & \text{DecodePtr}(\text{private } bty*, \ 1, \ \omega) = [1, \ [(l_1, \mu_1)], \ [1], \ i] \\ i > 1 & ((l_2, \mu_2), 1) = \text{GetLocation}((l_1, \mu_1), \tau(\text{private } bty*), \sigma) \\ & \text{UpdatePtr}(\sigma, \ (l, 0), \ [1, \ [(l_2, \mu_2)], \ [1], \ i], \ \text{private } bty) = (\sigma_1, \ 1)\end{array}}{((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ {++}\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin7])} ((\text{p}, \gamma, \ \sigma_1, \ \Delta, \ \text{acc}, \ (l_2, \mu_2)) \parallel C)}$$

Fig. 18. SMC^2 pre-increment rules for private int values and for private pointers.

Pre-Increment Public Variable

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{public } bty) & \sigma(l) = (\omega,\ \text{public } bty,\ 1,\ \text{PermL(Freeable, public } bty\text{, public, 1))} \\ \text{acc} = 0 & \text{DecodeVal(public } bty,\ \omega) = n \\ n_1 = n + 1 & \text{UpdateVal}(\sigma,\ l,\ n_1,\ \text{public } bty) = \sigma_1\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p},[pin])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ n_1) \parallel C)}$$

Pre-Increment Public Pointer Single Location

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{public } bty*) & \sigma(l) = (\omega,\ \text{public } bty*,\ 1,\ \text{PermL\_Ptr(Freeable, public } bty*\text{, public, 1))} \\ & \text{DecodePtr(public } bty*,\ 1,\ \omega) = [1,\ [(l_1,\ \mu_1)],\ [1],\ 1] \\ & ((l_2,\ \mu_2),\ 1) = \text{GetLocation}((l_1,\ \mu_1),\ \tau(\text{public } bty),\ \sigma) \\ & \text{UpdatePtr}(\sigma,\ (l,\ 0),\ [1,\ [(l_2,\ \mu_2)],\ [1],\ 1],\ \text{public } bty*) = (\sigma_1,\ 1)\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p},[pin1])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2,\ \mu_2)) \parallel C)}$$

Pre-Increment Public Pointer Higher Level Indirection Single Location

$$\frac{\begin{array}{ll}\gamma(x) = (l,\ \text{public } bty*) & \sigma(l) = (\omega,\ \text{public } bty*,\ 1,\ \text{PermL\_Ptr(Freeable, public } bty*\text{, public, 1))} \\ & \text{DecodePtr(public } bty*,\ 1,\ \omega) = [1,\ [(l_1,\ \mu_1)],\ [1],\ i] \\ i > 1 & ((l_2,\ \mu_2),\ 1) = \text{GetLocation}((l_1,\ \mu_1),\ \tau(\text{public } bty*),\ \sigma) \\ & \text{UpdatePtr}(\sigma,\ (l,\ 0),\ [1,\ [(l_2,\ \mu_2)],\ [1],\ i],\ \text{public } bty) = (\sigma_1,\ 1)\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p},[pin2])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2,\ \mu_2)) \parallel C)}$$

Fig. 19. Additional SMC^2 semantic rules for the Public Pre-Increment Operator

1.7 Memory Management Rules

Memory allocation (public or private) occurs locally. Freeing allocated memory from a pointer with a single location occurs locally, regardless of if the pointer is public or private. This is because the true location of the pointer is publicly known.

Public Malloc

$$\frac{\text{acc} = 0 \quad (e) \nvdash \gamma \quad ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n) \parallel C_1) \quad l = \phi() \quad \sigma_2 = \sigma_1\big[l \to \big(\text{NULL}, \text{void}*, n, \text{PermL}(\text{Freeable}, \text{void}*, \text{public}, n)\big)\big]}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{malloc}(e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[mal])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma, \sigma_2, \Delta, \text{acc}, (l, 0)) \parallel C_1)}$$

Private Malloc

$$\frac{(e) \nvdash \gamma \quad (ty = \text{private } bty*) \vee (ty = \text{private } bty) \quad \text{acc} = 0 \quad ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n) \parallel C_1) \quad l = \phi() \quad \sigma_2 = \sigma_1\big[l \to \big(\text{NULL}, \text{void}*, n \cdot \tau(ty), \text{PermL}(\text{Freeable}, \text{void}*, \text{private}, n \cdot \tau(ty))\big)\big]}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{pmalloc}(e, ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[malp])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma, \sigma_2, \Delta, \text{acc}, (l, 0)) \parallel C_1)}$$

Public Free

$$\frac{\gamma(x) = (l, \text{public } bty*) \quad \sigma(l) = (\omega, \text{public } bty*, 1, \text{PermL}(\text{Freeable}, \text{public } bty*, \text{public}, 1)) \quad \text{acc} = 0 \quad \text{DecodePtr}(\text{public } bty*, 1, \omega) = [1, [(l_1, 0)], [1], 1] \quad \text{CheckFreeable}(\gamma, [(l_1, 0)], [1], \sigma) = 1 \quad \text{Free}(\sigma, l_1) = (\sigma_1, (l_1, 0))}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{free}(x)) \parallel C) \Downarrow_{(\text{p},[fre])}^{(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)}$$

Private Free Single Location

$$\frac{\gamma(x) = (l, \text{private } bty*) \quad \sigma(l) = (\omega, \text{private } bty*, 1, \text{PermL}(\text{Freeable}, \text{private } bty*, \text{private}, 1)) \quad \text{acc} = 0 \quad \text{DecodePtr}(\text{private } bty*, 1, \omega) = [1, [(l_1, 0)], [j], 1] \quad \text{CheckFreeable}(\gamma, [(l_1, 0)], [j], \sigma) = 1 \quad \text{Free}(\sigma, l_1) = (\sigma_1, (l_1, 0))}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{pfree}(x)) \parallel C) \Downarrow_{(\text{p},[pfre])}^{(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)}$$

Fig. 20. SMC^2 semantic rules for memory allocation and deallocation.

Cast Private Location

$$\frac{\begin{array}{ll}(ty = \text{private } bty*) & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1) \\ & \sigma_1 = \sigma_2\big[l \to \big(\omega,\ \text{void}*,\ n,\ \text{PermL\_Ptr}(\text{Freeable, void}*, \text{private}, n)\big)\big] \\ & \sigma_3 = \sigma_2\big[l \to \big(\omega,\ ty,\ \dfrac{n}{\tau(ty)},\ \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, \dfrac{n}{\tau(ty)})\big)\big]\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)}$$

Cast Public Location

$$\frac{\begin{array}{ll}(ty = \text{public } bty*) & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1) \\ \text{acc} = 0 & \sigma_1 = \sigma_2\big[l \to \big(\omega,\ \text{void}*,\ n,\ \text{PermL\_Ptr}(\text{Freeable, void}*, \text{public}, n)\big)\big] \\ & \sigma_3 = \sigma_2\big[l \to \big(\omega,\ ty,\ \dfrac{n}{\tau(ty)},\ \text{PermL\_Ptr}(\text{Freeable}, ty, \text{public}, \dfrac{n}{\tau(ty)})\big)\big]\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)}$$

Cast Public Value

$$\frac{\begin{array}{ll}(e) \nvdash \gamma & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1) \\ (ty = \text{public } bty) & n_1 = \text{Cast}(\text{public},\ ty,\ n)\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cv])}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)}$$

Cast Private Value

$$\frac{\begin{array}{ll}(e) \vdash \gamma & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1) \\ (ty = \text{private } bty) & n_1 = \text{Cast}(\text{private},\ ty,\ n)\end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cv1])}^{\mathcal{L}_1} ((\text{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)}$$

Address Of

$$\frac{\gamma(x) = (l,\ ty)}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \&x) \parallel C) \Downarrow_{(\text{p},[loc])}^{\epsilon} ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (l, 0)) \parallel C)}$$

Size of Type

$$\frac{(ty) \nvdash \gamma \qquad n = \tau(ty)}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{sizeof}(ty)) \parallel C) \Downarrow_{(\text{p},[ty])}^{\epsilon} ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C)}$$

Fig. 21. SMC^2 semantic rules for casting and obtaining a memory address and size of type

1.8 Function Rules

At the top level (as shown within our function rules), functions do not need to be executed in a multiparty setting. Given our model uses big-step semantics, we show the overall results of executing the statement(s) for the function - any statements that require multiparty execution will be subsequently executed using their respective multiparty rules, without requiring the top-level rules to be executed in a multiparty setting.

When functions are defined, we evaluate whether or not they have public side effects. This is necessary to know which functions should not be allowed to execute within either branch of a private-conditioned if else statement, as neither branch can have public side effects in order to prevent leakage of information about which branch was intended to be executed (and therefore leakage about the private condition itself).

Function Declaration

$$\frac{\begin{array}{lll} acc = 0 & l = \phi() & \text{GetFunTypeList}(P) = tyL \\ \gamma_1 = \gamma[x \rightarrow (l,\ tyL \rightarrow ty)] & & \sigma_1 = \sigma[l \rightarrow (\text{NULL},\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public}))] \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ ty\ x(P)) \parallel C) \Downarrow_{(\text{p},[df])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma_1,\ \sigma_1,\ \Delta,\ acc,\ \text{skip}) \parallel C)}$$

Function Definition

$$\frac{\begin{array}{lll} acc = 0 & \text{GetFunTypeList}(P) = tyL & \text{CheckPublicEffects}(s,\ x,\ \gamma,\ \sigma) = n \\ x \notin \gamma & \text{EncodeFun}(s,\ n,\ P) = \omega & \\ l = \phi() & \gamma_1 = \gamma[x \rightarrow (l,\ tyL \rightarrow ty)] & \sigma_1 = \sigma[l \rightarrow (\omega,\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public}))] \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fd])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma_1,\ \sigma_1,\ \Delta,\ acc,\ \text{skip}) \parallel C)}$$

Pre-Declared Function Definition

$$\frac{\begin{array}{lll} acc = 0 & x \in \gamma & \text{CheckPublicEffects}(s,\ x,\ \gamma,\ \sigma) = n \\ \gamma(x) = (l,\ tyL \rightarrow ty) & & \sigma = \sigma_1[l \rightarrow (\text{NULL},\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public}))] \\ \text{EncodeFun}(s,\ n,\ P) = \omega & & \sigma_2 = \sigma_1[l \rightarrow (\omega,\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public}))] \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fpd])}^{(\text{p},[(l,0)])} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta,\ acc,\ \text{skip}) \parallel C)}$$

Function Call Without Public Side Effects

$$\frac{\begin{array}{ll} \gamma(x) = (l,\ tyL \rightarrow ty) & \sigma(l) = (\omega,\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public})) \\ \text{DecodeFun}(\omega) = (s,\ n,\ P) & \text{GetFunParamAssign}(P,\ E) = s_1 \\ n = 0 & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, acc, \text{skip}) \parallel C_1) \\ & ((\text{p}, \gamma_1, \sigma_1, \Delta_1, acc, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, acc, \text{skip}) \parallel C_2) \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc1])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ acc,\ \text{skip}) \parallel C_2)}$$

Function Call With Public Side Effects

$$\frac{\begin{array}{ll} \gamma(x) = (l,\ tyL \rightarrow ty) & \sigma(l) = (\omega,\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun}(\text{public})) \\ \text{DecodeFun}(\omega) = (s,\ n,\ P) & \text{GetFunParamAssign}(P,\ E) = s_1 \\ acc = 0 & ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, acc, \text{skip}) \parallel C_1) \\ n = 1 & ((\text{p}, \gamma_1, \sigma_1, \Delta_1, acc, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, acc, \text{skip}) \parallel C_2) \end{array}}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ acc,\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ acc,\ \text{skip}) \parallel C_2)}$$

Fig. 22. SMC^2 semantic rules for functions.

1.9 Binary Operation Rules

Public Addition

$$\frac{(e_1, e_2) \nvdash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad n_1 + n_2 = n_3}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [bp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)}$$

Public Subtraction

$$\frac{(e_1, e_2) \nvdash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad n_1 - n_2 = n_3}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 - e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [bs])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)}$$

Public Multiplication

$$\frac{(e_1, e_2) \nvdash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad n_1 \cdot n_2 = n_3}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 \cdot e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [bm])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)}$$

Public Division

$$\frac{(e_1, e_2) \nvdash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \quad ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad n_1 \div n_2 = n_3}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 \div e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [bd])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)}$$

Fig. 23. SMC^2 semantics for public addition, subtraction, and multiplication.

Public Less Than True

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 < n_2) = 1}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 1) \parallel C_2)}$$

Public Less Than False

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 < n_2) = 0}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 0) \parallel C_2)}$$

Public Equal To True

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 = n_2) = 1}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[eqt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 1) \parallel C_2)}$$

Public Equal To False

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 = n_2) = 0}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[eqf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 0) \parallel C_2)}$$

Public Not Equal To True

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 \neq n_2) = 1}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 != e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[net])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 1) \parallel C_2)}$$

Public Not Equal To False

$$\frac{(e_1, e_2) \nvDash \gamma \qquad ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n_1) \parallel C_1) \\ ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_2) \parallel C_2) \qquad (n_1 =\neq n_2) = 0}{((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 != e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[nef])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, 0) \parallel C_2)}$$

Fig. 24. SMC^2 semantics for public comparisons.

1.10 General Rules

Public Declaration

$$\frac{\begin{array}{c}(ty = \text{public } bty) \qquad \text{acc} = 0 \qquad l = \phi() \qquad \gamma_1 = \gamma[x \to (l,\ ty)] \\ \omega = \text{EncodeVal}(ty, \text{NULL}) \qquad \sigma_1 = \sigma[l \to (\omega,\ ty,\ 1,\ \text{PermL}(\text{Freeable}, ty, \text{public}, 1))]\end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [dv])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)}$$

Private Declaration

$$\frac{\begin{array}{c}((ty = bty) \vee (ty = \text{private } bty)) \wedge ((bty = \text{int}) \vee (bty = \text{float})) \qquad l = \phi() \\ \omega = \text{EncodeVal}(ty, \text{NULL}) \qquad \gamma_1 = \gamma[x \to (l, ty)] \qquad \sigma_1 = \sigma[l \to (\omega, ty, 1, \text{PermL}(\text{Freeable}, ty, \text{private}, 1))]\end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [d1])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)}$$

Fig. 25. SMC^2 Declaration Semantics.

Statement Sequencing

$$\frac{\begin{array}{c}((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, v_1) \parallel C_1) \\ ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, s_2) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, v_2) \parallel C_2)\end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1;\ s_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss])} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, v_2) \parallel C_2)}$$

Statement Block

$$\frac{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, v) \parallel C_1)}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \{s\}) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [sb])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)}$$

Parentheses

$$\frac{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, v) \parallel C_1)}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (e)) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [ep])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, v) \parallel C_1)}$$

Declaration Assignment

$$\frac{\begin{array}{c}((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1) \\ ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, x = e) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\text{p}, \gamma_1, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)\end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ds])} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_2)}$$

Fig. 26. SMC^2 sequencing rules.

Read Public Variable

$$\frac{\gamma(x) = (l,\ \text{public } bty) \qquad \sigma(l) = (\omega,\ \text{public } bty,\ 1,\ \text{PermL(Freeable, public } bty\text{, public, 1)}) \quad \text{DecodeVal(public } bty,\ \omega) = n}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow^{(\text{p},[(l,0)])}_{(\text{p},[r])} ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C)}$$

Read Private Variable

$$\frac{\gamma(x) = (l,\ \text{private } bty) \qquad \sigma(l) = (\omega,\ \text{private } bty,\ 1,\ \text{PermL(Freeable, private } bty\text{, private, 1)}) \quad \text{DecodeVal(private } bty,\ \omega) = n}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow^{(\text{p},[(l,0)])}_{(\text{p},[r1])} ((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C)}$$

Write Public Variable

$$\frac{(e) \nvdash \gamma \qquad ((\text{p},\gamma,\sigma,\Delta,\text{acc},e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p},\gamma,\sigma_1,\Delta_1,\text{acc},n) \parallel C_1) \qquad \gamma(x) = (l,\ \text{public } bty) \qquad \text{UpdateVal}(\sigma_1,\ l,\ n,\ \text{public } bty) = \sigma_2}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[w])} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Write Private Variable

$$\frac{(e) \vdash \gamma \qquad ((\text{p},\gamma,\sigma,\Delta,\text{acc},e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p},\gamma,\sigma_1,\Delta_1,\text{acc},n) \parallel C_1) \qquad \gamma(x) = (l,\ \text{private } bty) \qquad \text{UpdateVal}(\sigma_1,\ l,\ n,\ \text{private } bty) = \sigma_2}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[w1])} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Write Private Variable Public Value

$$\frac{(e) \nvdash \gamma \qquad ((\text{p},\gamma,\sigma,\Delta,\text{acc},e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p},\gamma,\sigma_1,\Delta_1,\text{acc},n) \parallel C_1) \qquad \gamma(x) = (l,\text{private } bty) \qquad \text{UpdateVal}(\sigma_1, l, \text{encrypt}(n), \text{private } bty) = \sigma_2}{((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[w2])} ((\text{p},\ \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)}$$

Fig. 27. SMC^2 reading and writing semantic rules.

SMC Input Public Value

$$\frac{\begin{array}{lll} (e) \nvdash \gamma & \gamma(x) = (l, \text{public } bty) & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1) \\ \text{acc} = 0 & \text{InputValue}(x, n) = n_1 & ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, x = n_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2) \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [inp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)}$$

SMC Input Private Value

$$\frac{\begin{array}{lll} (e) \nvdash \gamma & \gamma(x) = (l, \text{private } bty) & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1) \\ \text{acc} = 0 & \text{InputValue}(x, n) = n_1 & ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, x = n_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2) \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [inp2])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)}$$

SMC Input Public 1D Array

$$\frac{\begin{array}{ll} (e_1, e_2) \nvdash \gamma & ((\text{p}, \gamma, \sigma, \Delta_1, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1) \\ \text{acc} = 0 & ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \alpha) \parallel C_2) \\ \gamma(x) = (l, \text{public const } bty*) & \text{InputArray}(x, n, \alpha) = [m_0, \ldots, m_\alpha] \\ & ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, x = [m_0, \ldots, m_\alpha]) \parallel C_2) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3) \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\text{p}, [inp1])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)}$$

SMC Input Private 1D Array

$$\frac{\begin{array}{ll} (e_1, e_2) \nvdash \gamma & ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1) \\ \text{acc} = 0 & ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \alpha) \parallel C_2) \\ \gamma(x) = (l, \text{private const } bty*) & \text{InputArray}(x, n, \alpha) = [m_0, \ldots, m_\alpha] \\ & ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, x = [m_0, \ldots, m_\alpha]) \parallel C_2) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3) \end{array}}{((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\text{p}, [inp3])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)}$$

Fig. 28. SMC^2 semantic rules for input.

SMC Output Public Value

$$\frac{\begin{array}{ll}(e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \\ \gamma(x) = (l, \text{public } bty) & \sigma_1(l) = (\omega,\ \text{public } bty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, \text{public } bty, \text{public}, 1)) \\ \mathrm{DecodeVal}(\text{public } bty,\ \omega) = n_1 & \mathrm{OutputValue}(x, n, n_1)\end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[out])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

SMC Output Private Value

$$\frac{\begin{array}{ll}(e) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \\ \gamma(x) = (l,\ \text{private } bty) & \sigma_1(l) = (\omega,\ \text{private } bty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, \text{private } bty, \text{private}, 1)) \\ \mathrm{DecodeVal}(\text{private } bty,\ \omega) = n_1 & \mathrm{OutputValue}(x,\ n, n_1)\end{array}}{((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[out2])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p},\ \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)}$$

SMC Output Public Array

$$\frac{\begin{array}{ll}(e_1, e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \\ \gamma(x) = (l, \text{public const } bty*) & ((\mathrm{p},\ \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \alpha) \parallel C_2) \\ & \sigma_2(l) = (\omega, \text{public const } bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \text{public const } bty*, \text{public}, 1)) \\ & \mathrm{DecodePtr}(\text{public const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ \text{public } bty,\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \text{public } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable}, \text{public } bty, \text{public}, \alpha)) \\ & \forall i \in \{0, \dots, \alpha - 1\} \quad \mathrm{DecodeArr}(\text{public } bty,\ i,\ \omega_1) = m_i \\ & \mathrm{OutputArray}(x,\ n,\ [m_0,\ \dots,\ m_{\alpha-1}])\end{array}}{\begin{array}{l}((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e_1,\ e_2)) \parallel C) \\ \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[out1])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,0),\dots,(l_1,\alpha-1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)\end{array}}$$

SMC Output Private Array

$$\frac{\begin{array}{ll}(e_1, e_2) \nvdash \gamma & ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p},\ \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \\ \gamma(x) = (l, \text{private const } bty*) & ((\mathrm{p},\ \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p},\ \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \alpha) \parallel C_2) \\ & \sigma_2(l) = (\omega, \text{private const } bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \text{private const } bty*, \text{private}, 1)) \\ & \mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ \text{private } bty,\ 1] \\ & \sigma_2(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable}, \text{private } bty, \text{private}, \alpha)) \\ & \forall i \in \{0, \dots, \alpha - 1\} \quad \mathrm{DecodeArr}(\text{private } bty,\ i,\ \omega_1) = m_i \\ & \mathrm{OutputArray}(x,\ n,\ [m_0,\ \dots,\ m_{\alpha-1}])\end{array}}{\begin{array}{l}((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e_1,\ e_2)) \parallel C) \\ \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[out3])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,0),\dots,(l_1,\alpha-1)])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)\end{array}}$$

Fig. 29. SMC^2 semantic rules for output.

2 SMC^2 ALGORITHMS

In this section, we present the algorithms used in SMC^2.

2.1 Secure Multiparty Computation Protocols

In our semantics, we leverage multiparty protocols to compartmentalize the complexity of handling private data. In the formal treatment this corresponds to using Axioms in our proofs to reason about protocols. These Axioms allow us to guarantee the desired properties of correctness and noninterference for the overall model, to provide easy integration with new, more efficient protocols as they become available, and to avoid re-proving the formal guarantees for the entire model when new protocols are added. Proving that these Axioms hold is a responsibility of the library implementor in order to have the system fully encompassed by our formal model. Secure multiparty computation protocols that already come with guarantees of correctness and security are the only ones worth considering, so the implementor would only need to ensure that these guarantees match our definitions of correctness and noninterference.

For example, if private values are represented using Shamir secret sharing [2], Algorithm 1, MPC$_{mult}$, represents a simple multiparty protocol for multiplying private values from [1]. In Algorithm 1, lines 2 and 3 define the protocol, while lines 1, 4, and 5 relate the protocol to our semantic representation.

When computation is performed by q parties, at most t of whom may collude ($t < q/2$), Shamir secret sharing encodes a private integer a by choosing a polynomial $f(x)$ of degree t with random coefficients such that $f(0) = a$ (all computation takes place over a finite field). Each participant obtains evaluation of f on a unique non-zero point as their representation of private a; for example, party p obtains $f(\mathrm{p})$. This representation has the property that combining t or fewer shares reveals no information about a as all values of a are equally likely; however, possession of $t + 1$ or more shares permits recovering of $f(x)$ via polynomial interpolation and thus learning $f(0) = a$.

In several of these Multiparty Algorithms, the outer loop (whose condition is $\mathrm{p} \in \{1...\mathrm{q}\}$) indicates that the statements inside the loop would be run in parallel at each party. This notation facilitates showing that all parties are working together to compute the true value for each element that was modified within either branch.

Multiplication in Algorithm 1 corresponds to each party locally multiplying shares of inputs a and b, which computes the product, but raises the polynomial degree to $2t$. The parties consequently re-share their private intermediate results to lower the polynomial degree to t and re-randomize the shares. Values λ_p refer to interpolation coefficients which are derived from the computation setup and party p index.

In order to preserve the correctness and noninterference guarantees of our model when such an algorithm is added, a library developer will need to guarantee that the implementation of this algorithm is correct, meaning that it has the expected input output behavior, and it guarantees noninterference on what is observable.

Algorithm 1 $n_3^\mathrm{p} \leftarrow \mathrm{MPC}_{mult}(n_1^\mathrm{p}, n_2^\mathrm{p})$

1: Let $f_a(\mathrm{p}) = n_1^\mathrm{p}$ and $f_b(\mathrm{p}) = n_1^\mathrm{p}$.
2: Party p computes the value $f_a(\mathrm{p}) \cdot f_b(\mathrm{p})$ and creates its shares by choosing a random polynomial $h_\mathrm{p}(x)$ of degree t, such that $h_\mathrm{p}(0) = f_a(\mathrm{p}) \cdot f_b(\mathrm{p})$. Party p sends to each party i the value $h_\mathrm{p}(i)$.
3: After receiving shares from all other parties, party p computes their share of $a \cdot b$ as the linear combination $H(\mathrm{p}) = \sum_{i=1}^{\mathrm{q}} \lambda_i h_i(\mathrm{p})$.
4: Let $n_3^\mathrm{p} = H(\mathrm{p})$
5: **return** n_3^p

Algorithm 2, MPC_b, is a selection control algorithm that directs the evaluation to the relevant multiparty computation algorithm based on the given binary operation $bop \in \{\cdot, \div, +, -\}$, and Algorithm 3, MPC_b, is a selection control algorithm that directs the evaluation to the relevant multiparty computation algorithm based on the given comparison operation $bop \in \{==, !=, <\}$.

Algorithm 2 $(n_3^1, ..., n_3^q) \leftarrow \text{MPC}_b(bop, [n_1^1, ..., n_1^q], [n_2^1, ..., n_2^q])$

for all p $\in \{1...q\}$ **do**
  $n_3^p = \text{NULL}$
  if $(bop = \cdot)$ **then**
    $n_3^p = \text{MPC}_{mult}(n_1^p, n_2^p)$
  else if $(bop = \div)$ **then**
    $n_3^p = \text{MPC}_{div}(n_1^p, n_2^p)$
  else if $(bop = -)$ **then**
    $n_3^p = \text{MPC}_{sub}(n_1^p, n_2^p)$
  else if $(bop = +)$ **then**
    $n_3^p = \text{MPC}_{add}(n_1^p, n_2^p)$
  end if
end for
return $(n_3^1, ..., n_3^q)$

Algorithm 3 $(n_3^1, ..., n_3^q) \leftarrow \text{MPC}_{cmp}([n_1^1, ..., n_1^q], [n_2^1, ..., n_2^q])$

for all p $\in \{1...q\}$ **do**
  $n_3^p = \text{NULL}$
  if $(bop = ==)$ **then**
    $n_3^p = \text{MPC}_{eq}(n_1^p, n_2^p)$
  else if $(bop = !=)$ **then**
    $n_3^p = \text{MPC}_{neq}(n_1^p, n_2^p)$
  else if $(bop = <)$ **then**
    $n_3^p = \text{MPC}_{lt}(n_1^p, n_2^p)$
  end if
end for
return $(n_3^1, ..., n_3^q)$

Each of the given multiparty protocols in Algorithm 2 (i.e., MPC_{mult}, MPC_{sub}, MPC_{add}, MPC_{div}) and each of the given multiparty protocols in Algorithm 3 (i.e., MPC_{eq}, MPC_{neq}, MPC_{lt}) must be defined using protocols that have been proven to uphold the desired properties within our proofs (i.e., correctness and noninterference). We give an example definition for MPC_{mult} in Algorithm 1, but this definition can be swapped out with any protocol for the secure multiparty computation of multiplication that maintains the properties of correctness and noninterference. We defer the definition of all other SMC binary operations, rely on assertions that the protocols chosen to be used with this model will maintain both correctness and noninterference in our proofs. We chose this strategy as SMC implementations of such protocols will be proven to hold our desired properties on their own, and this allows us to not only leverage those proofs, but to also improve the versatility of our model by allowing such algorithms to be easily swapped out as newer, improved versions become available.

Algorithm 4 $(n_2^1, ..., n_2^q) \leftarrow \text{MPC}_u(uop, [n_1^1, ..., n_1^q])$

if $uop ==$ ++ **then**
  for all p $\in \{1...q\}$ **do**
    $n_2^p = \text{MPC}_{plpl}(n_1^p)$
  end for
  return $(n_2^1, ..., n_2^q)$
end if

Algorithm 4, MPC_{unop}, is like MPC_b in that it is a selection control algorithm for multiparty unary operations. We only include the pre-increment operator here, as that is the only unary operation of this type that is within the scope of our current grammar (i.e., pointer dereferencing with $*$ is handled separately, and the address-of operator & is handled locally). Other types of operations that would be handled here are pre-decrement, post-increment and post-decrement of values, as well as negation. We chose not to include these elements in our grammar as they are trivial extensions of the current grammar.

The following Algorithms are given as placeholders for the SMC definitions of each function; for the model to be complete, these placeholder Algorithms would need to reflect the chosen implementations of each used within the system. Algorithm 5, MPC_{sub}, is for the SMC implementation of subtraction. This algorithm will securely compute whether $n_1^{\text{p}} - n_2^{\text{p}}$ for all parties p. Algorithm 6, MPC_{add}, is for the SMC implementation of addition. This algorithm will securely compute whether $n_1^{\text{p}} + n_2^{\text{p}}$ for all parties p. Algorithm 7, MPC_{div}, is for the SMC implementation of addition. This algorithm will securely compute whether $n_1^{\text{p}} \div n_2^{\text{p}}$ for all parties p. Algorithm 8, MPC_{plpl}, is for the SMC implementation of the pre-increment operation on a value. This algorithm will securely compute $n_1^{\text{p}} + 1$ for all parties p. Algorithm 10, MPC_{eq}, is for the SMC implementation of equality. This algorithm will securely compute whether $n_1^{\text{p}} == n_2^{\text{p}}$ for all parties p. Algorithm 9, MPC_{neq}, is for the SMC implementation of inequality. This algorithm will securely compute whether $n_1^{\text{p}}! = n_2^{\text{p}}$ for all parties p. Algorithm 11, MPC_{lt}, is for the SMC implementation of the less than operation. This algorithm will securely compute whether $n_1^{\text{p}} < n_2^{\text{p}}$ for all parties p.

Algorithm 5 $n_3^{\text{p}} \leftarrow \text{MPC}_{sub}(n_1^{\text{p}}, n_2^{\text{p}})$

Algorithm 6 $n_3^{\text{p}} \leftarrow \text{MPC}_{add}(n_1^{\text{p}}, n_2^{\text{p}})$

Algorithm 7 $n_3^{\text{p}} \leftarrow \text{MPC}_{div}(n_1^{\text{p}}, n_2^{\text{p}})$

Algorithm 8 $n_2^{\text{p}} \leftarrow \text{MPC}_{plpl}(n_1^{\text{p}})$

Algorithm 9 $n_3^{\text{p}} \leftarrow \text{MPC}_{neq}(n_1^{\text{p}}, n_2^{\text{p}})$

Algorithm 10 $n_3^{\text{p}} \leftarrow \text{MPC}_{eq}(n_1^{\text{p}}, n_2^{\text{p}})$

Algorithm 11 $n_3^{\text{p}} \leftarrow \text{MPC}_{lt}(n_1^{\text{p}}, n_2^{\text{p}})$

$\text{MPC}_{resolve}$ and MPC_{free}, though multiparty algorithms, are diverted from this subsection to be shown in Algorithm 22 and 26, respectively, and discussed in conjunction with their corresponding algorithms.

2.2 Private-conditioned branching algorithms

In this section, we will discuss the helper algorithms used when branching on private conditionals. First, we will discuss extraction of what variables are modified and how we choose which strategy to use. Second, we will discuss our variable-based tracking algorithms. Third, we will discuss our location-based tracking algorithms. Finally, we will discuss our multiparty resolution algorithms.

Algorithm 12, Extract, iterates over the statements contained in both branches, checking for which variables are modified (i.e., pre-increment operations, assignment statements) and whether either branch contains a pointer dereference write or array write at a public index. All variables that are modified through pre-increment operations and regular assignment statements are added to the variable list that is returned. Pointer variables that are used in a pointer dereference write are not added to this list, as the data at the location that is referred to is being modified (and not

Algorithm 12 $(x_{mod}, j) \leftarrow \text{Extract}(s_1,\ s_2, \gamma)$

```
j = 0
x_local = [ ]
x_mod = [ ]
for all s ∈ {s_1; s_2} do
    if ((s == ty x) ∨ (s == ty x[e])) then
        x_local.append(x)
    else if ((s == x = e) ∧ (¬x_local.contains(x))) then
        x_mod = x_mod ∪ [x]
        for all e_1 ∈ e do
            if ((e_1 == ++ x_1) ∧ (¬x_local.contains(x_1))) then
                x_mod = x_mod ∪ [x_1]
            end if
        end for
    else if ((s == x[e_1] = e_2) ∧ (¬x_local.contains(x))) then
        if (e_1) ⊢ γ then
            x_mod = x_mod ∪ [x]
        else
            j = 1
        end if
        for all e ∈ {e_1, e_2} do
            if ((e == ++ x_1) ∧ (¬x_local.contains(x_1))) then
                x_mod = x_mod ∪ [x_1]
            end if
        end for
    else if ((s == ++ x) ∧ (¬x_local.contains(x))) then
        x_mod = x_mod ∪ [x]
    else if (s == *x = e) then
        j = 1
        for all e_1 ∈ e do
            if ((e_1 == ++ x_1) ∧ (¬x_local.contains(x_1))) then
                x_mod = x_mod ∪ [x_1]
            end if
        end for
    end if
end for
return (x_mod, j)
```

the data stored at the pointer's location). This is also true for arrays that are only updated at public indices. When a pointer dereference write or array write at a public index is found, we update the tag to be 1 (i.e., true), otherwise the tag remains as 0. We later use this tag to decide whether we can proceed with the standard, flat basic block tracking using temporary variables (when no pointer dereference write operations or potential out of bounds writes occur), or whether we need to use the dynamic basic block tracking using locations.

It is important to note that if a pointer dereference write occurs inside a private-conditioned branch, we must proceed with location tracking at the level that it occurs as well as any outer levels of nesting of private-conditioned branches; however, if a lower level of nesting does not contain any pointer dereference writes, we can use variable tracking at that level. This algorithm will also filter out modifications made to any local variables, as we do not need to track and propagate those modifications outside of this local scope.

It is also important to note here that, currently, when we find an array has been modified as a whole, we simply add the array variable name to the list and track all locations. When an array has been modified at a private index, we add the entire array to be tracked, as we will be modifying all indices within the array to hide the true index that was updated. When an array has been modified at a specific public index, we trigger location tracking. We do this because we cannot easily tell what the value of the index will be at execution when we run Extract (unless the array index is hard-coded, but this is rare), and therefore we do not know if the array access will be in bounds or not.

2.2.1 Variable Tracking Algorithms. Algorithms 13 (InitializeVariables), 14 (RestoreVariables), 15 (ResolveVariables_Retrieve), and 16 (ResolveVariables_Store) are specific to the variable tracking style of conditional code block tracking, as shown in rule Private If Else (Variable Tracking) in Figure 9.

Algorithm 13 $(\gamma_1, \sigma_1, L) \leftarrow \text{InitializeVariables}(x_{list},\ \gamma, \sigma, n, \text{acc})$

1: $l_{res} = \phi(temp)$
2: $\gamma_1 = \gamma[res\_acc \rightarrow (l_{res}, \text{private int})]$
3: $\omega_{res} = \text{EncodeVal}(\text{private int}, n)$
4: $\sigma_1 = \sigma[l_{res} \rightarrow (\omega_{res}, \text{private int}, 1, \text{PermL}(\text{Freeable}, \text{private int}, \text{private}, 1))]$
5: $L = [(l_{res}, 0)]$
6: **for all** $x \in x_{list}$ **do**
7: $\quad (l_x, ty) = \gamma(x)$
8: $\quad l_t = \phi(temp)$
9: $\quad l_e = \phi(temp)$
10: $\quad L = L :: [(l_x, 0), (l_t, 0), (l_e, 0)]$
11: $\quad \gamma_1 = \gamma_1[x\_t\_acc \rightarrow (l_t, ty)][x\_e\_acc \rightarrow (l_e, ty)]$
12: $\quad (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma_1(l_x)$
13: $\quad$ **if** $(ty = \text{private const } bty*)$ **then**
14: $\quad\quad l_{ta} = \phi(temp)$
15: $\quad\quad l_{ea} = \phi(temp)$
16: $\quad\quad [1, [(l_{xa}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_x)$
17: $\quad\quad (\omega_{xa}, \text{private } bty, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty, \text{private}, \alpha)) = \sigma_1(l_{xa})$
18: $\quad\quad \sigma_1 = \sigma_1[l_{ta} \rightarrow (\omega_{xa}, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha))]$
19: $\quad\quad \sigma_1 = \sigma_1[l_{ea} \rightarrow (\omega_{xa}, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha))]$
20: $\quad\quad \omega_t = \text{EncodePtr}(ty, [1, [(l_t, 0)], [1], 1])$
21: $\quad\quad \omega_e = \text{EncodePtr}(ty, [1, [(l_e, 0)], [1], 1])$
22: $\quad\quad \sigma_1 = \sigma_1[l_t \rightarrow (\omega_t, ty, 1, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, 1))]$
23: $\quad\quad \sigma_1 = \sigma_1[l_e \rightarrow (\omega_e, ty, 1, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, 1))]$
24: $\quad\quad$ **for all** $i \in \{0 \ldots \alpha - 1\}$ **do**
25: $\quad\quad\quad L = L :: [(l_{xa}, i), (l_{ta}, i), (l_{ea}, i)]$
26: $\quad\quad$ **end for**
27: $\quad$ **else**
28: $\quad\quad \sigma_1 = \sigma_1[l_t \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha))]$
29: $\quad\quad \sigma_1 = \sigma_1[l_e \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha))]$
30: $\quad$ **end if**
31: **end for**
32: **return** (γ_1, σ_1, L)

First, Algorithm 13 stores the result of the conditional expression (n) in res_{acc} (lines 1:4). It grabs a new temporary variable location (line 1), adds the mapping to the environment (line 2), encodes the value n into its byte-representation (line 3), then adds the mapping into memory (line 4). It is important to note here that we pull new locations from the partition of memory designated for such temporary variables, as this simplifies the mapping of memory between SMC[2] and Vanilla C.

Then, for each variable x in x_{list}, we look up x in the environment and memory (line 7, 12), grab new temporary variable locations (lines 8-9), and create `then` and `else` temporary variables initialized with the value of x. To do this, we first add the mapping of these temporaries to the environment (line 11). When x refers to an entire array, we have the special case of needing to look up the array data from the pointer that refers to it. To handle this, we have split out the behavior for arrays within the `if` branch in the algorithm, and the `else` branch handles both pointers and regular variables appropriately, as these are single-level temporary variables.

If the variable is an array type, we must grab new temporary variable locations for the array data of the `then` and `else` variables (lines 14-15), look up the array data of x (lines 16-17), then add the mappings for both the array data (lines 18-19) and the array pointer (lines 20-23) to memory. For other types of variables, we can simply add the mappings for the `then` and `else` variables to memory directly using the data from x (lines 28-29), as the data that will be changed within the branches for these variables is at this level.

Lines 5, 10, and 24-26 facilitate our analysis of which locations have been accessed or modified, which allows us to more easily reason about this within the rules as needed for our noninterference result.

Algorithm 14 $(\sigma_4, L) \leftarrow \text{RestoreVariables}(x_{list}, \gamma, \sigma, \text{acc})$

1: $L = [\,]$
2: **for all** $x \in x_{list}$ **do**
3: $\quad (l_x, ty) = \gamma(x)$
4: $\quad (l_t, ty) = \gamma(x\_t\_\text{acc})$
5: $\quad (l_e, ty) = \gamma(x\_e\_\text{acc})$
6: $\quad L = L :: [(l_x, 0), (l_t, 0), (l_e, 0)]$
7: $\quad$ **if** $(ty = \text{private const } bty*)$ **then**
8: $\quad\quad (\omega_{xa}, ty, 1, \text{PermL}(\text{Freeable}, ty, \text{private}, 1)) = \sigma(l_x)$
9: $\quad\quad (\omega_{ta}, ty, 1, \text{PermL}(\text{Freeable}, ty, \text{private}, 1)) = \sigma(l_t)$
10: $\quad\quad (\omega_{ea}, ty, 1, \text{PermL}(\text{Freeable}, ty, \text{private}, 1)) = \sigma(l_e)$
11: $\quad\quad [1, [(l_{xa}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_{xa})$
12: $\quad\quad [1, [(l_{ta}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_{ta})$
13: $\quad\quad [1, [(l_{ea}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_{ea})$
14: $\quad\quad \sigma_1[l_{xa} \rightarrow (\omega_t, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha))] = \sigma$
15: $\quad\quad \sigma_2[l_{ta} \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha))] = \sigma_1$
16: $\quad\quad \sigma_3 = \sigma_2[l_{ta} \rightarrow (\omega_t, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)]$
17: $\quad\quad (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma_3(l_{ea})$
18: $\quad\quad \sigma_4 = \sigma_3[l_{xa} \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)]$
19: $\quad\quad$ **for all** $i \in \{0 \ldots \alpha - 1\}$ **do**
20: $\quad\quad\quad L = L :: [(l_{xa}, i), (l_{ta}, i), (l_{ea}, i)]$
21: $\quad\quad$ **end for**
22: $\quad$ **else**
23: $\quad\quad \sigma_1[l_x \rightarrow (\omega_t, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha))] = \sigma$
24: $\quad\quad \sigma_2[l_t \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)] = \sigma_1$
25: $\quad\quad \sigma_3 = \sigma_2[l_t \rightarrow (\omega_t, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)]$
26: $\quad\quad (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma_3(l_e)$
27: $\quad\quad \sigma_4 = \sigma_3[l_x \rightarrow (\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)]$
28: $\quad$ **end if**
29: $\quad \sigma = \sigma_4$
30: **end for**
31: **return** (σ_4, L)

In Algorithm 14, for each variable x within x_{list}, we must save the current value for the variable and then restore it to other value it had before execution of the `then` branch. We first look up x and its associated temporary variables within our environment. Then we proceed to restore based on the type (array vs. non-array). For arrays, we first find where the array data is stored (lines 8-13), then proceed to pull out the data for x and `then` from memory (lines 14-15). We then take the data that was in x, which is the resulting data from the `then` branch, and store it back into memory as the updated mapping for the `then` temporary (line 16). Finally, we look up the original data of x stored in the `else` temporary (line 17), and store it back into memory as the data for x (line 18).

It is useful to note that within this rule, we explicitly show that x currently contains the value for the else branch (ω_t), which we proceed to store in the `then` variable, and the `else` variable contains the original value for x (ω_x), which we proceed to store back into x. When an entire array has been modified, we have the special case of needing to look up the array data from the pointer that refers to it. To handle this, we have split out the behavior for arrays within the `if` branch in the algorithm, and the `else` branch handles both pointers and regular variables appropriately, as these are single-level modifications (no pointer dereference writes occurred).

The behavior of lines 23-27 corresponds to the behavior of lines 14-18, but for variables that are not arrays. This is because the data that was modified for int, float, and pointer variables is at the first level lookup within memory, but for arrays it is stored at one level of indirection due to the structure of array variables as being a const pointer to the larger set of array data. In lines 6 and 19-21, we are facilitating the analysis of which locations have been accessed or modified, which allows us to more easily reason about this within the rules.

Algorithm 15 $(V, n_{res}, L) \leftarrow \text{ResolveVariables\_Retrieve}(x_{list}, \text{acc}, \gamma, \sigma)$

1: $V = [\,]$
2: $(l_{res}, \text{private int}) = \gamma(res\_\text{acc})$
3: $(\omega_{res}, \text{private int}, 1, \text{PermL}(\text{Freeable}, \text{private int}, \text{private}, 1)) = \sigma(l_{res})$
4: $n_{res} = \text{DecodeVal}(\text{private int}, \omega_{res})$
5: $L = [(l_{res}, 0)]$
6: **for all** $x \in x_{list}$ **do**
7:     $(l_x, ty) = \gamma(x)$
8:     $(l_t, ty) = \gamma(x_t)$
9:     $(\omega_x, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma(l_x)$
10:     $(\omega_t, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma(l_t)$
11:     $L = L :: [(l_x, 0), (l_t, 0)]$
12:     **if** $(ty = \text{private } bty)$ **then**
13:         $v_x = \text{DecodeVal}(\text{private } bty, \omega_x)$
14:         $v_t = \text{DecodeVal}(\text{private } bty, \omega_t)$
15:         $V = V.append((v_t, v_x))$
16:     **else if** $(ty = \text{private const } bty*)$ **then**
17:         $[1, [(l_{xa}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_x)$
18:         $[1, [(l_{ta}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_t)$
19:         $(\omega_{xa}, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha)) = \sigma(l_{xa})$
20:         $(\omega_{ta}, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha)) = \sigma(l_{ta})$
21:         **for all** $i \in \{0 \dots \alpha - 1\}$ **do**
22:             $v_{xi} = \text{DecodeArr}(\text{private } bty, i, \omega_{xa})$
23:             $v_{ti} = \text{DecodeArr}(\text{private } bty, i, \omega_{ta})$
24:             $V = V.append((v_{ti}, v_{xi}))$
25:             $L = L :: [(l_{xa}, i), (l_{ta}, i)]$
26:         **end for**
27:     **else if** $(ty = \text{private } bty*)$ **then**
28:         $[\alpha, L_x, J_x, i] = \text{DecodePtr}(ty, \alpha, \omega_x)$
29:         $[\alpha, L_t, J_t, i] = \text{DecodePtr}(ty, \alpha, \omega_t)$
30:         $V = V.append(([\alpha, L_t, J_t, i], [\alpha, L_x, J_x, i]))$
31:     **end if**
32: **end for**
33: **return** (V, n_{res}, L)

In Algorithm 15, we retrieve all of the data needed to resolve what the true value for each modified variable x within x_{list} should be. First, we retrieve the value for the result of the conditional expression (lines 2-4), then we retrieve the values for each variable within x_{list}. We will retrieve the `else` value by looking up the value currently stored in x, as we have just completed execution of the `else` branch (lines 7, 9 and 13/17,19,22/28 by type). We will retrieve the `then` value by looking up the value currently stored in the temporary variable x_t (lines 8,10, and 14/18,20,23/29 by type). We append a tuple of the `then` and `else` values for each variable to the list of values (lines 15/24/30 by type). This list of values will then be used within the multiparty resolve algorithm $\text{MPC}_{resolve}$ to obtain the true values for each variable. As with the previous helper algorithms, we collect a list of which locations we have accessed in order to facilitate our analysis of location accesses.

Algorithm 16 $(\sigma_1, L) \leftarrow \text{ResolveVariables\_Store}(x_{list}, \text{acc}, \gamma, \sigma, V)$

$L = [\,]$
$\sigma_1 = \sigma$
for all $i \in \{0 \ldots |V| - 1\}$ **do**
  $x = x_{list}[i]$
  $v_x = V[i]$
  $(l_x, ty) = \gamma(x)$
  $L = L.append((l_x, 0))$
  if $(ty = \text{private } bty)$ **then**
    $\sigma_2 = \text{UpdateVal}(\sigma_1,\ l_x,\ v_x,\ ty)$
    $\sigma_1 = \sigma_2$
  else if $(ty = \text{private const } bty*)$ **then**
    $[1, [(l_{xa}, 0)], [1], 1] = \text{DecodePtr}(ty, 1, \omega_x)$
    for all $\mu \in \{0 \ldots \alpha - 1\}$ **do**
      $v_\mu = v_x[\mu]$
      $\sigma_{2+\mu} = \text{UpdateArr}(\sigma_{1+\mu},\ (l_{xa}, \mu),\ v_\mu,\ ty)$
      $L = L.append((l_{xa}, \mu))$
    end for
    $\sigma_1 = \sigma_{2+\mu}$
  else if $(ty = \text{private } bty*)$ **then**
    $\sigma_2 = \text{UpdatePtr}(\sigma_1,\ (l_x, 0),\ v_x,\ ty)$
    $\sigma_1 = \sigma_2$
  end if
end for
return (σ_1, L)

Once we have completed resolution of true values, we then use Algorithm 16 to store the true value for each modified variable x within x_{list} back into memory. The list of values maintains its ordering during resolution, so we simply iterate through the list of variables and values, updating each variable with its corresponding value. As with the previous helper algorithms, we collect a list of which locations we have accessed in order to facilitate our analysis of location accesses.

2.2.2 Location Tracking Algorithms. Algorithms 17 (Initialize), 18 (DynamicUpdate), 19 (Restore), 20 (Resolve_Retrieve), and 21 (Resolve_Store) are specific to the location tracking style of conditional code block tracking, as shown in rule Private If Else (Location Tracking) in Figure 9.

It is worthwhile to start by noting the structure of Δ for SMC^2. Δ is a list of lists, with the inner lists storing the mapping of location to data for dynamic tracking at each level of nesting of private-conditioned branches. Each mapping is structured as $(l, \mu) \rightarrow (v_{orig}, v_{then}, j, ty)$, where (l, μ) is the location that is modified (stored as the memory block identifier and offset into the block), v_{orig} is the original data stored in a location, and v_{then} as the data stored in that location at the end of the execution of the `then` branch. The public tag j is set to 0 when a new mapping is added to

Δ, signifying that we have stored data into v_{orig}, but there is currently no data in v_{then}. During restoration between branches, we update j to 1 as we store the data from that location into the map. This is needed due to dynamic tracking of pointer dereference writes and potential out of bounds array accesses - we can see such a modification to a untracked location for the first time in the `else` branch, and this allows us to add these new locations without needing to track which branch we are currently in (i.e., for the current level of nesting and all outer levels, as this may be a new location for all levels). Using this tag, we are able to resolve at all levels of nesting with ease, using the tag to indicate whether we should use v_{orig} or v_{then} as the `then` data in resolution. This tag does not need to be private, as it is visible to an observer whether or not the data at a given location was modified during the execution of either branch.

Algorithm 17 $(\gamma_1, \sigma_1, \Delta_1, L_1) \leftarrow \text{Initialize}(\Delta, x_{list},\ \gamma,\ \sigma,\ \text{acc})$

$l_{res} = \phi(temp)$
$\gamma_1 = \gamma[res\_\text{acc} \rightarrow (l_{res}, \text{private int})]$
$\omega_{res} = \text{EncodeVal}(\text{private int}, n)$
$\sigma_1 = \sigma[l_{res} \rightarrow (\omega_{res}, \text{private int}, 1, \text{PermL}(\text{Freeable}, \text{private int}, \text{private}, 1))]$
$L_1 = [(l_{res}, 0)]$
for all $x \in x_{list}$ **do**
  $(l, ty) = \gamma(x)$
  $L_1 = L_1.append((l, 0))$
  if $(ty == \text{private } bty)$ **then**
    $(\omega, \text{private } bty, 1, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, 1)) = \sigma(l)$
    $v = \text{DecodeVal}(\text{private } bty, \omega)$
    $\Delta_1 = \Delta[\text{acc}].push(((l, 0) \rightarrow (v, \text{NULL}, 0, \text{private } bty)))$
  else if $(ty == \text{private } bty*)$ **then**
    $(\omega,\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) = \sigma(l)$
    $[\alpha,\ L,\ J,\ i] = \text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega)]$
    $\Delta_1 = \Delta[\text{acc}].push(((l, 0) \rightarrow ([\alpha,\ L,\ J,\ i], \text{NULL}, 0, \text{private } bty*)))$
  else if $(ty = \text{private const } bty*)$ **then**
    $(\omega,\ \text{private const } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1)) = \sigma(l)$
    $[1,\ [(l_1, 0)],\ [1],\ 1] = \text{DecodePtr}(\text{private const } bty*,\ 1,\ \omega)]$
    $(\omega_1, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha)) = \sigma_2(l_1)$
    for all $i \in \{0...\alpha - 1\}$ **do**
      $L_1 = L_1.append((l_1, i))$
      $v_i = \text{DecodeArr}(\text{private } bty, i, \omega_1)$
      $\Delta_1 = \Delta_1[\text{acc}].push(((l_1, i) \rightarrow (v_i, \text{NULL}, 0, \text{private } bty)))$
    end for
  end if
  $\Delta = \Delta_1$
end for
return $(\gamma_1, \sigma_1, \Delta_1, L_1)$

Algorithm 17, Initialize, stores the result of the conditional and then iterates through all of the variables within the variable list obtained from Extract, adding a new mapping for the location at which it is stored and storing its current value into the *orig* portion, as well as initializing the tag j for that location as 0. As the modification of the location where a variable is stored is not allowed, it is safe to add all of these locations and their original values into Δ before the execution of the `then` branch. This allows execution of the `then` branch to proceed as normal, only incurring additional costs when a pointer dereference write or an array write at a public index occurs. In this Algorithm, we have built in the tracking of which locations we are accessing, adding them to the variable L_1 and then returning this information to the rule which called it.

Algorithm 18 $(\Delta_1, L_1) \leftarrow \text{DynamicUpdate}(\Delta, \sigma,\ L, \text{acc}, ty)$

if (acc = 0) **then**
  return $(\Delta, [\,])$
end if
$L_1 = [\,]$
$\Delta_1 = \Delta$
for $(l, \mu) \in L$ **do**
  if $((l, \mu) \notin \Delta_1[\text{acc}])$ **then**
    $L_1 = L_1 :: [(l, \mu)]$
    $\sigma_1[l \to (\omega,\ ty',\ \alpha,\ \text{PermL}(\text{Freeable}, ty', \text{private}, \alpha))] = \sigma$
    if $(ty = ty' = \text{private } bty) \wedge (0 \leq \mu < \alpha)$ **then**
      $v = \text{DecodeArr}(ty,\ \mu, \omega)$
      $\Delta_1 = \Delta_1[\text{acc}].push(((l, \mu) \to (v,\ \text{NULL},\ 0, \text{private } bty)))$
    else if $(ty = ty' = \text{private } bty*) \wedge (\mu = 0)$ **then**
      $[\alpha, L_1, J, i] = \text{DecodePtr}(\text{private } bty*, \alpha, \omega)$
      $\Delta_1 = \Delta_1[\text{acc}].push(((l, 0) \to ([\alpha, L_1, J, i], \text{NULL}, 0, \text{private } bty*)))$
    else
      $v = \text{GetBytes}((l, \mu), ty, \sigma)$
      $\Delta_1 = \Delta_1[\text{acc}].push(((l, \mu) \to (v,\ \text{NULL},\ 0, ty)))$
    end if
    if (acc > 0) **then**
      $\Delta_1 = \text{DynamicUpdate}(\Delta_1, \sigma,\ [(l, \mu)], \text{acc} - 1)$
    end if
  end if
end for
return (Δ_1, L_1)

Algorithm 18, DynamicUpdate, is used prior to performing a pointer dereference write, an array write at a public index, and within Algorithm WriteOOB in order to ensure that we are *correctly* tracking *all* locations that get modified. It takes the location that is about to be modified and ensures that this location is either already being tracked by Δ for the current level of nesting, or adds the location and its original value to Δ for that level of nesting. If this location is not already in Δ for the current level of nesting, and we are not in the outer-most private-conditioned branch, it will recursively call itself for all outer levels of nesting. This is to ensure that the location will be properly tracked at all levels. If the location is found to already be tracked at an outer level, it will return. We chose to perform this more costly checking at this point of execution, as we know whether or not the location is new to this level of nesting at this point, and can easily propagate this information upward to the outer levels of nesting here. The most costly check, where this new location needs to be added to all outer levels of nesting, can only occur once for each new location and will only occur once as subsequent modifications will find that the location is already being tracked. This propagation must happen at some point during execution, and would only require additional memory resources if it is not performed at this point, as, in order to put off the propagation until later, it would be necessary to tag this location as one that had been added during this level of nesting. Algorithm GetBytes is used within this Algorithm when the location that is given and what is stored at that location do not match up perfectly, such as the case when the given type and the type at that location do not match (in which case, we would need to grab additional bytes from the next location in order to properly decode a value based on our expected type.

It is important to reiterate here that during an out of bounds array write, DynamicUpdate is called from within WriteOOB in order to properly track the location being written to, since we are overshooting the location containing the array data. For pointer dereference writes, we use this to ensure we are tracking the most current location referred to by the pointer, since it is possible that it has changed during execution of either branch. For array writes at public indices, we must track dynamically (whether out of bounds or in bounds) due to the possibility of an out of bounds access.

Algorithm 19 $(\sigma_2, \Delta_3, L) \leftarrow \text{Restore}(\sigma,\ \Delta,\ \text{acc})$

$\Delta_1 = \Delta$
$L = [\,]$
for all $((l, \mu) \rightarrow (v_{orig}, \text{NULL}, 0, ty)) \in \Delta[\text{acc}]$ **do**
  $v_{then} = \text{NULL}$
  if $(\mu = 0)$ **then**
    if $(ty = \text{private } bty)$ **then**
      $\sigma_1[l \rightarrow (\omega, \text{private } bty, 1, \text{PermL}(\text{Freeable, private } bty, \text{private}, 1))] = \sigma$
      $\omega_1 = \text{EncodeVal}(\text{private } bty, v_{orig})$
      $\sigma_2 = \sigma_1[l \rightarrow (\omega_1, \text{private } bty, 1, \text{PermL}(\text{Freeable, private } bty, \text{private}, 1))]$
      $v_{then} = \text{DecodeVal}(\text{private } bty, \omega)$
    else if $(ty = \text{private } bty*)$ **then**
      $[\alpha_{orig}, L_{orig}, J_{orig}, i] = v_{orig}$
      $\sigma_1[l \rightarrow (\omega_{then}, \text{private } bty*, \alpha_{then}, \text{PermL\_Ptr}(\text{Freeable, private } bty*, \text{private}, \alpha_{then}))] = \sigma$
      $\omega_{orig} = \text{EncodePtr}(ty, [\alpha_{orig}, L_{orig}, J_{orig}, i])$
      $\sigma_2 = \sigma_1[l \rightarrow (\omega_{orig}, \text{private } bty*, \alpha_{orig}, \text{PermL\_Ptr}(\text{Freeable, private } bty, \text{private}, \alpha_{orig}))]$
      $v_{then} = \text{DecodePtr}(\text{private } bty*, \alpha_{then}, \omega_{then})$
    end if
  else
    $v_{then} = \text{GetBytes}((l, \mu), ty, \sigma)$
    $\sigma_2 = \text{SetBytes}((l, \mu), ty, v_{orig}, \sigma)$
  end if
  $\Delta_2[\text{acc}][(l, 0) \rightarrow (v_{orig}, \text{NULL}, 0, ty)] = \Delta_1$
  $\Delta_3 = \Delta_2[\text{acc}][(l, 0) \rightarrow (v_{orig}, v_{then}, 1, ty)]$
  $L = L.append(l, \mu)$
  $\sigma = \sigma_2$
  $\Delta_1 = \Delta_3$
end for
return (σ_2, Δ_3, L)

Algorithm 19, Restore, iterates through all locations in Δ at the current level of nesting acc, storing the current data for each location into the *then* portion of the mapping for the given location, and restoring the original data to the location from the *orig* portion. Additionally, it will update the tag j to 1 for all locations. This allows Resolve to know whether a new location was added during the execution of the `else` branch, and to use the value stored in *orig* when such a location is found.

Algorithm 20 $(V, n_{res}, L) \leftarrow$ Resolve_Retrieve$(\gamma, \sigma,\ \Delta,\ \text{acc})$

$V = [\,]$
$(l_{res}, \text{private int}) = \gamma(res\_acc)$
$(\omega_{res}, \text{private int}, 1, \text{PermL}(\text{Freeable}, \text{private int}, \text{private}, 1)) = \sigma(l_{res})$
$n_{res} = \text{DecodeVal}(\text{private int}, \omega_{res})$
$L = [(l_{res}, 0)]$
for all $((l, \mu) \to (v_{orig}, v_{then}, j, ty)) \in \Delta[\text{acc}]$ **do**
  $v_t = \text{NULL}$
  if $j = 0$ **then**
    $v_t = v_{orig}$
  else
    $v_t = v_{then}$
  end if
  $v_e = \text{NULL}$
  if $(\mu = 0)$ **then**
    $(\omega, ty, \alpha, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma(l)$
    if $(ty = \text{private } bty)$ **then**
      $(\omega, \text{private } bty, 1, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, 1)) = \sigma(l)$
      $v_e = \text{DecodeVal}(\text{private } bty, \omega)$
    else if $(ty = \text{private } bty*)$ **then**
      $(\omega, \text{private } bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha)) = \sigma(l)$
      $v_e = \text{DecodePtr}(\text{private } bty*, \alpha, \omega)$
    end if
  else
    $v_e = \text{GetBytes}((l, \mu), ty, \sigma)$
  end if
  $V = V.append(v_t, v_e)$
  $L = L.append((l, \mu))$
end for
return (V, n_{res}, L)

Algorithm 20, Resolve_Retrieve, returns the result of the conditional, a list of tuples of the `then` and `else` values for each location in $\Delta[\text{acc}]$, and a list of locations that it has accessed. To get values for each branch, it iterates through all the locations in $\Delta[\text{acc}]$. To get the `then` value, it uses the tag indicating whether that location was modified in the `then` branch or not; if it is 0, it will use the stored original value from before execution of either branch, if it is 1, it will use the stored then value. The data currently stored in each location is used for the `else` data, as execution of the `else` branch has just completed.

Algorithm 21, Resolve_Store, stores all the true values back into memory. It iterates through all the locations in $\Delta[\text{acc}]$, encoding the values as their expected type and writing this byte representation into memory.

2.2.3 Multiparty resolution. Algorithm 22, ($\text{MPC}_{resolve}$), is the multiparty algorithm for facilitating the secure resolution of the values of which branch are the true values.

We have already read the elements from memory, so each tuple within the parties value list V^{p} is either a pointer data structure or an int (or float) value. We proceed to find the true value based upon what type of value we are currently viewing, leveraging Algorithm 23 to compute the final pointer data structure for each pointer.

Algorithm 23 (CondAssign) is an Algorithm that requires multiparty computation. Due to the complexity of this Algorithm, that it is always called by each party within a different multiparty algorithm, and that it directly calls specific multiparty protocols that give the behavior for a single party, we show the behavior as it would occur at a single party. CondAssign takes two pointer data

Algorithm 21 $(\sigma_1, \Delta_1, L) \leftarrow \text{Resolve\_Store}(\Delta,\ \sigma,\ \text{acc},\ V)$

$L = [\,]$
$\sigma_1 = \sigma$
for all $i \in \{0 \ldots |V| - 1\}$ **do**
  $v_f = V[i]$
  $((l, \mu) \rightarrow (v_{orig}, v_{then}, j, ty)) = \Delta[\text{acc}][i]$
  if $(\mu = 0)$ **then**
    if $(ty = \text{private } bty)$ **then**
      $(\omega, ty, \alpha, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, \alpha)) = \sigma_1(l)$
      if $(\alpha = 1)$ **then**
        $\sigma_1 = \text{UpdateVal}(\sigma,\ l,\ v_f,\ ty)$
      else
        $\sigma_1 = \text{UpdateArr}(\sigma,\ (l, 0),\ v_f,\ ty)$
      end if
    else if $(ty = \text{private } bty*)$ **then**
      $\sigma_1 = \text{UpdatePtr}(\sigma,\ (l, 0),\ v_f,\ ty)$
    end if
  else
    $\sigma_1 = \text{SetBytes}((l, \mu), ty, v_f, \sigma)$
  end if
  $L = L.append((l, \mu))$
  $\sigma = \sigma_1$
end for
$\Delta_1 = \Delta.pop()$
return $(\sigma_1, \Delta_1, s, L)$

Algorithm 22 $(V_f^1, \ldots, V_f^{\mathrm{q}}) \leftarrow \text{MPC}_{resolve}([n_{res}^1, \ldots, n_{res}^{\mathrm{q}}], [V^1, \ldots, V^{\mathrm{q}}])$

for all $\mathrm{p} \in \{1 \ldots \mathrm{q}\}$ **do**
  $V_f^{\mathrm{p}} = [\,]$
  for all $i \in \{0 \ldots |V^{\mathrm{p}}| - 1\}$ **do**
    $(v_t^{\mathrm{p}}, v_e^{\mathrm{p}}) = V^{\mathrm{p}}[i]$
    $v_f^{\mathrm{p}} = \text{NULL}$
    if $([\alpha_t, L_t, J_t, i] = v_t^{\mathrm{p}})$ **then**
      $[\alpha_e^{\mathrm{p}}, L_e^{\mathrm{p}}, J_e^{\mathrm{p}}, i] = v_e^{\mathrm{p}}$
      $[\alpha_f^{\mathrm{p}}, L_f^{\mathrm{p}}, J_f^{\mathrm{p}}] = \text{CondAssign}([\alpha_t^{\mathrm{p}}, L_t^{\mathrm{p}}, J_t^{\mathrm{p}}], [\alpha_e^{\mathrm{p}}, L_e^{\mathrm{p}}, J_e^{\mathrm{p}}], n_{res}^{\mathrm{p}})$
      $v_f^{\mathrm{p}} = [\alpha_f^{\mathrm{p}}, L_f^{\mathrm{p}}, J_f^{\mathrm{p}}, i]$
    else
      $v_f^{\mathrm{p}} = \text{MPC}_{add}(\text{MPC}_{mult}(n_{res}^{\mathrm{p}}, v_t^{\mathrm{p}}), \text{MPC}_{mult}(\text{MPC}_{sub}(1, n_{res}^{\mathrm{p}}), v_e^{\mathrm{p}}))$
    end if
    $V_f^{\mathrm{p}}.append(v_f^{\mathrm{p}})$
  end for
end for
return $(V_f^1, \ldots, V_f^{\mathrm{q}})$

structures with the associated number of locations, lists of locations, and lists of tags as well as a flag n_{res}. Its primary purpose is to merge two pointer data structures during the execution of conditional statements with private conditions. Here, n_{res} is a flag that indicates whether the true pointer location should be taken from the first or the second data structure; $n_{res} == 1$ means that the true location is in the first data structure. For example, when executing code `if (priv) p1 = p2;`, n_{res} is the result of evaluating private condition `priv`, the first data structure corresponds to `p1`'s data structure prior to executing this statement, and the second data structure corresponds to

Algorithm 23 $[\alpha_3,\ L_3,\ J_3] \leftarrow \mathrm{CondAssign}([\alpha_1,\ L_1,\ J_1],\ [\alpha_2,\ L_2,\ J_2],\ n_{res})$

$L_3 = L_1 \cup L_2$
$\alpha_3 = |L_3|$
$J_3 = [\ ]$
for all $(l_m, \mu_m) \in L_3$ **do**
    $pos_1 = L_1.\mathrm{find}((l_m, \mu_m))$
    $pos_2 = L_2.\mathrm{find}((l_m, \mu_m))$
    if $(pos_1 \wedge pos_2)$ **then**
        $j''_m = \mathrm{MPC}_{add}(\mathrm{MPC}_{mult}(n_{res}, j'_{pos_2}), \mathrm{MPC}_{mult}(\mathrm{MPC}_{sub}(1, n_{res}), j_{pos_1}))$
    else if $(\neg\ pos_2)$ **then**
        $j''_m = \mathrm{MPC}_{mult}(\mathrm{MPC}_{sub}(n_{res}), j_{pos_1})$
    else
        $j''_m = \mathrm{MPC}_{mult}(n_{res}, j'_{pos_2})$
    end if
    $J_3.\mathrm{append}(j''_m)$
end for
return $[\alpha_3,\ L_3,\ J_3]$

p2's data structure. The function first computes the union of the two lists of locations and then updates their corresponding tags based on their tags at the time of calling this function and the value of n_{res}. For example, if a particular location l_m is found on both lists, we retain its tag from the first list if n_{res} is set and otherwise retain its tag from the second list if n_{res} is not set. When l_m is found only in one of the lists, we use a similar logic and conditionally retain its original tag based on the value of n_{res}. If a tag is not retained, it is reset to 0. This ensures that for any pointer data structure only one tag is set to 1 and all others are set to 0.

2.3 Freeing locations

Algorithm 24 corresponds to conventional memory deallocation when we call free to deallocate memory associated with some pointer. We simply set the permissions for this location to be None, indicating that it has been freed and is no longer intended to be in use.

Algorithm 24 $\sigma_3 \leftarrow \mathrm{Free}(\sigma_1,\ l)$

$\sigma_2[l \rightarrow (\omega,\ a\ bty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, a\ bty, a, 1))] = \sigma_1$
$\sigma_3 = \sigma_2\big[l \rightarrow (\omega,\ a\ bty,\ 1,\ \mathrm{PermL}(\mathrm{None}, a\ bty, a, 1))\big]$
return $(\sigma_3, (l, 0))$

CheckFreeable is depicted as Algorithm 25 and ensures the behavior expected of free: if the location was properly allocated via a call to malloc, it is deallocatable for the purpose of this function. In particular, the default location $l_{default}$ that corresponds to uninitialized pointers is not deallocatable (and freeing such a pointer has no effect); similarly memory associated with statically declared variables is not de-allocatable via this mechanism (and freeing it here also has no effect). Thus, if CheckFreeable returns 1, we will proceed to mark location l as unavailable within the rules this is called from, otherwise the freeing rules have no effect on the state of memory.

Algorithm 26, MPC_{free}, corresponds to deallocating memory associated with a pointer to private data which may be associated with multiple locations where the data may actually reside. The true location is not publicly known and the location to be removed should be chosen based on public knowledge. For the purposes of this functionality, and without loss of generality, we deallocate the first location on the list, l_0. Deallocation of l_0 requires additional work because that location might not be the true location, and may still be validly in use by other pointers. In other words, based on

Algorithm 25 $j \leftarrow \text{CheckFreeable}(\gamma, L, J, \sigma)$

```
1: if (l_default, 0) ∈ L then
2:     return 0
3: end if
4: for all (l_m, μ_m) ∈ L do
5:     if μ_m ≠ 0 then
6:         return 0
7:     end if
8: end for
9: if 1 ∉ J then
10:    return 0
11: end if
12: for all x ∈ γ do
13:    (l_x, ty_x) = γ(x)
14:    if (l_x, 0) ∈ L then
15:        return 0
16:    else if ty_x = a const bty* then
17:        (ω, ty_x, 1, PermL(Freeable, ty_x, a, 1)) = σ(l_x)
18:        [1, [(l_1, 0)], [1], 1] = DecodePtr(ty_x, 1, ω)
19:        if (l_1, 0) ∈ L then
20:            return 0
21:        end if
22:    end if
23: end for
24: return 1
```

Algorithm 26 $([[\omega_0'^1, ..., \omega_n'^1], ..., [\omega_0'^q, ..., \omega_n'^q]], [J'^1, ..., J'^q]) \leftarrow \text{MPC}_{free}([[\omega_0^1, ..., \omega_n^1], ..., [\omega_0^q, ..., \omega_n^q]], [J^1, ... J^q])$

```
1: for all p ∈ {1...q} do
2:     ω_0'^p = ω_0^p
3:     [j_0^p, ..., j_n^p] = J^p
4:     j_0'^p = j_0^p
5:     for all m ∈ {1...n} do
6:         ω_m'^p = MPC_add(MPC_mult(ω_m^p, MPC_sub(1, j_m^p)), MPC_mult(ω_0^p, j_m^p))
7:         ω_0'^p = MPC_add(MPC_mult(ω_0'^p, MPC_sub(1, j_m^p)), MPC_mult(ω_m^p, j_m^p))
8:         j_0'^p = MPC_add(j_0'^p, j_m^p)
9:     end for
10:    J'^p = [j_0'^p, j_1^p, ..., j_n^p]
11: end for
12: return ([[ω_0'^1, ..., ω_n'^1], ..., [ω_0'^q, ..., ω_n'^q]], [J'^1, ..., J'^q])
```

the fact that freeing a pointer has been called, we know that the true location can be released, but it might not be safe to deallocate other locations associated with the pointer.

For that reason, in Algorithm 26 we iterate through all locations l_1 through $l_{\alpha-1}$ and swap the content of the current location l_m and l_0 if l_m is in fact the true location (i.e., flag j_m is set). That is, ω'_m corresponds to the updated content of location l_m: the content will remain unchanged if j_m is not set, and otherwise, it will be replaced with the content of location l_0. Similarly, ω'_0 corresponds to the updated content of location l_0. Note that it may be modified in at most one iteration of the loop, namely, when j_m is set. All other iterations will keep the value unchanged (and it will never be modified if none of the tags $j_1, \ldots, j_{\alpha-1}$ are set and j_0 is). The function is written to be data-oblivious, i.e., to not reveal the true location associated with the pointer. We additionally compute an update to the tag for l_0, ensuring that if it was swapped with another location, we will have two tags set to 1 to indicate the two locations whose data we swapped. This algorithm then returns the updated set of bytes and tag list with the updated first tag.

Algorithm 27 $(\sigma_1, L_1) \leftarrow \text{UpdatePointerLocations}(\sigma, L, J, (l_r, \mu_r), j_r)$

$\sigma_1 = [\,]$
$L_1 = [\,]$
for all $l_k \in \sigma$ **do**
  $(\omega_k,\ ty,\ n,\ \text{PermL}(\text{Freeable},\ ty,\ a,\ n)) = \sigma(l_k)$
  if $(ty = \text{private } bty*)$ **then**
    $L_1 = L_1 :: [(l_k, 0)]$
    $[n,\ L_k,\ J_k,\ i] = \text{DecodePtr}(\text{private } bty*,\ n,\ \omega)$
    if $(l_r, \mu_r) \in L_k$ **then**
      $pos = L_k.\text{find}((l_r, \mu_r))$
      $J'_k = J_k \setminus J_k[pos]$
      $L'_k = L_k \setminus (l_r, \mu_r)$
      $[\alpha_{new}, L_{new}, J_{new}] = \text{CondAssign}([|L|,\ L,\ J],\ [n-1,\ L'_k,\ J'_k],\ J_k[pos])$
      $\omega'_k = \text{EncodePtr}(\text{private } bty*,\ [\alpha_{new}, L_{new}, J_{new}, i])$
      $\sigma_1 = \sigma_1[l_k \rightarrow (\omega'_k,\ ty,\ n,\ \text{PermL}(\text{Freeable},\ ty,\ a,\ n))]$
    else
      $\sigma_1 = \sigma_1[l_k \rightarrow (\omega_k,\ ty,\ n,\ \text{PermL}(\text{Freeable},\ ty,\ a,\ n))]$
    end if
  else
    $\sigma_1 = \sigma_1[l_k \rightarrow (\omega_k,\ ty,\ n,\ \text{PermL}(\text{Freeable},\ ty,\ a,\ n))]$
  end if
end for
return (σ_1, L_1)

In UpdatePointerLocations, we are given location l_r which is being removed and a list of other locations L associated with the pointer in question. In the event that l_r was not the true pointer location, its content has been moved to another location, but it still may remain in the lists of other pointers, which is what this function is to correct. In particular, the function iterates through other pointers in the system and searches for location l_r in their lists. If l_r is present (i.e., $l_r \in L_k$), we need to remove it and replace it with another location from L to which the data has been moved. However, because we do not know which location in L is set and contains the relevant data, we are left with merging all locations in L with the pointer's current locations L'_k after removing l_r. This is done using Algorithm 23, CondAssign.

Notice that we are also merging two pointer data structures based on a condition. This time the condition is $j_{k_{pos}}$, which indicates whether the true location is in the first or second list of locations. That is, if l_r was the true location of the pointer, the data has been moved and resides in one of the locations in L. Otherwise, if l_r was not the true location, the data resides at one of the remaining locations associated with the pointer on its location list L'_k. Thus, we merge the list of locations and update the corresponding tags in the same way this is done during evaluation of conditional statements with private conditions.

Algorithm 28, UpdateBytesFree, if the final step of the rule for `pfree` when the pointer has multiple locations. Here, we are modifying the permissions of the first location l_0 to be None, indicating that this location has been freed, and storing the updated set of bytes for this location into memory. We then iterate through all other locations in the list, storing their modified byte representations into memory. Once this is complete, we will have completed the swap of data if l_0 was not the true location. Otherwise, we are simply writing the original data into memory again.

Algorithm 28 $\sigma_2 \leftarrow$ UpdateBytesFree$(\sigma,\ [(l_0, 0), ..., (l_n, 0)],\ [\omega_0, ..., \omega_n])$

```
σ1[l0 → (ω'0, ty, α, PermL(Freeable, ty, private, α))] = σ
σ2 = σ1[l0 → (ω0, ty, α, PermL(Freeable, ty, private, α))]
for all m ∈ {1...n} do
    σ3[lm → (ω'm, ty, αm, PermL(Freeable, ty, private, αm))] = σ2
    σ4 = σ3[lm → (ωm, ty, αm, PermL(Freeable, ty, private, αm))]
    σ2 = σ4
end for
return σ2
```

2.4 Evaluation code and locations-touched tracking

Algorithm 29 defines how new memory block identifiers are obtained - each party will have a counter that is monotonically increasing after each time ϕ is called, and a *temp* counter that is monotonically decreasing after each time $\phi(temp)$ is called. The *temp* argument is optional, and it signifies when the *temp* counter is to be used – that is, only during the allocation of temporary variables used within the Private If Else rules. We separate these elements into their own partition of memory in order to easily show correctness of the memory with regards to Vanilla C- it is possible to provide a more robust mapping scheme between locations in Vanilla C and locations in SMC^2, but this extension provides unnecessary complexity for our proofs.

Algorithm 29 $l \leftarrow \phi^{\mathrm{p}}(\{temp\})$

```
next = l_default
if temp then
    next = p_global_location_temp_counter − −
else
    next = p_global_location_counter ++
end if
return l_next
```

Algorithm 30 illustrates how two locations-touched data structures are merged. This merging maintains the ordering of which locations were touched and how many times for each party.

Algorithm 30 $\mathcal{L}_3 \leftarrow \mathcal{L}_1 :: \mathcal{L}_2$

```
ℒ3 = ℒ1
for all (p, L) ∈ ℒ2 do
    if (ℒ3 = (p, L1) || ℒ4) then
        ℒ3 = (p, L1 :: L) || ℒ4
    else
        ℒ3 = (p, L) || ℒ3
    end if
end for
return ℒ3
```

Algorithm 31 illustrates how two evaluation code data structures are merged. This merging maintains the ordering of when the evaluation was completed in respect to other evaluations completed by each party (i.e., a local party evaluation ordering, not a total ordering). When a multiparty evaluation code is entered (i.e., $\mathcal{D}_2 == (\text{ALL}, d)$), we iterate through and add the code to the evaluation code lists of each of the parties.

Algorithm 32 illustrates the filtering of the evaluation codes executed by a single party from the overall data structure showing the evaluation codes executed by each party, respectively.

Algorithm 31 $\mathcal{D}_3 \leftarrow \mathcal{D}_1 :: \mathcal{D}_2$

```
1: if ((ALL, [d]) == 𝒟2) then
2:     𝒟3 = ε
3:     for all (p, D) ∈ 𝒟1 do
4:         𝒟3 = (p, D :: d) || 𝒟3
5:     end for
6: else
7:     𝒟3 = 𝒟1
8:     for all (p, D) ∈ 𝒟2 do
9:         if (𝒟3 = (p, D1) || 𝒟4) then
10:            𝒟3 = (p, D1 :: D) || 𝒟4
11:        else
12:            𝒟3 = (p, D) || 𝒟3
13:        end if
14:    end for
15: end if
16: return 𝒟3
```

Algorithm 32 $\mathcal{D}^{\mathrm{p}} \leftarrow \mathcal{D}(\mathrm{p})$

```
1: 𝒟^p = ε
2: if (((p, D) || 𝒟1) == 𝒟) then
3:     𝒟^p = (p, D)
4: end if
5: return 𝒟^p
```

Algorithm 33 illustrates the filtering of the locations touched in the memory of a single party from the overall data structure showing the locations touched by each party in their respective memories.

Algorithm 33 $\mathcal{L}^{\mathrm{p}} \leftarrow \mathcal{L}(\mathrm{p})$

```
1: ℒ^p = ε
2: if (((p, L) || ℒ1) == ℒ) then
3:     ℒ^p = (p, L)
4: end if
5: return ℒ^p
```

2.5 Expression label judgement (public vs. private)

Algorithm 34 illustrates how $(E) \vdash \gamma$ is evaluated, finding if there is at least one private element in the expression list E. As we iterate through each expression in E, if we find an expression that is private, $(E) \vdash \gamma$ holds as true. Otherwise, if we have evaluated all expressions and found none are private, we return false. In this case, as we show in Algorithm 35, $(E) \nvdash \gamma$ holds as true, because all elements are public.

Algorithm 34 $j \leftarrow (E) \vdash \gamma$

```
for all e ∈ E do
    if (e == x(E)) then
        (l, tyL → ty) = γ(x)
        if ((ty == private bty∗) ∨ (ty == private bty)) then
            return 1
        end if
    else if ((e == uop var) ∨ (e == var)) then
        if (var == x) then
            (l, ty) = γ(x)
            if ((ty == private bty∗) ∨ (ty == private bty)) then
                return 1
            end if
        else if (var == x[e1]) then
            (l, ty) = γ(x)
            if (ty == private bty∗) then
                return 1
            else if (e1) ⊢ γ then
                return 1
            end if
        end if
    else if (e == e1 bop e2) then
        if ((e1, e2) ⊢ γ) then
            return 1
        end if
    else if (e == (e1)) then
        if (e1) ⊢ γ then
            return 1
        end if
    else if (e == (ty) e1) then
        if ((ty == private bty) ∨ (ty == private bty∗) then
            return 1
        else if (e1) ⊢ γ then
            return 1
        end if
    else if (e == v) then
        if (e == [v0, ..., vn]) then
            if (v0, ..., vn) ⊢ γ then
                return 1
            end if
        else if (e == encrypt(n)) then
            return 1
        end if
    end if
end for
return 0
```

Algorithm 35 $j \leftarrow (E) \nvdash \gamma$

```
if ((E) ⊢ γ) then
    return 0
else
    return 1
end if
```

2.6 Helpers for Functions

Algorithm 36 $tyL \leftarrow \text{GetFunTypeList}(P)$

```
tyL = [ ]
while P ≠ void do
    if P == ty then
        tyL = ty :: tyL
        P = void
    else if P == P′, ty then
        tyL = ty :: tyL
        P = P′
    end if
end while
return tyL
```

Algorithm 37 $s \leftarrow \text{GetFunParamAssign}(P, E)$

```
Require: length(P) = length(E)
s = skip
while P ≠ void do
    if (P == ty var) ∧ (E == e) then
        s = ty var = e; s
        P = void
        E = void
    else if (P == P′, ty) ∧ (E == E′, e) then
        s = ty var = e; s
        P = P′
        E = E′
    end if
end while
return s
```

2.7 Pointer Helper Algorithms

Algorithm 38 $(L_2,\ \eta_{final}) \longleftarrow \text{IncrementList}(L_1,\ n,\ \sigma)$

```
L_2 = [ ]
η_final = 1
for all (l, μ) ∈ L_1 do
    if l == l_default then
        L_2.append((l_default, 0))
    else
        ((l_1, μ_1), η) = GetLocation(l, μ), n, σ)
        η_final = η ∧ η_final
        L_2.append((l_1, μ_1))
    end if
end for
return (L_2, η_final)
```

Algorithm 39 $(\sigma_2,\ \eta) \longleftarrow \text{UpdateOffset}(\sigma,\ (l, \mu),\ v,\ a\ bty)$

```
if μ == 0 then
    σ_2 = UpdateVal(σ, l, v, a bty)
    return (σ_2, 1)
end if
ω = EncodeVal(a bty, v)
σ_1[l → (ω_1, a_1 bty_1, n, PermL(Freeable, a_1 bty_1, a_1, n))] = σ
if (a bty == a_1 bty_1) ∧ (μ < n) then
    ω_2 = UpdateBytes(ω, ω_1, μ)
    σ_2 = σ_1[l → (ω_2, a_1 bty_1, n, PermL(Freeable, a_1 bty_1, a_1, n))]
    η = 1
else
    σ_2 = UpdateOvershooting(σ, (l, μ), ω, a bty)
    η = 0
end if
return (σ_2, η)
```

2.8 Updating memory

In this subsection, we present the algorithms used to update memory within the semantics. The following algorithms are for regular (int or float) values, array values, and pointer values, respectively, when updating these values in memory – for regular values and pointers, at offset 0, and for arrays at an offset within the bounds of the array. The algorithms to assist with other pointer and array updates are located in their corresponding subsections.

Algorithm 40 $\sigma_2 \longleftarrow \text{UpdateVal}(\sigma,\ l,\ v,\ a\ bty)$

```
ω_2 = EncodeVal(a bty, v)
σ_1[l → (ω_1, ty, 1, PermL(Freeable, ty, a, 1))] = σ
σ_2 = σ_1[l → (ω_2, ty, 1, PermL(Freeable, ty, a, 1))]
return σ_2
```

Algorithm 40 (UpdateVal) is used to update regular (int or float) values in memory. It takes as input memory σ, the memory block identifier of the location we will be updating, the value to store into memory, and the type to store it as. UpdateVal first encodes the value as the specified type, then removes the original mapping from memory and inserts the new mapping with the updated byte data. It then returns the updated memory.

Algorithm 41 $\sigma_2 \leftarrow \text{UpdateArr}(\sigma,\ (l, i),\ v,\ a\ bty)$

1: $\sigma_1[l \rightarrow (\omega,\ ty,\ \alpha,\ \text{PermL}(\text{Freeable}, ty, a, \alpha))] = \sigma$
2: $\mu = i \cdot \text{sizeof}(a\ bty)$
3: $\omega_1 = \omega[0 : \mu]$
4: $\omega_2 = \text{EncodeVal}(a\ bty,\ v)$
5: $\omega_3 = \omega[\mu + \mu :]$
6: $\omega_4 = \omega_1 :: \omega_2 :: \omega_3$
7: $\sigma_2 = \sigma_1[l \rightarrow (\omega_4,\ ty,\ \alpha,\ \text{PermL}(\text{Freeable}, ty, a, \alpha))]$
8: **return** σ_2

Algorithm 41 (UpdateArr) is used to update a value in memory at an index within an array. It takes as input memory σ, the location (memory block identifier and offset) and we will be updating, the value to store into memory, and the type to store the value as. Here, we first remove the mapping from memory (line 1), then find where the offset we will be updating will be within the array data (line 2). Next, we separate out the bytes before (line 3) and after (line 5) the data we will be replacing. We encode the new value based on the specified type (line 4), then combine these byte data to obtain the updated array byte data (line 6). We then place the new mapping with the updated data into memory (line 7) and return the updated memory. Here, we would like to highlight that we only update the portion of memory associated with the given offset (array index), which is public.

Algorithm 42 $(\sigma_2,\ \eta) \leftarrow \text{UpdatePtr}(\sigma,\ (l, \mu),\ [\alpha,\ L,\ J,\ i],\ a\ bty*)$

$\omega = \text{EncodePtr}(a\ bty*,\ [\alpha,\ L,\ J,\ i])$
$\sigma_1[l \rightarrow (\omega_1,\ ty_1,\ \alpha_1,\ \text{PermL}(\text{Freeable}, ty, a_1, \alpha_1))] = \sigma$
if $(\mu == 0) \wedge (a\ bty* = ty_1)$ **then**
  $\sigma_2 = \sigma_1[l \rightarrow (\omega,\ ty_1,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable}, ty_1, a_1, \alpha))]$
  $\eta = 1$
else
  $\sigma_2 = \text{UpdateOvershooting}(\sigma,\ (l, \mu),\ \omega,\ a\ bty*)$
  $\eta = 0$
end if
return (σ_2, η)

Algorithm 42 (UpdatePtr) is used to update the pointer data structure for a pointer. It takes as input memory σ, the location (memory block identifier and offset) and we will be updating, the value to store into memory, and the type to store the value as. It then returns the updated memory.

2.9 Encoding and Decoding

In this subsection, we present the algorithms used for encoding and decoding bytes in memory in our semantics. First, it is important to note that we leave the specifics of encoding to bytes and decoding from bytes up to the implementation, as this low-level function may vary based on the system and underlying architecture.

Algorithm 43 $\omega \leftarrow \text{EncodeVal}(ty, v)$

Algorithm 44 $v \leftarrow \text{DecodeVal}(ty, \omega)$

Algorithm 43, EncodeVal, takes as input a type and a value. It encodes the given value of the given type as bytes of data, and returns those bytes.

Algorithm 44, DecodeVal, takes as input a type and bytes of data. It interprets the given bytes of data as a value of the given type, and returns that value.

Algorithm 45 $\omega \leftarrow \text{EncodeArr}(ty, \alpha, v)$

$\omega_v = \text{EncodeVal}(ty, v)$
$\omega = \omega_v$
for all $i \in \{1 \ldots \alpha - 1\}$ **do**
  $\omega = \omega + \omega_v$
end for
return ω

Algorithm 46 $v \leftarrow \text{DecodeArr}(ty, i, \omega)$

$\mu = i \cdot \text{sizeof}(ty)$
$\omega_1 = \omega[\mu : \mu + \mu]$
$v = \text{DecodeVal}(ty, \omega_1)$
return v

Algorithm 45 (EncodeArr) takes an value and creates byte data for an array of length α, with every element initialized to the value. It is currently only used in the semantics when declaring an array, to initialize the newly declared array as being filled with NULL elements. EncodeArr takes as input the type, number of elements, and the value to be used to initialize the array. It will encode the given value as byte data based on the type, and duplicate that α times to get the byte data for the entire array initialized with that value. This full byte data is then returned.

Algorithm 46 (DecodeArr) takes byte data and returns the element of the given type at the specified index from the byte data. It takes as input a type, an index, and bytes of data for an array. It then finds the portion of bytes corresponding to that index, and calls Algorithm DecodeVal to obtain the value represented by those bytes. This value is then returned.

Algorithm 47 $\omega \leftarrow \text{EncodeFun}(s, n, P)$

Algorithm 48 $(s, n, P) \leftarrow \text{DecodeFun}(\omega)$

Algorithm 47 (EncodeFun) takes the function data and encodes it into its byte representation. It takes as input a statement (body of the function), the tag for whether it contains public side effects, and the function's parameter list. EncodeFun then encodes this information into byte data and returns the byte data.

Algorithm 48 (DecodeFun) takes the byte representation of a function and decodes it into the function's information: the statement (body of the function), the tag for whether it contains public side effects, and the parameter list. It takes as input the byte data and then returns the function's information.

Algorithm 49 $\omega \leftarrow \text{EncodePtr}(ty, [\alpha, L, J, i])$

Algorithm 50 $[\alpha, L, J, i] \leftarrow \text{DecodePtr}(ty, \alpha, \omega)$

Algorithm 49 (EncodePtr) takes a pointer data structure and encodes it into byte data. It takes a pointer type, number, and byte data as input. It then encodes the pointer data structure containing the number α indicating the number of locations, a list of α locations L, a list of α tags, and a number indicating the level of indirection of the pointer into byte data. This byte data is then returned.

Algorithm 50 (DecodePtr) does the opposite of EncodePtr, taking byte data and retrieving the pointer data structure from it. It takes a pointer type, number, and byte data as input. It then interprets the given set of bytes as a pointer data structure containing the number α indicating the number of locations, a list of α locations L, a list of α tags, and a number indicating the level of indirection of the pointer. This pointer data structure is then returned.

3 VANILLA C SEMANTICS IN SMC[2] STYLE

Multiparty Binary Operation

$$\frac{\begin{array}{c}((1,\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1))\Downarrow'_{\hat{\mathcal{D}}_1}((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1))\\ ((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_2}((1,\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2))\\ \hat n_1\, bop\ \hat n_2=\hat n_3 \qquad bop\in\{\cdot,+,-,\div\}\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\square,\square,\hat e_1\, bop\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\square,\square,\hat e_1\, bop\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{ALL},[\hat{mpb}])}\\ ((1,\hat\gamma,\hat\sigma_2,\square,\square,\hat n_3)\qquad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_2,\square,\square,\hat n_3))\end{array}}$$

Multiparty Comparison True Operation

$$\frac{\begin{array}{c}((1,\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1))\Downarrow'_{\hat{\mathcal{D}}_1}((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1))\\ ((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_2}((1,\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2))\\ (\hat n_1\, bop\ \hat n_2)=1 \qquad bop\in\{==,\ !=,\ <\}\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\ \square,\square,\hat e_1\, bop\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\ \square,\square,\hat e_1\, bop\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{ALL},[\hat{mpcmpt}])}\\ ((1,\hat\gamma,\hat\sigma_2,\square,\square,1)\qquad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_2,\square,\square,1))\end{array}}$$

Multiparty Comparison False Operation

$$\frac{\begin{array}{c}((1,\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma,\ \square,\square,\ \hat e_1))\Downarrow'_{\hat{\mathcal{D}}_1}((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat n_1))\\ ((1,\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_1,\ \square,\square,\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_2}((1,\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\ \hat\sigma_2,\ \square,\square,\ \hat n_2))\\ (\hat n_1\, bop\ \hat n_2)=0 \qquad bop\in\{==,\ !=,\ <\}\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\ \square,\square,\hat e_1\, bop\ \hat e_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\ \square,\square,\hat e_1\, bop\ \hat e_2))\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{ALL},[\hat{mpcmpf}])}\\ ((1,\hat\gamma,\hat\sigma_2,\square,\square,0)\qquad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_2,\square,\square,0))\end{array}}$$

Multiparty Pre-Increment Variable

$$\frac{\begin{array}{lll}\hat\gamma(\hat x)=(\hat l,\hat{bty}) & \hat\sigma(\hat l)=(\hat\omega,\hat{bty},1,\mathrm{PermL}(\mathrm{Freeable},\hat{bty},\mathrm{public},1)) & \\ \mathrm{DecodeVal}(\hat{bty},\hat\omega)=\hat n_1 & \hat n_2=\hat n_1+1 & \mathrm{UpdateVal}(\hat\sigma,\hat l,\hat n_2,\hat{bty})=\hat\sigma_1\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\ \square,\square,{+}{+}\,\hat x)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\ \square,\square,{+}{+}\,\hat x))\Downarrow'_{(\mathrm{ALL},[\hat{mppin}])}\\ ((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat n_2)\quad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat n_2))\end{array}}$$

Multiparty If Else False

$$\frac{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\square,\square,\hat e)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\square,\square,\hat e))\Downarrow'_{\hat{\mathcal{D}}_1}((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat n)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat n))\qquad \hat n=0\\ ((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat s_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat s_1))\Downarrow'_{\hat{\mathcal{D}}_2}((1,\hat\gamma_1,\hat\sigma_2,\square,\square,\mathrm{skip})\parallel\ldots\parallel(\mathrm{q},\hat\gamma_1,\hat\sigma_2,\square,\square,\mathrm{skip}))\\ ((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat s_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat s_2))\Downarrow'_{\hat{\mathcal{D}}_3}((1,\hat\gamma_2,\hat\sigma_3,\square,\square,\mathrm{skip})\parallel\ldots\parallel(\mathrm{q},\hat\gamma_2,\hat\sigma_3,\square,\square,\mathrm{skip}))\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\ \square,\square,\mathrm{if}(\hat e)\ \hat s_1\ \mathrm{else}\ \hat s_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\ \square,\square,\mathrm{if}(\hat e)\ \hat s_1\ \mathrm{else}\ \hat s_2))\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::\hat{\mathcal{D}}_3::(\mathrm{ALL},[\hat{mpief}])}\\ ((1,\hat\gamma,\hat\sigma_3,\square,\square,\mathrm{skip})\qquad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_3,\square,\square,\mathrm{skip}))\end{array}}$$

Multiparty If Else True

$$\frac{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\square,\square,\hat e)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\square,\square,\hat e))\Downarrow'_{\hat{\mathcal{D}}_1}((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat n)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat n))\qquad \hat n\neq 0\\ ((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat s_1)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat s_1))\Downarrow'_{\hat{\mathcal{D}}_2}((1,\hat\gamma_1,\hat\sigma_2,\square,\square,\mathrm{skip})\parallel\ldots\parallel(\mathrm{q},\hat\gamma_1,\hat\sigma_2,\square,\square,\mathrm{skip}))\\ ((1,\hat\gamma,\hat\sigma_1,\square,\square,\hat s_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_1,\square,\square,\hat s_2))\Downarrow'_{\hat{\mathcal{D}}_3}((1,\hat\gamma_2,\hat\sigma_3,\square,\square,\mathrm{skip})\parallel\ldots\parallel(\mathrm{q},\hat\gamma_2,\hat\sigma_3,\square,\square,\mathrm{skip}))\end{array}}{\begin{array}{l}((1,\hat\gamma,\hat\sigma,\ \square,\square,\mathrm{if}(\hat e)\ \hat s_1\ \mathrm{else}\ \hat s_2)\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma,\ \square,\square,\mathrm{if}(\hat e)\ \hat s_1\ \mathrm{else}\ \hat s_2))\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::\hat{\mathcal{D}}_3::(\mathrm{ALL},[\hat{mpiet}])}\\ ((1,\hat\gamma,\hat\sigma_2,\square,\square,\mathrm{skip})\qquad\parallel\ldots\parallel(\mathrm{q},\hat\gamma,\hat\sigma_2,\square,\square,\mathrm{skip}))\end{array}}$$

Fig. 30. Selected Vanilla C multiparty semantics.

Multiparty Pointer Dereference Write Value

$$\frac{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n})) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*) \qquad \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1)) \\ \mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1] \qquad \mathrm{UpdateOffset}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, 1)\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})) \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{ALL}, [\hat{mpwdp}])} \\ ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}))\end{array}}$$

Multiparty Pointer Dereference Write Value Higher Level Indirection

$$\frac{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e))) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*) \qquad \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1)) \\ \mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}] \\ \hat{i} > 1 \qquad \mathrm{UpdatePtr}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i} - 1], \hat{bty}*) = (\hat{\sigma}_2, 1)\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})) \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{ALL}, [\hat{mpwdp1}])} \\ ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}))\end{array}}$$

Multiparty Pointer Dereference

$$\frac{\begin{array}{l}\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*) \qquad \hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1)) \\ \mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1] \qquad \mathrm{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x})) \Downarrow'_{(\mathrm{ALL}, [\hat{mprdp}])} \\ ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}))\end{array}}$$

Multiparty Pointer Dereference Higher Level Indirection

$$\frac{\begin{array}{l}\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*) \qquad \hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1)) \qquad \hat{i} > 1 \\ \mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}] \qquad \mathrm{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i} - 1], 1)\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x})) \Downarrow'_{(\mathrm{ALL}, [\hat{mprdp1}])} \\ ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)))\end{array}}$$

Multiparty Free

$$\frac{\begin{array}{l}\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*) \qquad \sigma(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1)) \\ \mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], \hat{i}] \qquad \mathrm{CheckFreeable}(\hat{\gamma}, [(\hat{l}_1, 0)], [1], \hat{\sigma}) = 1 \qquad \mathrm{Free}(\hat{\sigma}, \hat{l}_1) = \hat{\sigma}_1\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{free}(\hat{x})) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{free}(\hat{x}))) \Downarrow'_{(\mathrm{ALL}, [\hat{mpfre}])} \\ ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}))\end{array}}$$

Multiparty Array Read

$$\frac{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i})) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*) \qquad \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1)) \\ 0 \leq \hat{i} \leq \hat{\alpha} - 1 \qquad \mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] \\ \hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha})) \qquad \mathrm{DecodeArr}(\hat{bty}, \hat{i}, \hat{\omega}_1) = \hat{n}_{\hat{i}}\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}])) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{ALL}, [\hat{mpra}])} \\ ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}))\end{array}}$$

Multiparty Array Write

$$\frac{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1)) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i})) \\ ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n})) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*) \qquad \hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1)) \\ \mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] \qquad 0 \leq \hat{i} \leq \hat{\alpha} - 1 \\ \hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha})) \qquad \mathrm{UpdateArr}(\hat{\sigma}_2, (\hat{l}_1, \hat{i}), \hat{n}, \hat{bty}) = \hat{\sigma}_3\end{array}}{\begin{array}{l}((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{ALL}, [\hat{mpwa}])} \\ ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}))\end{array}}$$

Fig. 31. Multiparty Vanilla C semantic rules for pointers and arrays.

Equal To False

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 = \hat{n}_2) = 0\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\square,\square,\ \hat{e}_1 == \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{eqf}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\square,\square,0) \parallel \hat{C}_2)}$$

Equal To True

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 = \hat{n}_2) = 1\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\square,\square,\ \hat{e}_1 == \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{eqt}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\square,\square,1) \parallel \hat{C}_2)}$$

Not Equal To True

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 \neq \hat{n}_2) = 1\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\square,\square,\ \hat{e}_1! = \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{net}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\square,\square,\ 1) \parallel \hat{C}_2)}$$

Not Equal To False

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 \neq \hat{n}_2) = 0\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\square,\square,\ \hat{e}_1! = \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{nef}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\square,\square,0) \parallel \hat{C}_2)}$$

Less Than False

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 < \hat{n}_2) = 0\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\square,\square,\ \hat{e}_1 < \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{ltf}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\square,\square,0) \parallel \hat{C}_2)}$$

Less Than True

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\square,\square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \parallel \hat{C}_2) \qquad (\hat{n}_1 < \hat{n}_2) = 1\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1 < \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{ltt}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ 1) \parallel \hat{C}_2)}$$

Fig. 32. Vanilla C semantic rules for comparison operations within the scope of the grammar shown in Figure 1.

Subtraction

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{n}_1) \parallel \hat{C}_1) \quad ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1, \square, \square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_2) \parallel \hat{C}_2) \quad \hat{n}_1 - \hat{n}_2 = \hat{n}_3}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}, \square, \square,\ \hat{e}_1 - \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bs}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2, \square, \square,\ \hat{n}_3) \parallel \hat{C}_2)}$$

Addition

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{n}_1) \parallel \hat{C}_1) \quad ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1, \square, \square,\ \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_2) \parallel \hat{C}_2) \quad \hat{n}_1 + \hat{n}_2 = \hat{n}_3}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1 + \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bp}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_3) \parallel \hat{C}_2)}$$

Multiplication

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{n}_1) \parallel \hat{C}_1) \quad ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1, \square,\ \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_2) \parallel \hat{C}_2) \quad \hat{n}_1 \cdot \hat{n}_2 = \hat{n}_3}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1 \cdot \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bm}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_3) \parallel \hat{C}_2)}$$

Division

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{n}_1) \parallel \hat{C}_1) \quad ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1, \square,\ \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_2) \parallel \hat{C}_2) \quad \hat{n}_1 \div \hat{n}_2 = \hat{n}_3}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1 \div \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bd}])} ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_3) \parallel \hat{C}_2)}$$

Fig. 33. Vanilla C semantic rules for binary operations within the scope of the grammar shown in Figure 1.

Declaration Assignment

$$\frac{\begin{array}{c}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{ty}\ \hat{x})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{x} = \hat{e})\ \parallel \hat{C}_1)\ \Downarrow'_{\hat{\mathcal{D}}_2}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_2)\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{ty}\ \hat{x} = \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{ds}])}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_2)}$$

Address Of

$$\frac{\hat{\gamma}(\hat{x}) = (\hat{l},\ \hat{ty})}{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \&\hat{x})\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{loc}])} \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ (\hat{l},\ 0))\ \parallel \hat{C})\end{array}}$$

Write

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{n})\ \parallel \hat{C}_1) \\ \hat{\gamma}(\hat{x}) = (\hat{l},\ \hat{bty}) \qquad \mathrm{UpdateVal}(\hat{\sigma}_1,\ \hat{l},\ \hat{n},\ \hat{bty}) = \hat{\sigma}_2\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x} = \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\hat{w}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_1)}$$

Size of type

$$\frac{\hat{n} = \tau(\hat{ty})}{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \mathrm{sizeof}(\hat{ty}))\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{ty}])} \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{n})\ \parallel \hat{C})\end{array}}$$

Declaration

$$\frac{\begin{array}{ll}\hat{l} = \phi() & \hat{\omega} = \mathrm{EncodeVal}(\hat{bty},\ \mathrm{NULL}) \\ \hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l},\ \hat{bty})] & \hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega},\ \hat{bty},\ 1,\ \mathrm{PermL}(\mathrm{Freeable},\ \hat{bty},\ \mathrm{public},\ 1))]\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{bty}\ \hat{x})\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{dv}])}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C})}$$

Statement Sequencing

$$\frac{\begin{array}{c}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{s}_1)\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{v}_1)\ \parallel \hat{C}_1) \\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{s}_2)\ \parallel \hat{C}_1)\ \Downarrow'_{\hat{\mathcal{D}}_2}\ ((\mathrm{p},\ \hat{\gamma}_2,\ \hat{\sigma}_2,\ \square,\ \square,\ \hat{v}_2)\ \parallel \hat{C}_2)\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{s}_1;\hat{s}_2)\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{ss}])}\ ((\mathrm{p},\ \hat{\gamma}_2,\ \hat{\sigma}_2,\ \square,\ \square,\ \hat{v}_2)\ \parallel \hat{C}_2)}$$

Parentheses

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{v})\ \parallel \hat{C}_1)}{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ (\hat{e}))\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p},[\hat{ep}])} \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{v})\ \parallel \hat{C}_1)\end{array}}$$

Statement Block

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{s})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_1)}{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \{\hat{s}\})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p},[\hat{sb}])} \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_1)\end{array}}$$

Read

$$\frac{\begin{array}{l}\hat{\gamma}(\hat{x}) = (\hat{l},\ \hat{bty}) \\ \hat{\sigma}(\hat{l}) = (\hat{\omega},\ \hat{bty},\ 1,\ \mathrm{PermL}(\mathrm{Freeable},\ \hat{bty},\ \mathrm{public},\ 1)) \\ \mathrm{DecodeVal}(\hat{bty},\ \hat{\omega}) = \hat{n}\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x})\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{r}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{n})\ \parallel \hat{C})}$$

Pointer Read Location

$$\frac{\begin{array}{l}\hat{\gamma}(\hat{x}) = (\hat{l},\ \hat{bty}*) \\ \hat{\sigma}(\hat{l}) = (\hat{\omega},\ \hat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \hat{bty}*,\ \mathrm{public},\ 1)) \\ \mathrm{DecodePtr}(\hat{bty}*,\ 1,\ \hat{\omega}) = [1,\ [(\hat{l}_1,\ \hat{\mu}_1)],\ [1],\ \hat{i}]\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x})\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{rp}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ (\hat{l}_1,\ \hat{\mu}_1))\ \parallel \hat{C})}$$

Pre-Increment Variable

$$\frac{\begin{array}{lll}\hat{\gamma}(\hat{x}) = (\hat{l},\ \hat{bty}) & \hat{\sigma}(\hat{l}) = (\hat{\omega},\ \hat{bty},\ 1,\ \mathrm{PermL}(\mathrm{Freeable},\ \hat{bty},\ \mathrm{public},\ 1)) & \\ \mathrm{DecodeVal}(\hat{bty},\ \hat{\omega}) = \hat{n}_1 & \hat{n}_2 = \hat{n}_1 + 1 & \mathrm{UpdateVal}(\hat{\sigma},\ \hat{l},\ \hat{n}_2,\ \hat{bty}) = \hat{\sigma}_1\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ {+}{+}\ \hat{x})\ \parallel \hat{C})\ \Downarrow'_{(\mathrm{p},[\hat{pin}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{n}_2)\ \parallel \hat{C})}$$

While End

$$\frac{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{n})\ \parallel \hat{C}_1) \qquad \hat{n} = 0}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \mathrm{while}(\hat{e})\ \hat{s})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p},[\hat{wle}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_1)}$$

While Continue

$$\frac{\begin{array}{l}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{n})\ \parallel \hat{C}_1) \qquad \hat{n} \neq 0 \\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{s})\ \parallel \hat{C}_1)\ \Downarrow'_{\hat{\mathcal{D}}_2}\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \parallel \hat{C}_2)\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \mathrm{while}(\hat{e})\ \hat{s})\ \parallel \hat{C})\ \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{wlc}])}\ ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{while}(\hat{e})\ \hat{s})\ \parallel \hat{C}_2)}$$

Fig. 34. Some Vanilla C semantic rules within the scope of the grammar shown in Figure 1.

If Else False

$$\frac{\begin{array}{ll}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{n}) \parallel \hat{C}_1) & \hat{n}=0 \\ ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{s}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2) & \end{array}}{((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\widehat{ief}])} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2)}$$

If Else True

$$\frac{\begin{array}{ll}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{n}) \parallel \hat{C}_1) & \hat{n}\neq 0 \\ ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{s}_1) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2) & \end{array}}{((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\widehat{iet}])} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2)}$$

Function Call

$$\frac{\begin{array}{lll}\hat{\gamma}(\hat{x}) = (\hat{l},\ \widehat{tyL} \to \widehat{ty}) & \hat{\sigma}(\hat{l}) = (\hat{\omega},\ \widehat{tyL} \to \widehat{ty},\ 1,\ \mathrm{PermL\_Fun(public)}) & \mathrm{DecodeFun}(\hat{\omega}) = (\hat{s},\square,\hat{P}) \\ \mathrm{GetFunParamAssign}(\hat{P},\hat{E}) = \hat{s}_1 & ((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\hat{s}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_1,\square,\square,\mathrm{skip}) \parallel \hat{C}_1) & \\ & ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_1,\square,\square,\hat{s}) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\hat{\gamma}_2,\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2) & \end{array}}{((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\hat{x}(\hat{E})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\widehat{fc}])} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C}_2)}$$

Pre-Declared Function Definition

$$\frac{\begin{array}{ll}\hat{x} \in \hat{\gamma} & \hat{\gamma}(\hat{x}) = (\hat{l},\ \widehat{tyL} \to \widehat{ty}) \\ & \mathrm{EncodeFun}(\hat{s},\square,\hat{P}) = \hat{\omega} \\ \multicolumn{2}{l}{\hat{\sigma} = \hat{\sigma}_1[\hat{l} \to (\mathrm{NULL},\ \widehat{tyL} \to \widehat{ty},\ 1,\ \mathrm{PermL\_Fun(public)})]} \\ \multicolumn{2}{l}{\hat{\sigma}_2 = \hat{\sigma}_1[\hat{l} \to (\hat{\omega},\ \widehat{tyL} \to \widehat{ty},\ 1,\ \mathrm{PermL\_Fun(public)})]}\end{array}}{\begin{array}{l}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\widehat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\widehat{fpd}])} \\ ((\mathrm{p},\hat{\gamma},\hat{\sigma}_2,\square,\square,\mathrm{skip}) \parallel \hat{C})\end{array}}$$

Function Definition

$$\frac{\begin{array}{ll}\hat{x} \notin \hat{\gamma} & \mathrm{GetFunTypeList}(\hat{P}) = \widehat{tyL} \\ \hat{l} = \phi() & \hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l},\ \widehat{tyL} \to \widehat{ty})] \\ & \mathrm{EncodeFun}(\hat{s},\square,\hat{P}) = \hat{\omega} \\ \multicolumn{2}{l}{\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega},\ \widehat{tyL} \to \widehat{ty},\ 1,\ \mathrm{PermL\_Fun(public)})]}\end{array}}{\begin{array}{l}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\widehat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\widehat{fd}])} \\ ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_1,\square,\square,\mathrm{skip}) \parallel \hat{C})\end{array}}$$

Function Declaration

$$\frac{\begin{array}{ll}\hat{l} = \phi() & \mathrm{GetFunTypeList}(\hat{P}) = \widehat{tyL} \\ & \hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l},\ \widehat{tyL} \to \widehat{ty})] \\ \multicolumn{2}{l}{\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\mathrm{NULL},\ \widehat{tyL} \to \widehat{ty},\ 1,\ \mathrm{PermL\_Fun(public)})]}\end{array}}{\begin{array}{l}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\widehat{ty}\ \hat{x}(\hat{P})) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\widehat{df}])} \\ ((\mathrm{p},\hat{\gamma}_1,\hat{\sigma}_1,\square,\square,\mathrm{skip}) \parallel \hat{C})\end{array}}$$

Cast Value

$$\frac{\begin{array}{l}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,\hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{n}) \parallel \hat{C}_1) \\ \hat{n}_1 = \mathrm{Cast}(\mathrm{public},\ \widehat{ty},\ \hat{n})\end{array}}{\begin{array}{l}((\mathrm{p},\hat{\gamma},\hat{\sigma},\square,\square,(\widehat{ty})\ \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\widehat{cv}])} \\ ((\mathrm{p},\hat{\gamma},\hat{\sigma}_1,\square,\square,\hat{n}_1) \parallel \hat{C}_1)\end{array}}$$

Fig. 35. Vanilla C semantic rules for branching, functions, and casting values.

Input Value

$$\frac{\begin{array}{lll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}) & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) \\ \mathrm{InputValue}(\hat{x}, \hat{n}) = \hat{n}_1 & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{x} = \hat{n}_1) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcinput}(\hat{x}, \hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\widehat{inp}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)}$$

Output Value

$$\frac{\begin{array}{lll}((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) & \hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}) & \\ \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \widehat{bty}, 1, \mathrm{PermL}(\mathrm{Freeable}, \widehat{bty}, \mathrm{public}, 1)) & \mathrm{DecodeVal}(\widehat{bty}, \hat{\omega}) = \hat{n}_1 & \mathrm{OutputValue}(\hat{x}, \hat{n}, \hat{n}_1)\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcoutput}(\hat{x}, \hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{out}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)}$$

Input Array

$$\frac{\begin{array}{ll}((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) & \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \widehat{bty}*) \\ ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{\alpha}) \parallel \hat{C}_2) & \mathrm{InputArray}(\hat{x}, \hat{n}, \hat{\alpha}) = [\hat{m}_0, \ldots, \hat{m}_{\hat{\alpha}}] \\ ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{x} = [\hat{m}_0, \ldots, \hat{m}_{\hat{\alpha}}]) \parallel \hat{C}_2) \Downarrow'_{\hat{\mathcal{D}}_3} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_3) & \end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcinput}(\hat{x}, \hat{e}_1, \hat{e}_2)) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\widehat{inp1}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_3)}$$

Output Array

$$\frac{\begin{array}{ll}((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) & \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \widehat{bty}*) \\ ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{\alpha}) \parallel \hat{C}_2) & \\ \multicolumn{2}{l}{\hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \widehat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \widehat{bty}*, \mathrm{public}, 1))} \\ \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] & \hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \widehat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \widehat{bty}, \mathrm{public}, \hat{\alpha})) \\ \forall i \in \{0, \ldots, \hat{\alpha} - 1\} \quad \mathrm{DecodeArr}(\widehat{bty}, i, \hat{\omega}_1) = \hat{m}_i & \mathrm{OutputArray}(\hat{x}, \hat{n}, \hat{\alpha}) = [\hat{m}_0, \ldots, \hat{m}_{\hat{\alpha}-1}]\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcoutput}(\hat{x}, \hat{e}_1, \hat{e}_2)) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\widehat{out1}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)}$$

Free

$$\frac{\begin{array}{lll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \multicolumn{2}{l}{\hat{\sigma}(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \widehat{bty}*, \mathrm{public}, 1))} \\ \mathrm{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] & \mathrm{CheckFreeable}(\hat{\gamma}, [(\hat{l}_1, 0)], [1], \hat{\sigma}) = 1 & \mathrm{Free}(\hat{\sigma}, \hat{l}_1) = \hat{\sigma}_1\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{free}(\hat{x})) \parallel \hat{C}) \Downarrow'_{(\mathrm{p}, [\widehat{fre}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C})}$$

Malloc

$$\frac{\begin{array}{ll} & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) \\ \hat{l} = \phi() & \hat{\sigma}_2 = \hat{\sigma}_1\big[\hat{l} \to \big(\mathrm{NULL}, \mathrm{void}*, \hat{n}, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{public}, \hat{n})\big)\big]\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{malloc}(\hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{mal}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)}$$

Cast Location

$$\frac{\begin{array}{ll}(\hat{ty} = \widehat{bty}*) & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1) \\ & \hat{\sigma}_1 = \hat{\sigma}_2\big[\hat{l} \to \big(\hat{\omega}, \mathrm{void}*, \hat{n}, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{public}, \hat{n})\big)\big] \\ & \hat{\sigma}_3 = \hat{\sigma}_2\left[\hat{l} \to \left(\hat{\omega}, \hat{ty}, \frac{\hat{n}}{\tau(\hat{ty})}, \mathrm{PermL}\left(\mathrm{Freeable}, \hat{ty}, \mathrm{public}, \frac{\hat{n}}{\tau(\hat{ty})}\right)\right)\right]\end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{cl}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)}$$

Fig. 36. Vanilla C semantic rules for input and output, memory allocation and deallocation, and casting locations.

Pointer Declaration

$$\frac{\begin{array}{llll}(\hat{ty} = \widehat{bty}*) & \text{GetIndirection}(*) = \hat{i} & \hat{l} = \phi() & \hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l}, \hat{ty})] \\ \multicolumn{2}{l}{\hat{\omega} = \text{EncodePtr}(\hat{ty}*, [1, [(\hat{l}_{default}, 0)], [1], \hat{i}])} & \multicolumn{2}{l}{\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega}, \hat{ty}, 0, \text{PermL\_Ptr}(\text{Freeable}, \hat{ty}, \text{public}, 0))]}\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\hat{dp}])} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C})}$$

Pointer Write Location

$$\frac{\begin{array}{ll} & ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}] & \text{UpdatePtr}(\hat{\sigma}_1, (\hat{l}, 0), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}], \widehat{bty}*) = (\hat{\sigma}_2, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{wp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)}$$

Pre-Increment Pointer

$$\frac{\begin{array}{ll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{\sigma}(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ & \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1] \\ ((\hat{l}_2, \hat{\mu}_2), 1) = \text{GetLocation}((\hat{l}_1, \hat{\mu}_1), \tau(\widehat{bty}), \hat{\sigma}) & \text{UpdatePtr}(\hat{\sigma}, (\hat{l}, 0), [1, [(\hat{l}_2, \hat{\mu}_2)], [1], 1], \widehat{bty}*) = (\hat{\sigma}_1, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, {+}{+}\ \hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\widehat{pin1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})}$$

Pre-Increment Pointer Higher Level Indirection

$$\frac{\begin{array}{ll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{\sigma}(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \hat{i} > 1 & \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}] \\ ((\hat{l}_2, \hat{\mu}_2), 1) = \text{GetLocation}((\hat{l}_1, \hat{\mu}_1), \tau(\widehat{bty}*), \hat{\sigma}) & \text{UpdatePtr}(\hat{\sigma}, (\hat{l}, 0), [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}], \widehat{bty}*) = (\hat{\sigma}_1, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, {+}{+}\ \hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\widehat{pin2}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})}$$

Pointer Dereference Write Value

$$\frac{\begin{array}{ll} & ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1] & \text{UpdateOffset}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), \hat{n}, \widehat{bty}) = (\hat{\sigma}_2, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\widehat{wdp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)}$$

Pointer Dereference Write Higher Level Indirection

$$\frac{\begin{array}{ll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1) \\ \hat{i} > 1 & \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}] & \text{UpdatePtr}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i} - 1], \widehat{bty}*) = (\hat{\sigma}_2, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\widehat{wdp1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)}$$

Pointer Dereference

$$\frac{\begin{array}{ll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{\sigma}(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1] & \text{DerefPtr}(\hat{\sigma}, \widehat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\widehat{rdp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})}$$

Pointer Dereference Higher Level Indirection

$$\frac{\begin{array}{lll}\hat{\gamma}(\hat{x}) = (\hat{l}, \widehat{bty}*) & \hat{i} > 1 & \hat{\sigma}(\hat{l}) = (\hat{\omega}, \widehat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \widehat{bty}*, \text{public}, 1)) \\ \multicolumn{2}{l}{\text{DecodePtr}(\widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]} & \text{DerefPtrHLI}(\hat{\sigma}, \widehat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i} - 1], 1)\end{array}}{((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\widehat{rdp1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})}$$

Fig. 37. Additional Vanilla C semantic rules for pointers.

Array Declaration Assignment

$$\frac{\begin{array}{c}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{ty}\ \hat{x}[\hat{e}])\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1}((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_1)\\ ((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{x}=\hat{e})\ \|\ \hat{C}_1)\Downarrow'_{\hat{\mathcal{D}}_2}((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_2)\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{ty}\ \hat{x}[\hat{e}]=\hat{e})\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\widehat{das}])}((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_2,\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_2)}$$

Read Entire Array

$$\frac{\begin{array}{ll}\hat{\gamma}(\hat{x})=(\hat{l},\ \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}(\hat{l})=(\hat{\omega},\ \mathrm{const}\ \widehat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{const}\ \widehat{bty}*,\ \mathrm{public},\ 1))\\ & \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*,\ 1,\ \hat{\omega})=[1,\ [(\hat{l}_1,\ 0)],\ [1],\ 1]\\ \multicolumn{2}{l}{\hat{\sigma}(\hat{l}_1)=(\hat{\omega}_1,\ \widehat{bty},\ \hat{\alpha},\ \mathrm{PermL}(\mathrm{Freeable},\ bty,\ \mathrm{public},\ \hat{\alpha}))\quad \forall\hat{i}\in\{0\ldots\hat{\alpha}-1\}.\quad \mathrm{DecodeArr}(\widehat{bty},\ \hat{i},\ \hat{\omega}_1)=\hat{n}_{\hat{i}}}\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x})\ \|\ \hat{C})\Downarrow'_{(\mathrm{p},[\widehat{rea}])}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ [\hat{n}_0,\ \ldots,\ \hat{n}_{\hat{\alpha}-1}])\ \|\ \hat{C})}$$

Write Entire Array

$$\frac{\begin{array}{ll} & ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ [\hat{n}_0,\ \ldots,\ \hat{n}_{\hat{\alpha}_e-1}])\ \|\ \hat{C}_1)\\ \hat{\gamma}(\hat{x})=(\hat{l},\ \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}_1(\hat{l})=(\hat{\omega},\ \mathrm{const}\ \widehat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{const}\ \widehat{bty}*,\ \mathrm{public},\ 1))\\ \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*,\ 1,\ \hat{\omega})=[1,\ [(\hat{l}_1,\ 0)],\ [1],\ 1] & \hat{\sigma}_1(\hat{l}_1)=(\hat{\omega}_1,\ \widehat{bty},\ \hat{\alpha},\ \mathrm{PermL}(\mathrm{Freeable},\ bty,\ \mathrm{public},\ \hat{\alpha}))\\ \hat{\alpha}_e=\hat{\alpha} & \forall\hat{i}\in\{0\ldots\hat{\alpha}-1\}\quad \mathrm{UpdateArr}(\hat{\sigma}_{1+\hat{i}},\ (\hat{l}_1,\ \hat{i}),\ \hat{n}_{\hat{i}},\ \widehat{bty})=\sigma_{2+\hat{i}}\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x}=\hat{e})\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p},[\widehat{wea}])}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_{2+\hat{\alpha}-1},\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_1)}$$

Array Declaration

$$\frac{\begin{array}{ll}\hat{l}=\phi()\qquad \hat{l}_1=\phi() & ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{\alpha})\ \|\ \hat{C}_1)\\ \hat{\alpha}>0 & \hat{\omega}=\mathrm{EncodePtr}(\mathrm{const}\ \widehat{bty}*,\ [1,\ [(\hat{l}_1,\ 0)],\ [1],\ 1])\\ \hat{\gamma}_1=\hat{\gamma}[\hat{x}\to(\hat{l},\ \mathrm{const}\ \widehat{bty}*)] & \hat{\sigma}_2=\hat{\sigma}_1[\hat{l}\to(\hat{\omega},\ \mathrm{const}\ \widehat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{const}\ \widehat{bty}*,\ \mathrm{public},\ 1))]\\ \hat{\omega}_1=\mathrm{EncodeArr}(\widehat{bty},\ 0,\ \hat{\alpha},\ \mathrm{NULL}) & \hat{\sigma}_3=\hat{\sigma}_2[\hat{l}_1\to(\hat{\omega}_1,\ \widehat{bty},\ \hat{\alpha},\ \mathrm{PermL}(\mathrm{Freeable},\ \widehat{bty},\ \mathrm{public},\ \hat{\alpha}))]\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \widehat{bty}\ \hat{x}[\hat{e}])\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\widehat{da}])}((\mathrm{p},\ \hat{\gamma}_1,\ \hat{\sigma}_3,\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_1)}$$

Array Read

$$\frac{\begin{array}{ll} & ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e})\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{i})\ \|\ \hat{C}_1)\\ \hat{\gamma}(\hat{x})=(\hat{l},\ \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}_1(\hat{l})=(\hat{\omega},\ \mathrm{const}\ \widehat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{const}\ \widehat{bty}*,\ \mathrm{public},\ 1))\\ & \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*,\ 1,\ \hat{\omega})=[1,\ [(\hat{l}_1,\ 0)],\ [1],\ 1]\\ & \hat{\sigma}_1(\hat{l}_1)=(\hat{\omega}_1,\ \widehat{bty},\ \hat{\alpha},\ \mathrm{PermL}(\mathrm{Freeable},\ \widehat{bty},\ \mathrm{public},\ \hat{\alpha}))\\ 0\le\hat{i}\le\hat{\alpha}-1 & \mathrm{DecodeArr}(\widehat{bty},\ \hat{i},\ \hat{\omega}_1)=\hat{n}_{\hat{i}}\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x}[\hat{e}])\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\widehat{ra}])}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{n}_{\hat{i}})\ \|\ \hat{C}_1)}$$

Array Write

$$\frac{\begin{array}{ll} & ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{e}_1)\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{i})\ \|\ \hat{C}_1)\\ & ((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_1,\ \square,\ \square,\ \hat{e}_2)\ \|\ \hat{C}_1)\Downarrow'_{\hat{\mathcal{D}}_2}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_2,\ \square,\ \square,\ \hat{n})\ \|\ \hat{C}_2)\\ \hat{\gamma}(\hat{x})=(\hat{l},\ \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}_2(\hat{l})=(\hat{\omega},\ \mathrm{const}\ \widehat{bty}*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable},\ \mathrm{const}\ \widehat{bty}*,\ \mathrm{public},\ 1))\\ & \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*,\ 1,\ \hat{\omega})=[1,\ [(\hat{l}_1,\ 0)],\ [1],\ 1]\\ & \hat{\sigma}_2(\hat{l}_1)=(\hat{\omega}_1,\ \widehat{bty},\ \hat{\alpha},\ \mathrm{PermL}(\mathrm{Freeable},\ \widehat{bty},\ \mathrm{public},\ \hat{\alpha}))\\ 0\le\hat{i}\le\hat{\alpha}-1 & \mathrm{UpdateArr}(\hat{\sigma}_2,\ (\hat{l}_1,\ \hat{i}),\ \hat{n},\ \widehat{bty})=\hat{\sigma}_3\end{array}}{((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma},\ \square,\ \square,\ \hat{x}[\hat{e}_1]=\hat{e}_2)\ \|\ \hat{C})\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\widehat{wa}])}((\mathrm{p},\ \hat{\gamma},\ \hat{\sigma}_3,\ \square,\ \square,\ \mathrm{skip})\ \|\ \hat{C}_2)}$$

Fig. 38. Vanilla C semantic rules for arrays.

Array Read Out of Bounds

$$\frac{\begin{array}{ll} & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \widehat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \widehat{bty}*, \mathrm{public}, 1)) \\ & \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] \\ & \hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \widehat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \widehat{bty}, \mathrm{public}, \hat{\alpha})) \\ (\hat{i} < 0) \vee (\hat{i} \geq \hat{\alpha}) & \mathrm{ReadOOB}(\hat{i}, \hat{\alpha}, \hat{l}_1, \widehat{bty}, \hat{\sigma}_1) = (\hat{n}, 1) \end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{rao}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)}$$

Array Write Out of Bounds

$$\frac{\begin{array}{ll} & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1) \\ & ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \hat{C}_2) \\ \hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \widehat{bty}*) & \hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \widehat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \widehat{bty}*, \mathrm{public}, 1)) \\ & \mathrm{DecodePtr}(\mathrm{const}\ \widehat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1] \\ & \hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \widehat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \widehat{bty}, \mathrm{public}, \hat{\alpha})) \\ (\hat{i} < 0) \vee (\hat{i} \geq \hat{\alpha}) & \mathrm{WriteOOB}(\hat{n}, \hat{i}, \hat{\alpha}, \hat{l}_1, \widehat{bty}, \hat{\sigma}_2) = (\hat{\sigma}_3, 1) \end{array}}{((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\widehat{wao}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)}$$

Fig. 39. Vanilla C semantic rules for array out of bounds.

4 CORRECTNESS

In our semantics, we give each evaluation an identifying code as a shorthand way to refer to that specific evaluation, as well as to allow us to quickly reason about the Vanilla C and SMC^2 evaluations that are congruent to each other (i.e., a Vanilla C rule and an identical one handling only public data in SMC^2).

The list of Vanilla C codes are as follows: $VanC$ = [*mpb*, *mpcmpt*, *mpcmpf*, *mppin*, *mpra*, *mpwe*, *mpfre*, *mpiet*, *mpief*, *mprdp*, *mprdp1*, *mpwdp*, *mpwdp1*, *fls*, *ss*, *sb*, *ep*, *cv*, *cl*, *r*, *w*, *ds*, *dv*, *dp*, *da*, *wle*, *wlc*, *bp*, *bs*, *bm*, *bd*, *ltf*, *ltt*, *eqf*, *eqt*, *nef*, *net*, *mal*, *fre*, *wp*, *wdp*, *wdp1*, *rp*, *rdp*, *rdp1*, *ra*, *wa*, *rao*, *wao*, *rae*, *wae*, *loc*, *iet*, *ief*, *inp*, *inp1*, *out*, *out1*, *df*, *ty*, *fd*, *fpd*, *fc*, *pin*, *pin1*, *pin2*].

The list of SMC^2 codes are as follows: $SmcC$ = [*mpb*, *mpcmp*, *mpra*, *mpwa*, *mppin*, *mpdp*, *mpdph*, *mpfre*, *mprdp*, *mprdp1*, *mpwdp*, *mpwdp1*, *mpwdp2*, *mpwdp3*, *iet*, *ief*, *iep*, *iepd*, *wle*, *wlc*, *dp*, *dp1*, *rp*, *rp1*, *rdp*, *rdp1*, *rdp2*, *wp*, *wp1*, *wp2*, *wdp*, *wdp1*, *wdp2*, *wdp3*, *wdp4*, *wdp5*, *da*, *da1*, *das*, *ra*, *ra1*, *rea*, *wa*, *wa1*, *wa2*, *wea*, *wea1*, *wea2*, *rao*, *rao1*, *wao*, *wao1*, *wao2*, *pin*, *pin1*, *pin2*, *pin3*, *pin4*, *pin5*, *pin6*, *pin7*, *mal*, *malp*, *fre*, *pfre*, *cv*, *cv1*, *cl*, *cl1*, *loc*, *ty*, *df*, *fd*, *fpd*, *fc*, *fc1*, *bp*, *bs*, *bm*, *bd*, *ltf*, *ltt*, *eqf*, *eqt*, *nef*, *net*, *dv*, *d1*, *r*, *r1*, *w*, *w1*, *w2*, *ds*, *ss*, *sb*, *ep*, *inp*, *inp1*, *inp2*, *inp3*, *out*, *out1*, *out2*, *out*].

The list of Vanilla C codes that would lead to differences with a SMC^2 evaluation are as follows: $VanCX = [rao'*, wao'*, pin2'*, pin3'*]$ The list of SMC^2 codes that would lead to differences with a Vanilla C evaluation are as follows: $SmcCX = [rao*, rao1*, wao*, wao1*, wao2*, pin2*, pin3*, pin4*, pin5*, pin6*, pin7*]$. In all of these rules, where the algorithms return the tag 1 to indicate the access is well-aligned, the * versions of the rules would return 0. With these rules, it is not possible to prove correctness, as they would return garbage values that no longer are congruent between SMC^2 and Vanilla C. We can prove all of these rules to maintain noninterference - each case is similar to the corresponding non-* version, and therefore does not add anything of interest to the proof, so we omit these cases from this document.

SMC C	Vanilla C Equivalent Cases	
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpcmp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[\hat{mpcmpt}])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[\hat{mpcmpf}])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[iep])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{mpiet}])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{mpief}])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[iepd])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{mpiet}])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{mpief}])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[malp])}$	$\Downarrow'_{\hat{\mathcal{D}}::[(\text{p},[\hat{ty}]),(\text{p},[\hat{bm}]),(\text{p},[\hat{mal}])]}$	

Fig. 40. Table of more complex SMC^2 evaluation codes and their congruent Vanilla C evaluation codes.

4.1 Correctness: Erasure Function

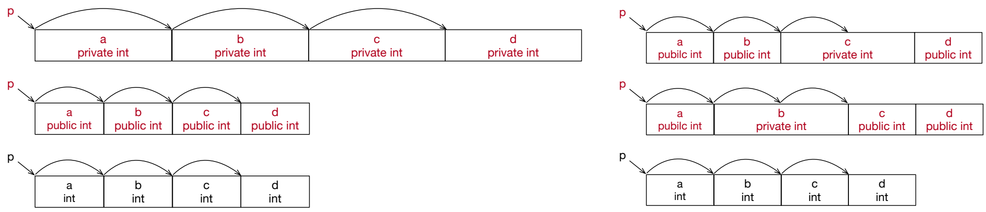
Here, we show the full erasure function in Figure 43. This function is intended to take a SMC^2 program or configuration and remove all private privacy labels, decrypt any private data, and clear any additional tracking features that are specific to SMC^2; this process will result in a Vanilla C program or configuration.

Figure 43b shows erasure over an entire configuration, calling Erase on the four-tuple of the environment, memory, and two empty maps needed as the base for the Vanilla C environment and memory; removing the accumulator (i.e., replacing it with □); and calling Erase on the statement. Figure 43c shows erasure over types and type lists (i.e., for function types). Here, we remove any privacy labels given to the types, with unlabeled types being returned as is. For function types, we must iterate over the entire list of types as well as the return type. Figure 43d shows erasure over expression lists (i.e., from function calls) and parameter lists (i.e., from function definitions).

$SmcC$	$VanC$	$SmcC$	$VanC$	$SmcC$	$VanC$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpra])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[m\hat{p}ra])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[fc1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{fc}])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[fc])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{fc}])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpwa])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[m\hat{p}wa])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[fpd])}$	$\Downarrow'_{(\text{p},[\hat{f}pd])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[df])}$	$\Downarrow'_{(\text{p},[\hat{d}f])}$
$\Downarrow^{\mathcal{L}}_{(\text{ALL},[mprdp])}$	$\Downarrow'_{(\text{ALL},[mp\hat{r}dp])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin])}$	$\Downarrow'_{(\text{p},[p\hat{i}n])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[fd])}$	$\Downarrow'_{(\text{p},[\hat{f}d])}$
$\Downarrow^{\mathcal{L}}_{(\text{ALL},[mprdp1])}$	$\Downarrow'_{(\text{ALL},[mp\hat{r}dp1])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin3])}$	$\Downarrow'_{(\text{p},[p\hat{i}n])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[cl1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{c}l])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpwdp2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[mp\hat{w}dp1])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin1])}$	$\Downarrow'_{(\text{p},[p\hat{i}n1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[cl])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{c}l])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpwdp1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[mp\hat{w}dp1])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin2])}$	$\Downarrow'_{(\text{p},[p\hat{i}n2])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[cv1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{c}v])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpwdp3])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[mp\hat{w}dp])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin4])}$	$\Downarrow'_{(\text{p},[p\hat{i}n1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[cv])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{c}v])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpwdp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[mp\hat{w}dp])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin5])}$	$\Downarrow'_{(\text{p},[p\hat{i}n2])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ltt])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[l\hat{t}t])}$
$\Downarrow^{\mathcal{L}}_{(\text{ALL},[mppin])}$	$\Downarrow'_{(\text{ALL},[m\hat{p}pin])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin6])}$	$\Downarrow'_{(\text{p},[p\hat{i}n1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ltf])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[l\hat{t}f])}$
$\Downarrow^{\mathcal{L}}_{(\text{ALL},[mpfre])}$	$\Downarrow'_{(\text{ALL},[mp\hat{f}re])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pin7])}$	$\Downarrow'_{(\text{p},[p\hat{i}n2])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[eqt])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[e\hat{q}t])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{ALL},[mpb])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{ALL},[m\hat{p}b])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[eqf])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[e\hat{q}f])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[sb])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{s}b])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[net])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[n\hat{e}t])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[d1])}$	$\Downarrow'_{(\text{p},[\hat{d}v])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[nef])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[n\hat{e}f])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[dv])}$	$\Downarrow'_{(\text{p},[\hat{d}v])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[fre])}$	$\Downarrow'_{(\text{p},[\hat{f}re])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[pfre])}$	$\Downarrow'_{(\text{p},[\hat{f}re])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp3])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ief])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{e}f])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[iet])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{e}t])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp4])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wle])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}le])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wlc])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}lc])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wdp4])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ss])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[s\hat{s}])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ds])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{d}s])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[w])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[w1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[w2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}])}$
$\Downarrow^{\mathcal{L}}_{(\text{p},[dp1])}$	$\Downarrow'_{(\text{p},[\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[dp])}$	$\Downarrow'_{(\text{p},[\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[rp])}$	$\Downarrow'_{(\text{p},[\hat{r}p])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[mal])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[m\hat{a}l])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[rp1])}$	$\Downarrow'_{(\text{p},[\hat{r}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wp1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}p])}$
$\Downarrow^{\mathcal{L}}_{(\text{p},[r])}$	$\Downarrow'_{(\text{p},[\hat{r}])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[r1])}$	$\Downarrow'_{(\text{p},[\hat{r}])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}p])}$
$\Downarrow^{\mathcal{L}}_{(\text{p},[ty])}$	$\Downarrow'_{(\text{p},[\hat{t}y])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[loc])}$	$\Downarrow'_{(\text{p},[l\hat{o}c])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[rdp2])}$	$\Downarrow'_{(\text{p},[r\hat{d}p2])}$
$\Downarrow^{\mathcal{L}}_{(\text{p},[rdp1])}$	$\Downarrow'_{(\text{p},[r\hat{d}p1])}$	$\Downarrow^{\mathcal{L}}_{(\text{p},[rdp])}$	$\Downarrow'_{(\text{p},[r\hat{d}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wp2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}p])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[da])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{d}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[da1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{d}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[das])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[d\hat{a}s])}$
$\Downarrow^{\mathcal{L}}_{(\text{p},[rea])}$	$\Downarrow'_{(\text{p},[r\hat{e}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ra1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{r}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ra])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{r}a])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wea2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{e}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wea1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{e}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wea])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{e}a])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wa])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wa1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}a])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wa2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{w}a])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[rao1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[r\hat{a}o])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[rao])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[r\hat{a}o])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[ep])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{e}p])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wao2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{a}o])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wao1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{a}o])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[wao])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[w\hat{a}o])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[bd])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{b}d])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[bp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{b}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[bs])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{b}s])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[inp3])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{n}p1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[inp2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{n}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[bm])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[\hat{b}m])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[inp1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{n}p1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[inp])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[i\hat{n}p])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[out])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[o\hat{u}t])}$
$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[out3])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[o\hat{u}t1])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[out2])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[o\hat{u}t])}$	$\Downarrow^{\mathcal{L}}_{\mathcal{D}::(\text{p},[out1])}$	$\Downarrow'_{\hat{\mathcal{D}}::(\text{p},[o\hat{u}t1])}$

Fig. 41. Table of SMC^2 evaluation codes in $SmcC$ and their congruent Vanilla C evaluation codes in $VanC$.

Figure 43a shows erasure over statements. For statements, we case over the various possible statements. When we reach a private value (i.e., encrypt(n)), we decrypt and then return the decrypted value. For function pmalloc, we replace the function name with malloc, modifying the argument to appropriately evaluate the expected size of the type. For functions pfree, smcinput, and smcoutput, we simply replace the function name with its Vanilla C equivalent. All other cases



(a) Well-aligned accesses

(b) Not well-aligned accesses

Fig. 42. Examples of alignment between SMC^2 and Vanilla C in overshooting accesses by incrementing pointer p three times.

recursively call the erasure function as needed, with the last case (_) handling all cases that are already identical to the Vanilla Cequivalent (i.e., NULL, locations).

Figure 43e shows erasure over bytes stored in memory, which is used from within the erasure on the environment and memory. This function takes the byte-wise data representation, the type that it should be interpreted as, and the size expected for the data. For regular public types, we do not need to modify the byte-wise data. For regular private types (i.e., single values and array data), we get back the value(s) from the representation, decrypt, and obtain the byte-wise data for the decrypted value(s). For pointers with a single location, we must get back the pointer data structure, then simply remove the privacy label from the type stored there. For private pointers with multiple locations, we must declassify the pointer, retrieving it's true location and returning the byte-wise data for the pointer data structure with only that location. For functions, we get back the function data, then call Erase on the function body, remove the tag for whether the function has public side effects (i.e., replace with □), and call Erase on the function parameter list.

Figure 44 shows erasure over the environment and memory. In order to properly handle all types of variables and data stored, we must iterate over both the SMC^2 environment and memory maps, and pass along the Vanilla C environment and memory maps as we remove elements from the SMC^2 maps and either add to them to the Vanilla C maps or discard them. The first case is the base case, when the SMC^2 environment and memory are both empty, and we return the Vanilla C environment and memory. Next, we have three cases which continue to iterate through the SMC^2 memory after the environment has been emptied. These cases are possible due to the fact that in SMC^2 we remove mappings from the environment once they are out of scope, but we never remove mappings from memory.

Then we have three cases to handle regular variables. The first adds mappings to the Vanilla C environment and memory without the privacy annotations on the types, and calls Erase on the byte-wise data stored at that location (the behavior of this is shown in Figure 43e and described later in this section). The other two remove temporary variables (an their corresponding data) inserted by an `if else` statement branching on private data. The cases for arrays, pointers, and functions behave similarly; however, when we have an array we handle the array pointer as well as the array data within those cases.

Erase(s) =
| $x[e]$ => $\boldsymbol{x}$[Erase($\boldsymbol{e}$)]
| $[v_0, \ldots, v_n]$ => [Erase($\boldsymbol{v_0}$), Erase($\boldsymbol{\ldots}$), Erase($\boldsymbol{v_n}$)]
| malloc(e) => malloc(Erase($\boldsymbol{e}$))
| pmalloc(e, ty) => malloc(sizeof(Erase($\boldsymbol{ty}$)) $\cdot$ Erase($\boldsymbol{e}$))
| free(e) => free(Erase($\boldsymbol{e}$))
| pfree(e) => free(Erase($\boldsymbol{e}$))
| sizeof(ty) => sizeof(Erase($\boldsymbol{ty}$))
| smcinput(E) => mcinput(Erase($\boldsymbol{E}$))
| smcoutput(E) => mcoutput(Erase($\boldsymbol{E}$))
| $x(E)$ => $\boldsymbol{x}$(Erase($\boldsymbol{E}$))
| e_1 bop e_2 => Erase($\boldsymbol{e_1}$) $\boldsymbol{bop}$ Erase($\boldsymbol{e_2}$)
| $uop\ x$ => $\boldsymbol{uop\ x}$
| (e) => (Erase($\boldsymbol{e}$))
| $(ty)\ e$ => Erase($\boldsymbol{ty}$)) Erase($\boldsymbol{e}$)
| $var = e$ => Erase($\boldsymbol{var}$) = Erase($\boldsymbol{e}$)
| $*x = e$ => $\boldsymbol{*x}$ = Erase($\boldsymbol{e}$)
| $s_1;\ s_2$ => Erase($\boldsymbol{s_1}$); Erase($\boldsymbol{s_2}$)
| $\{s\}$ => {Erase($\boldsymbol{s}$)}
| $ty\ var$ => Erase($\boldsymbol{ty}$) Erase($\boldsymbol{var}$)
| $ty\ var = e$ => Erase($\boldsymbol{ty}$) Erase($\boldsymbol{var}$) = Erase($\boldsymbol{e}$)
| $ty\ x(P)$ => Erase($\boldsymbol{ty}$) $\boldsymbol{x}$(Erase($\boldsymbol{P}$))
| $ty\ x(P)\ \{s\}$ => Erase($\boldsymbol{ty\ x(P)}$) {Erase($\boldsymbol{s}$)}
| if(e) s_1 else s_2 => if(Erase($\boldsymbol{e}$)) Erase($\boldsymbol{s_1}$) else Erase($\boldsymbol{s_2}$)
| while (e) s => while (Erase($\boldsymbol{e}$)) Erase($\boldsymbol{s}$)
| _ => $\boldsymbol{s}$

(a) Erasure function over statements

Erase(C) =
| $C_1 \parallel C_2$ => Erase($\boldsymbol{C_1}$) $\parallel$ Erase($\boldsymbol{C_2}$)
| (p, γ, σ, Δ, acc, s) =>
(p, Erase($\boldsymbol{\gamma}$, $\boldsymbol{\sigma}$, [ ], [ ]), $\square$, $\square$, Erase($\boldsymbol{s}$))

(b) Erasure function over configurations

Erase(ty) =
| $a\ bty$ => $\boldsymbol{bty}$
| $a\ bty*$ => $\boldsymbol{bty*}$
| $tyL \to ty$ => Erase($\boldsymbol{tyL}$) $\to$ Erase($\boldsymbol{ty}$))
| _ => $\boldsymbol{ty}$

Erase(tyL) =
| [ ] => [ ]
| $ty :: tyL$ => Erase($\boldsymbol{ty}$) :: Erase($\boldsymbol{tyL}$)

(c) Erasure function over types and type lists

Erase(E) =
| $E,\ e$ => Erase($\boldsymbol{E}$), Erase($\boldsymbol{e}$)
| e => Erase($\boldsymbol{e}$)
| void => void

Erase(P) =
| $P,\ ty\ var$ => Erase($\boldsymbol{P}$), Erase($\boldsymbol{ty\ var}$)
| $ty\ var$ => Erase($\boldsymbol{ty}$) Erase($\boldsymbol{var}$)
| void => void

(d) Erasure function over lists

Erase(ω, ty, α) =
| (ω, public bty, α) => $\boldsymbol{\omega}$
| (ω, private bty, 1) => v_1 = DecodeVal(ty, 1, ω); v_2 = decrypt(v_1); ω_1 = EncodeVal(bty, v_2); $\boldsymbol{\omega_1}$
| (ω, private bty, α) => [v_1 = DecodeVal(ty, $RT\alpha$, ω); $[v'_1, \ldots, v'_\alpha]$ = [decrypt(v_1), decrypt($\ldots$), decrypt(v_α)];
ω_1 = EncodeVal(bty, $[v'_1, \ldots, v'_\alpha]$); $\boldsymbol{\omega_1}$
| (ω, public $bty*$, 1) => $[1, [(l, \mu)], [1], i]$ = DecodePtr(public $bty*$, 1, ω);
ω_1 = EncodePtr($bty*$, $[1, [(l, \mu)], [1]$, Erase(ty'), $i]$); $\boldsymbol{\omega_1}$
| (ω, private $bty*$, 1) => $[1, [(l, \mu)], [1], i]$ = DecodePtr(private $bty*$, 1, ω);
if ($i = 1$) then {ty_1 = public bty; ty_2 = private bty} else {ty_1 = public $bty*$; ty_2 = private $bty*$};
$\mu_1 = \frac{\mu \cdot \tau(ty_1)}{\tau(ty_2)}$; ω_1 = EncodePtr($bty*$, $[1, [(l, \mu_1)], [1]$, Erase(ty'), $i]$); $\boldsymbol{\omega_1}$
| (ω, private $bty*$, α) => $[\alpha, L, J, i]$ = DecodePtr(private $bty*$, α, ω);
(l, μ) = DeclassifyPtr($[\alpha, L, J, i]$, private $bty*$);
if ($i = 1$) then {ty_1 = public bty; ty_2 = private bty} else {ty_1 = public $bty*$; ty_2 = private $bty*$};
$\mu_1 = \frac{\mu \cdot \tau(ty_1)}{\tau(ty_2)}$; ω_1 = EncodePtr($bty*$, $[1, [(l, \mu_1)], [1], i]$); $\boldsymbol{\omega_1}$
| (ω, $tyL \to ty$, 1) => (s, n, P) = DecodeFun(ω); ω_1 = EncodeFun(Erase(s), $\square$, Erase(P)); $\boldsymbol{\omega_1}$

(e) Erasure function over bytes

Fig. 43. The Erasure function, broken down into various functionalities.

$$
\begin{array}{l}
\mathbf{Erase}(\boldsymbol{\gamma},\ \boldsymbol{\sigma}, \hat{\gamma}, \hat{\sigma}) = \\
\text{match } (\gamma,\ \sigma) \text{ with} \\
\mid\ ([\ ], [\ ])\ \texttt{=>}\ (\hat{\gamma},\ \hat{\sigma}) \\
\mid\ ([\ ], \sigma_1[l \to (\text{NULL},\ \text{void}*,\ \alpha,\ \text{PermL}(\text{Freeable},\ \text{void}*,\ \text{public},\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}([\ ],\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma}[l \to (\text{NULL},\ \text{void}*,\ \hat{\alpha},\ \text{PermL}(p,\ \text{void}*,\ \text{public},\ \hat{\alpha}))])) \\
\mid\ ([\ ], \sigma_1[l \to (\text{NULL},\ \text{void}*,\ \alpha,\ \text{PermL}(\text{Freeable},\ ty,\ \text{private},\ \alpha))]) \\
\qquad \texttt{=>}\ \hat{\alpha} = \left(\frac{\alpha}{\tau(ty)}\right) \cdot \tau(\text{Erase}(ty)) \\
\qquad\qquad (\mathbf{Erase}([\ ],\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma}[l \to (\text{NULL},\ \text{void}*,\ \hat{\alpha},\ \text{PermL}(p,\ \text{void}*,\ \text{public},\ \hat{\alpha}))])) \\
\mid\ ([\ ], \sigma_1[l \to (\omega,\ ty,\ \alpha,\ \text{PermL}(p,\ ty,\ a,\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}([\ ],\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma}[l \to (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{ty}, \boldsymbol{\alpha}),\ \mathbf{Erase}(\boldsymbol{ty}),\ \boldsymbol{\alpha},\ \text{PermL}(p,\ \mathbf{Erase}(\boldsymbol{ty}),\ \text{public},\ \boldsymbol{\alpha}))])) \\
\mid\ ([\ ], \sigma_1[l \to (\omega,\ ty,\ \alpha,\ \text{PermL\_Ptr}(p,\ ty,\ a,\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}([\ ],\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma}[l \to (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{ty}, \boldsymbol{\alpha}),\ \mathbf{Erase}(\boldsymbol{ty}),\ \boldsymbol{\alpha},\ \text{PermL\_Ptr}(p,\ \mathbf{Erase}(\boldsymbol{ty}),\ \text{public},\ \boldsymbol{\alpha}))])) \\
\mid\ ([\ ], \sigma_1[l \to (\omega,\ ty,\ 1,\ \text{PermL\_Fun}(\text{public}))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}([\ ],\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma}[l \to (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{ty}, \mathbf{1}),\ \mathbf{Erase}(\boldsymbol{ty}),\ 1,\ \text{PermL\_Fun}(\text{public}))])) \\
\mid\ (\gamma_1[x \to (l,\ a\ bty)],\ \sigma_1[l \to (\omega,\ a\ bty,\ 1,\ \text{PermL}(p,\ a\ bty,\ a,\ 1))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma}[x \to (l,\ bty)],\ \hat{\sigma}[l \to\ (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{a\ bty}, \mathbf{1}),\ bty,\ 1, \text{PermL}(p,\ bty,\ \text{public},\ 1))])) \\
\mid\ (\gamma_1[res\_n \to (l,\ \text{private}\ bty)],\ \sigma_1[l \to (\omega,\ \text{private}\ bty,\ 1, \text{PermL}(p,\ \text{private}\ bty,\ \text{private},\ 1))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x\_then\_n \to (l,\ a\ bty)],\ \sigma_1[l \to (\omega,\ a\ bty,\ 1, \text{PermL}(p,\ a\ bty,\ a,\ 1))])\ \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x\_else\_n \to (l,\ a\ bty)],\ \sigma_1[l \to (\omega,\ a\ bty,\ 1, \text{PermL}(p,\ a\ bty,\ a,\ 1))])\ \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x \to (l,\ a\ \text{const}\ bty*)],\ \sigma_1[l \to (\omega,\ a\ \text{const}\ bty*,\ 1, \text{PermL}(p,\ a\ \text{const}\ bty*,\ a,\ 1))]) \\
\qquad \texttt{=>}\ \text{DecodePtr}(a\ \text{const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]; \\
\qquad\qquad \sigma_1 = \sigma_2[l_1 \to (\omega_1,\ a\ bty,\ \alpha,\ \text{PermL}(p,\ a\ bty,\ a,\ \alpha))]; \\
\qquad\qquad (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_2},\ \hat{\gamma}[x \to (l,\ \mathbf{Erase}(a\ \text{const}\ bty*))],\ \hat{\sigma}[l \to\ (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{a}\ \textbf{const}\ \boldsymbol{bty*}, \mathbf{1}), \text{const}\ bty*), 1, \\
\qquad\qquad \text{PermL\_Ptr}(p,\ \text{const}\ bty*,\ \text{public},\ 1))][l_1 \to\ (\mathbf{Erase}(\boldsymbol{\omega_1}, \boldsymbol{a\ bty}, \boldsymbol{\alpha}), bty, \alpha, \text{PermL}(p,\ bty,\ \text{public},\ \alpha))])) \\
\mid\ (\gamma_1[x\_then\_n \to (l,\ a\ \text{const}\ bty*)],\ \sigma_1[l \to (\omega,\ a\ \text{const}\ bty*,\ 1,\ \text{PermL\_Ptr}(p,\ a\ \text{const}\ bty*,\ a,\ 1))]) \\
\qquad \texttt{=>}\ \text{DecodePtr}(a\ \text{const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]; \\
\qquad\qquad \sigma_1 = \sigma_2[l_1 \to (\omega_1,\ a\ bty,\ \alpha,\ \text{PermL}(p,\ a\ bty,\ a,\ \alpha))]; (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_2},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x\_else\_n \to (l,\ a\ \text{const}\ bty*)],\ \sigma_1[l \to (\omega,\ a\ \text{const}\ bty*,\ 1,\ \text{PermL}(p,\ a\ \text{const}\ bty*,\ a,\ 1))]) \\
\qquad \texttt{=>}\ \text{DecodePtr}(a\ \text{const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]; \\
\qquad\qquad \sigma_1 = \sigma_2[l_1 \to (\omega_1,\ a\ bty,\ \alpha,\ \text{PermL}(p,\ a\ bty,\ a,\ \alpha))]; (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_2},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x \to (l,\ a\ bty*)],\ \sigma_1[l \to (\omega,\ a\ bty*,\ \alpha, \text{PermL\_Ptr}(p,\ a\ bty*,\ a,\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma}[x \to (l,\ \mathbf{Erase}(a\ bty*))], \\
\qquad\qquad\ \hat{\sigma}[l \to (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{ty}, \boldsymbol{n}),\ \mathbf{Erase}(\boldsymbol{ty}),\ \alpha,\ \text{PermL\_Ptr}(p,\ \mathbf{Erase}(\boldsymbol{ty}),\ \text{public},\ \alpha))])) \\
\mid\ (\gamma_1[x\_then\_n \to (l,\ a\ bty*)],\ \sigma_1[l \to (\omega,\ a\ bty*,\ \alpha, \text{PermL\_Ptr}(p,\ a\ bty*,\ a,\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x\_else\_n \to (l,\ a\ bty*)],\ \sigma_1[l \to (\omega,\ a\ bty*,\ \alpha, \text{PermL\_Ptr}(p,\ a\ bty*,\ a,\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[temp_{ctr}\_n \to (l,\ \text{private}\ bty*)],\ \sigma_1[l \to (\omega,\ \text{private}\ bty*,\ \alpha, \text{PermL\_Ptr}(p,\ \text{private}\ bty*,\ \text{private},\ \alpha))]) \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma},\ \hat{\sigma})) \\
\mid\ (\gamma_1[x \to (l,\ tyL \to ty)], \sigma_1[l \to (\omega,\ tyL \to ty,\ 1,\ \text{PermL\_Fun}(\text{public}))] \\
\qquad \texttt{=>}\ (\mathbf{Erase}(\boldsymbol{\gamma_1},\ \boldsymbol{\sigma_1},\ \hat{\gamma}[x \to (l,\ \mathbf{Erase}(\boldsymbol{tyL} \to \boldsymbol{ty}))], \\
\qquad\qquad\ \hat{\sigma}[l \to (\mathbf{Erase}(\boldsymbol{\omega}, \boldsymbol{tyL} \to \boldsymbol{ty}, \mathbf{1}),\ \mathbf{Erase}(\boldsymbol{tyL} \to \boldsymbol{ty}),\ 1,\ \text{PermL\_Fun}(\text{public}))]))
\end{array}
$$

Fig. 44. Erasure function over the environment and memory

4.2 Correctness: Algorithms

Algorithm 51 $j \leftarrow (l) \nvdash \sigma$

```
1: j = 0
2: (ω, ty, n, PermL(p, ty, a, n)) = σ(l)
3: if a = public then
4:     j = 1
5: end if
6: return j
```

Algorithm 52 $j \leftarrow (l) \vdash \sigma$

```
1: j = 0
2: (ω, ty, n, PermL(p, ty, a, n)) = σ(l)
3: if a = private then
4:     j = 1
5: end if
6: return j
```

Algorithm 53 $L_1 \leftarrow \text{GetLocationSwap}(L, J)$

```
1: L1 = [ ]
2: for all m ∈ {0, …, |J| − 1} do
3:     if J[m] =private 1 then
4:         L1.append(L[m])
5:     end if
6: end for
7: return L1
```

Algorithm 54 $\sigma_2 \leftarrow \text{SwapMemory}(\sigma, \psi)$

```
1: for all L ∈ ψ do
2:     if (L = [(l1, 0), (l2, 0)]) then
3:         σ1[l1 → (ω1, ty1, n1, PermL(p1, ty1, a1, n1))][l2 → (ω2, ty2, n2, PermL(p2, ty2, a2, n2))] = σ
4:         σ2 = σ1[l1 → (ω2, ty2, n2, PermL(p2, ty2, a2, n2))][l2 → (ω1, ty1, n1, PermL(p1, ty1, a1, n1))]
5:     end if
6:     σ = σ2
7: end for
8: return σ2
```

4.3 Correctness: Definitions

Definition 4.1 (ψ). A map ψ is defined as a list of lists of locations, in symbols $\psi = [\,] \mid \psi[L]$, that is formed by tracking which locations are privately switched during the execution of the statement pfree(x) in a SMC[2] program s to enable comparison with the *congruent* Vanilla C program $\hat{s}$.

Definition 4.2 (aligned memory location). A memory location $(l, \mu), (\hat{l}, \hat{\mu})$ is *aligned* if and only if the location refers to either the beginning of a memory block ($\mu = \hat{\mu} = 0$) or the beginning of an element inside an array.

Definition 4.3 (*well-aligned* memory access). An overshooting memory access by an array is *well-aligned* if and only if:

Algorithm 55 $\psi_1 \leftarrow \text{GetFinalSwap}(\psi)$

$\psi_1 = [\,]$
for all $L \in \psi$ **do**
  if $(L = [(l_1, 0), (l_2, 0)])$ **then**
    if $([(l_1, 0), (l_m, 0)] \notin \psi_1)$ **then**
      if $([(l_2, 0), (l_n, 0)] \notin \psi_1)$ **then**
        $\psi_1 = \psi_1[(l_1, 0), (l_2, 0)][(l_2, 0), (l_1, 0)]$
      else
        $\psi_2[(l_2, 0), (l_n, 0)] = \psi_1$
        $\psi_3 = \psi_2[(l_1, 0), (l_n, 0)][(l_2, 0), (l_1, 0)]$
        $\psi_1 = \psi_3$
      end if
    else
      if $([(l_2, 0), (l_n, 0)] \notin \psi_1)$ **then**
        $\psi_2[(l_1, 0), (l_m, 0)] = \psi_1$
        $\psi_3 = \psi_2[(l_1, 0), (l_2, 0)][(l_2, 0), (l_m, 0)]$
        $\psi_1 = \psi_3$
      else
        $\psi_2[(l_1, 0), (l_m, 0)][(l_2, 0), (l_n, 0)] = \psi_1$
        $\psi_3 = \psi_2[(l_1, 0), (l_n, 0)][(l_2, 0), (l_m, 0)]$
        $\psi_1 = \psi_3$
      end if
    end if
  end if
end for
return ψ_1

Algorithm 56 $j \leftarrow \text{CheckIDCongruence}(\psi,\ l_1,\ \hat{l})$

$l_2 = \hat{l}$
$\psi_1 = \text{GetFinalSwap}(\psi)$
if $([(l_1, 0), (l_2, 0)] \in \psi_1)$ **then**
  return 1
else if $(([(l_1, 0), (l_m, 0)] \in \psi_1) \wedge (l_m \neq l_2))$ **then**
  return 0
else if $(([(l_n, 0), (l_2, 0)] \in \psi_1) \wedge (l_n \neq l_1))$ **then**
  return 0
else if $(l_1 == l_2)$ **then**
  return 1
else
  return 0
end if

- the initial memory location is *aligned* and of the expected type,
- the ending memory location is *aligned* and of the expected type, and
- all memory blocks or elements iterated over are of the expected type.

Definition 4.4 ($\eta \cong \hat{\eta}$). A SMC^2 alignment indicator and a Vanilla C alignment indicator are *congruent*, in symbols $\eta \cong \hat{\eta}$,
if and only if either $\eta = 1$ and $\hat{\eta} = 1$
or $\eta = 0$ and $(\hat{\eta} = 0) \vee (\hat{\eta} = 1)$.

Definition 4.5 (*aligned* location list). A location list is *aligned* if and only if for all locations (l_i, μ_i) in the list:

- all memory block identifiers l_i are of the expected type,

Algorithm 57 $j \leftarrow$ CheckCodeCongruence$(D, \hat{D})$

if $(|D| = 0) \wedge (|\hat{D}| = 0)$ **then**
    return 1
else if $(|D| = 1) \wedge (|\hat{D}| = 1)$ **then**
    $[d] = D$
    $[\hat{d}] = \hat{D}$
    if $d \cong \hat{d}$ **then**
        return 1
    else
        return 0
    end if
else
    $[d_0, \ldots, d_n] = D$
    $[\hat{d}_0, \ldots, \hat{d}_m] = \hat{D}$
    if $d_0 = malp$ **then**
        if $(\hat{d}_0 = mal) \wedge (\hat{d}_1 = bm) \wedge (\hat{d}_2 = ty)$ **then**
            return CheckCodeCongruence$([d_1, \ldots, d_n], [\hat{d}_3, \ldots, \hat{d}_m])$
        else
            return 0
        end if
    else
        if $d_0 \cong \hat{d}_0$ **then**
            return CheckCodeCongruence$([d_1, \ldots, d_n], [\hat{d}_1, \ldots, \hat{d}_m])$
        else
            return 0
        end if
    end if
end if

- all memory block identifiers l_i are of the same size, and
- all offsets μ_i are equal.

Definition 4.6 (*well-aligned* pointer access)**.** An overshooting memory access by a pointer is *well-aligned* if and only if:

- the initial location list L_i is *aligned*,
- the final location list L_f is *aligned*, and
- for each location in the initial location list, all memory blocks or elements iterated over to get to the corresponding location in the final location list are of the expected type.

Definition 4.7 ($ty \cong \hat{ty}$)**.** A SMC^2 type and a Vanilla C type are *congruent*, in symbols $ty \cong \hat{ty}$, if and only if Erase$(ty) = \hat{ty}$.

Definition 4.8 ($ty \cong_\psi \hat{ty}$)**.** A SMC^2 type and a Vanilla C type are *ψ-congruent*, in symbols $ty \cong_\psi \hat{ty}$, if and only if $ty \cong \hat{ty}$.

Definition 4.9 ($tyL \cong \hat{tyL}$)**.** A SMC^2 type list and a Vanilla C type list are *congruent*, in symbols $tyL \cong \hat{tyL}$, if and only if Erase$(tyL) = \hat{tyL}$.

Definition 4.10 ($E \cong \hat{E}$)**.** A SMC^2 expression list and a Vanilla C expression list are *congruent*, in symbols $E \cong \hat{E}$, if and only if Erase$(E) = \hat{E}$.

Definition 4.11 ($P \cong \hat{P}$)**.** A SMC^2 parameter list and a Vanilla C parameter list are *congruent*, in symbols $P \cong \hat{P}$, if and only if Erase$(P) = \hat{P}$.

Definition 4.12. A SMC^2 statement and a Vanilla C statement are *congruent*, in symbols $s \cong \hat{s}$, if and only if Erase$(s) = \hat{s}$.

Definition 4.13 ($l \cong_\psi \hat{l}$)**.** A SMC^2 memory block identifier and a Vanilla C memory block identifier are *ψ-congruent*, in symbols $l \cong_\psi \hat{l}$, given map ψ,
if and only if CheckIDCongruence$(\psi, l, \hat{l}) = 1$.

Definition 4.14 ($(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$)**.** A SMC^2 location and a Vanilla C location are *ψ-congruent*, in symbols $(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$, given SMC^2 type ty correlating to (l, μ) and Vanilla C type $\hat{ty}$ correlating to $(\hat{l}, \hat{\mu})$,
if and only if $ty \cong \hat{ty}$, $l \cong_\psi \hat{l}$, and
either ty is a public type and $\mu = \hat{\mu}$,
or ty is a private type and $(\mu) \cdot \left(\frac{\tau(\hat{ty})}{\tau(ty)}\right) = \hat{\mu}$.

Definition 4.15 ($ptr \cong_\psi \hat{ptr}$)**.** A SMC^2 pointer data structure for a pointer of type $ty \in \{a \text{ const } bty*, a\ bty*\}$ and a Vanilla C pointer data structure for a pointer of type $\hat{ty} \in \{\text{const } \hat{bty}*, \hat{bty}*\}$ are *ψ-congruent*, in symbols $[\alpha, L, J, i] \cong_\psi [1, [(\hat{l}, \hat{\mu})], [1], \hat{i}]$, given map ψ,
if $ty \cong \hat{ty}$, $i = \hat{i}$ and
either $a = \text{public}$, $\alpha = 1$, $L = (l, \mu)$ such that $(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$ and $J = [1]$
or $a = \text{private}$ and DeclassifyPtr$([\alpha,\ L,\ J,\ i],\ \text{private } bty*) = (l, \mu)$ such that $(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$.

Definition 4.16 ($(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$)**.** A SMC^2 environment and memory pair and a Vanilla C environment and memory pair are *ψ-congruent*, in symbols $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$,
if and only if Erase$(\gamma, \sigma, [\,], [\,]) = (\hat{\gamma}, \hat{\sigma}')$ and SwapMemory$(\hat{\sigma}', \psi) = \hat{\sigma}$.

Definition 4.17 ($\omega \cong_\psi \hat{\omega}$)**.** A SMC^2 byte-wise representation ω of a given type ty and size n and a Vanilla C byte-wise representation $\hat{\omega}$ are *ψ-congruent*, in symbols $\omega \cong_\psi \hat{\omega}$,
if and only if either $ty \neq$ private $bty*$ and Erase$(\omega, ty, n) = \hat{\omega}$
or $ty ==$ private $bty*$ and Erase$(\omega, ty, n) = \hat{\omega}_1$ such that the pointer data structure stored in ω and the pointer data structure stored in $\hat{\omega}$ are *ψ-congruent* by Definition 4.15.

Definition 4.18 ($v \cong \hat{v}$)**.** A SMC^2 value and Vanilla C value are *congruent*, in symbols $v \cong \hat{v}$, if and only if Erase$(v) = \hat{v}$.

Definition 4.19 ($v \cong_\psi \hat{v}$)**.** A SMC^2 value and Vanilla C value are *ψ-congruent*, in symbols $v \cong_\psi \hat{v}$,
if and only if either $v \neq (l, \mu)$, $\hat{v} \neq (\hat{l}, \hat{\mu})$ and $v \cong \hat{v}$,
or $v = (l, \mu)$, $\hat{v} = (\hat{l}, \hat{\mu})$ and $(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$.

Definition 4.20 ($s \cong_\psi \hat{s}$)**.** A SMC^2 statement and Vanilla C statement are *ψ-congruent*, in symbols $s \cong_\psi \hat{s}$, if and only if for all $v_i \in s$, $\hat{v}_i \in \hat{s}$ such that $v_i \cong_\psi \hat{v}_i$ and otherwise $s \cong \hat{s}$.

Definition 4.21 ($E \cong_\psi \hat{E}$)**.** A SMC^2 expression list and a Vanilla C expression list are *ψ-congruent*, in symbols $E \cong_\psi \hat{E}$, given a map ψ, if and only if $\forall e \neq (l, \mu) \in E$, Erase$(e) = \hat{e}$ and $\forall e == (l, \mu) \in E$, $e \cong_\psi \hat{e}$ by Definition 4.20.

Definition 4.22 ($\{(\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}^\mathrm{p}, s^\mathrm{p}) \cong_\psi (\mathrm{p}, \hat{\gamma}^\mathrm{p}, \hat{\sigma}^\mathrm{p}, \square, \square, \hat{s}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$)**.** A SMC^2 configuration and a Vanilla C configuration are *ψ-congruent*, in symbols $\{(\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}^\mathrm{p}, s^\mathrm{p}) \cong_\psi (\mathrm{p}, \hat{\gamma}^\mathrm{p}, \hat{\sigma}^\mathrm{p}, \square, \square, \hat{s}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$ or $C \cong_\psi \hat{C}$, if and only if $\{(\gamma^\mathrm{p}, \sigma^\mathrm{p}) \cong_\psi (\hat{\gamma}^\mathrm{p}, \hat{\sigma}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and $\{s^\mathrm{p} \cong_\psi \hat{s}^\mathrm{p}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Definition 4.23 ($d \cong \hat{d}$)**.** We define *congruence* over SMC^2 codes $d \in SmcC$ and $\hat{d} \in VanC$, in symbols $d \cong \hat{d}$, by cases as follows:
if $d = \hat{d}$, then $d \cong \hat{d}$,
if $d = iep \oplus iepd$, then $\hat{d} = \widehat{mpiet} \oplus \widehat{mpief}$ and $d \cong \hat{d}$,
if $d = mpcmp$, then $\hat{d} = \widehat{mpcmpt} \oplus \widehat{mpcmpf}$ and $d \cong \hat{d}$,
otherwise we have $[malp] \cong [\hat{ty}, \hat{bm}, \hat{mal}]$, $fc1 \cong \hat{fc}$, $pin3 \cong \widehat{pin}$, $cl1 \cong \hat{cl}$, $mpwdp2 \cong \widehat{mpwdp1}$, $cv1 \cong \hat{cv}$, $mpwdp \cong \widehat{mpwdp}$, $pin4 \cong \widehat{pin1}$, $pin5 \cong \widehat{pin2}$, $mpwdp3 \cong \widehat{mpwdp}$, $pin6 \cong \widehat{pin1}$, $pin7 \cong \widehat{pin2}$, $r1 \cong \hat{r}$, $w1 \cong \hat{w}$, $w2 \cong \hat{w}$, $d1 \cong \hat{d}$, $wdp2 \cong \widehat{wdp1}$, $dp1 \cong \widehat{dp}$, $wdp3 \cong \widehat{wdp}$, $rp1 \cong \hat{rp}$, $wdp4 \cong \widehat{wdp}$, $wp1 \cong \hat{wp}$, $rdp1 \cong \widehat{rdp1}$,

$wp2 \cong \hat{wp}$, $da1 \cong \hat{da}$, $ra1 \cong \hat{ra}$, $wea2 \cong \hat{wea}$, $wea1 \cong \hat{wea}$, $rao1 \cong \hat{rao}$, $wa1 \cong \hat{wa}$, $wa2 \cong \hat{wa}$, $wa1p \cong \hat{wa}$, $wa2p \cong \hat{wa}$, $wao2 \cong \hat{wao}$, $wao1 \cong \hat{wao}$, $inp3 \cong \hat{inp}1$, $inp2 \cong \hat{inp}$, $out3 \cong \hat{out}1$, and $out2 \cong \hat{out}$.

Definition 4.24 ($D \cong \hat{D}$)**.** A SMC^2 evaluation code trace for a single party and a Vanilla C evaluation code trace for a single party are *congruent*, in symbols $D \cong \hat{D}$, if and only if CheckCodeCongruence$(D, \hat{D}) = 1$ by Algorithm 57.

Definition 4.25 ($(\mathrm{p}, D) \cong (\mathrm{p}, \hat{D})$)**.** A party-wise SMC^2 code trace (p, D) and a party-wise Vanilla C code trace $(\mathrm{p}, \hat{D})$ are *congruent*, in symbols $(\mathrm{p}, D) \cong (\mathrm{p}, \hat{D})$, if and only if $D \cong \hat{D}$.

Definition 4.26 ($\Pi \cong_{\psi} \Sigma$)**.** Two derivations and ψ*-congruent*, in symbols $\Pi \cong_{\psi} \Sigma$, if and only if $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, s^{\mathrm{q}}))$ $\Downarrow^{\mathcal{L}}_{\mathcal{D}} ((1, \gamma^1_1, \sigma^1_1, \Delta^1_1, \mathrm{acc}^1_1, v^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}_1, \sigma^{\mathrm{q}}_1, \Delta^{\mathrm{q}}_1, \mathrm{acc}^{\mathrm{q}}_1, v^{\mathrm{q}}))$ and $\Sigma \triangleright ((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \hat{s}^1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}^{\mathrm{q}}, \hat{\sigma}^{\mathrm{q}}, \square, \square, \hat{s}^{\mathrm{q}})) \Downarrow_{\hat{\mathcal{D}}} ((1, \hat{\gamma}^1_1, \hat{\sigma}^1_1, \square, \square, \hat{v}^1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}^{\mathrm{q}}_1, \hat{\sigma}^{\mathrm{q}}_1, \square, \square, \hat{v}^{\mathrm{q}}))$ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}^{\mathrm{p}}, s^{\mathrm{p}}) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}}, \hat{\sigma}^{\mathrm{p}}, \square, \square, \hat{s}^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$, $\mathcal{D} \cong \hat{\mathcal{D}}$, and $\{(\mathrm{p}, \gamma^{\mathrm{p}}_1, \sigma^{\mathrm{p}}_1, \Delta^{\mathrm{p}}_1, \mathrm{acc}^{\mathrm{p}}_1, v^{\mathrm{p}}) \cong_{\psi} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}}_1, \hat{\sigma}^{\mathrm{p}}_1, \square, \square, \hat{v}^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$ such that ψ was derived from ψ_1 and the derivation Π.

Definition 4.27. Two input files are *congruent*, in symbols $inp \cong \hat{inp}$, if and only if for all mappings of variables to number values $x = v \in inp$ and $\hat{x} = \hat{v} \in \hat{inp}$, $x = \hat{x}$ and $v \cong \hat{v}$ by Definition 4.12.

Definition 4.28. Two output files are *congruent*, in symbols $out \cong \hat{out}$, if and only if for all mappings of variables to number values $x = v \in out$ and $\hat{x} = \hat{v} \in \hat{out}$, $x = \hat{x}$ and $v \cong \hat{v}$ by Definition 4.12.

Definition 4.29 (non-constant location)**.** A statement s is considered to update a *non-constant location* if the location that is being updated by such a statement can be modified (such as that which a pointer refers to) or overshot (such as that of a public index into an array).

Definition 4.30 (constant location)**.** A statement s is considered to update a *constant location* if the location that is being updated by such a statement cannot be modified (such l when $\gamma(x) = (l, ty)$) or overshot (such as that of a private index into an array).

Definition 4.31 (Δ complete)**.** The given nesting level of a location map Δ[acc] is considered to be *complete* if all non-local locations that have been updated within the evaluation of the Private If Else statement have mappings within Δ[acc] such that the value in v_{orig} is the original value.

Definition 4.32 (Δ then-complete)**.** The given nesting level of a location map Δ[acc] is considered to be *then-complete* if Δ[acc] is *complete* and
either the location was updated in the `then` branch and therefore has a value stored for v_{then} and tag set to 1, or the location was not updated in the `then` branch.

Definition 4.33 (Δ else-complete)**.** The given nesting level of a location map Δ[acc] is considered to be *else-complete* if Δ[acc] was *then-complete* after the evaluation of restoration and Δ[acc] is *complete* after evaluation after the `else` branch.

Definition 4.34 ($(\gamma, \sigma) \models (x \equiv v)$)**.** Variable x is equivalent to value v in the environment and memory pair (γ, σ), in symbols $(\gamma, \sigma) \models (x \equiv v)$, if and only if there is a valid mapping for x in the environment $\gamma(x) = (l, ty)$ and a corresponding mapping in memory $\sigma(l) = (\omega, ty, \alpha, \mathrm{PermL}(p, ty, a, \alpha))$ such that the byte representation ω can be decoded by the given type ty to obtain value v.

Definition 4.35 ($(\sigma) \models_l ((l, \mu) \equiv_{ty} v)$)**.** The bytes at location (l, μ) interpreted as type ty in the given memory σ are equivalent to the given value v, in symbols $(\sigma) \models_l ((l, \mu) \equiv_{ty} v)$, if and only if there is a valid mapping for l in memory $\sigma(l) = (\omega, ty_1, \alpha, \mathrm{PermL}(p, ty_1, a, \alpha))$ such that the byte representation ω from the offset μ can be decoded by the given type ty to obtain value v.

4.4 Correctness: Lemmas

Lemma 4.1. *Given configuration* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C)$*, if* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C)$*, then* $(l, \mu) \notin s$.

Proof. By case analysis of the semantics, we can see that there is no rule that will evaluate a statement containing (l, μ) to a value n. □

Lemma 4.2. *Given map* ψ*, configuration* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C)$*, environment* $\hat{\gamma}$*, memory* $\hat{\sigma}$*, statement* $\hat{s}$*, and configuration* $\hat{C}$*, if* $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$*,* $s \cong_{\psi} \hat{s}$*, and* $C \cong_{\psi} \hat{C}$*, then* $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C})$ *such that* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C})$.

Proof. By Definitions 4.20 and 4.22. □

Lemma 4.3. *Given values* $v, \hat{v}$ *and environment* γ*, if* $v \cong_{\psi} \hat{v}$ *and* $(v) \nvdash \gamma$*, then* $v = \hat{v}$.

Proof. By Definitions 4.19, 4.18, and the definition of the erasure function Erase. Case analysis on $\mathrm{Erase}(v) = \hat{v}$ gives us that $v = \hat{v}$ when v is public. □

Lemma 4.4. *Given expression* e *and configuration* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C)$ *such that* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, v) \parallel C_1)$*, if* $(e) \nvdash \gamma$*, then* $(v) \nvdash \gamma$.

Proof. By definition of Algorithm 35, we have that all elements in e must be public. By case analysis on rules where $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, v) \parallel C_1)$ and $(e) \nvdash \gamma$, we find that $(v) \nvdash \gamma$ is true. □

Lemma 4.5. *Given map* ψ *and statement* $s, \hat{s}$*, if* $s \cong_{\psi} \hat{s}$ *and* $(l, \mu) \notin s$*, then* $s \cong \hat{s}$.

Proof. Given that s does not contain (l, μ), by Definition 4.12 we have $s \cong \hat{s}$.

This follows directly from the definition of function Erase, and can be proven by case analysis of all statements that are not locations. □

Lemma 4.6. *Given map* ψ *and statement* $s, \hat{s}$*, if* $s \cong \hat{s}$ *and* $(l, \mu) \notin s$*, then* $s \cong_{\psi} \hat{s}$.

Proof. Given that s does not contain (l, μ), by Definition 4.20 we have $s \cong_{\psi} \hat{s}$.

This follows directly from the definition of function Erase, and can be proven by case analysis of all statements that are not locations. □

Lemma 4.7. *Given map* ψ_1, ψ_2 *and statement* $s, \hat{s}$*, if* $s \cong_{\psi_1} \hat{s}$ *and* $(l, \mu) \notin s$*, then* $s \cong_{\psi_2} \hat{s}$.

Proof. Given that s does not contain a hard-coded location (l, μ), by Lemma 4.5 we have that $s \cong \hat{s}$.

Given $s \cong \hat{s}$ and s does not contain a hard-coded location (l, μ), by Lemma 4.6, we have $s \cong_{\psi_2} \hat{s}$.

This follows directly from the definition of function Erase, and can be proven by case analysis of all statements that are not locations – a statement not containing a location will maintain congruency and in turn ψ-congruency for any given map ψ. □

Lemma 4.8. *Given an initial map* ψ*, environment* γ*, memory* σ*, accumulator* acc*, and expression* e*, if* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, v) \parallel C)$ *such that* $v \neq \mathrm{skip}$*, then* $\mathrm{pfree}(e_1) \notin e$ *and the ending map* ψ_1 *is equivalent to* ψ.

Proof. By definition of SMC^2 rule pfree, skip is returned from the evaluation of $\mathrm{pfree}(e_1)$. Therefore, by case analysis of the rules, if $v \neq \mathrm{skip}$, then $\mathrm{pfree}(e_1) \notin e$. By Definition 4.1, ψ is only modified after the execution of function pfree; therefore we have that $\psi_1 == \psi$. □

Lemma 4.9. *Given* ψ *and* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C})$*, if* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma_1, \sigma_1, \Delta, \mathrm{acc}, v) \parallel C_1)$ *and* $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C}) \Downarrow_{\hat{\mathcal{D}}}' ((\hat{\gamma}_1, \hat{\sigma}_1, \square, \hat{v}) \parallel \hat{C})$ *such that* $((\mathrm{p}, \gamma_1, \sigma_1, \Delta, \mathrm{acc}, v) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$*, then* $(\gamma, \sigma_1) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$.

Proof. Proof Sketch: Proof by induction over congruent evaluations.
Using the definition of function Erase, we show that with every rule that adds to γ or adds to or modifies σ maintains both $(\gamma_1, \sigma_1) \cong_{\psi} (\hat{\gamma}_1, \hat{\sigma}_1)$ and $(\gamma, \sigma_1) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$ by Definition 4.16. □

Lemma 4.10 ($\mathcal{D}_1 :: \mathcal{D}_2 \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2$). *Given party-wise code lists $\mathcal{D}_1, \mathcal{D}_2, \hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2$ if $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$ and $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$ then $\mathcal{D}_1 :: \mathcal{D}_2 \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2$.*

Proof. By definition of Algorithm 31, the :: operation is deterministic and maintains party-wise ordering. □

Lemma 4.11. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, and variable name $x, \hat{x}$, if $x \notin \gamma$, $x = \hat{x}$, and $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, then $\hat{x} \notin \hat{\gamma}$.*

Proof. By Definition 4.16. □

Lemma 4.12. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, variable name $x, \hat{x}$, memory block identifier $l, \hat{l}$, and type $ty, \hat{ty}$, if $\gamma_1 = \gamma[x \rightarrow (l, ty)]$, $x = \hat{x}$, $l = \hat{l}$, $ty \cong \hat{ty}$, and $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, then $\hat{\gamma}_1 = \hat{\gamma}[\hat{x} \rightarrow (\hat{l}, \hat{ty})]$ such that $(\gamma_1, \sigma) \cong_{\psi} (\hat{\gamma}_1, \hat{\sigma})$.*

Proof. By Definition 4.16 and the structure of the environment. □

Lemma 4.13. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, type $ty \in \{a\ bty, a\ \mathrm{const}\ bty*, a\ bty*\}$, $\hat{ty}$, byte representation $\omega, \hat{\omega}$, number $n, \hat{n}$, and permission $p, \hat{p}$, if $\sigma_2 = \sigma_1[l \rightarrow (\omega, ty, n, \mathrm{PermL}(p, ty, a, n))]$, $(\gamma, \sigma_1) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$, $l \cong_{\psi} \hat{l}$, $\omega \cong_{\psi} \hat{\omega}$, $\frac{n}{\tau(ty)} = \frac{\hat{n}}{\tau(\hat{ty})}$, and $ty \cong \hat{ty}$, then $\hat{\sigma}_2 = \hat{\sigma}_1[\hat{l} \rightarrow (\omega, ty, \hat{n}, \mathrm{PermL}(p, \hat{ty}, \mathrm{public}, \hat{n}))]$ such that $(\gamma, \sigma_2) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_2)$.*

Proof. By Definition 4.16 and the structure of memory. □

Lemma 4.14. *Given ψ, $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, and $x = \hat{x}$ such that $x \in \gamma$ and $\hat{x} \in \hat{\gamma}$, if $\gamma(x) = (l, ty)$ then $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{ty})$, where $l = \hat{l}$, $(l, 0) \cong_{\psi} (\hat{l}, 0)$, and $ty \cong \hat{ty}$.*

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.15. *Given ψ, $\{(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}$, and $l \cong_{\psi} \hat{l}$ such that $l \in \sigma$ and $\hat{l} \in \hat{\sigma}$, if $\sigma(l) = (\omega, ty, n, \mathrm{PermL}(p, ty, a, n))$ and $ty \neq \mathrm{private}\ bty*$, then $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{ty}, \hat{n}, \mathrm{PermL}(p, \hat{ty}, \mathrm{public}, \hat{n}))$, where $\omega \cong_{\psi} \hat{\omega}$, $ty \cong \hat{ty}$, $n = \hat{n}$, and $p = p$.*

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.16. *Given ψ, $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, and $l \cong_{\psi} \hat{l}$ such that $l \in \sigma$ and $\hat{l} \in \hat{\sigma}$, if $\sigma(l) = (\omega, \mathrm{private}\ bty*, n, \mathrm{PermL\_Ptr}(p, \mathrm{private}\ bty*, \mathrm{private}, n))$ then $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(p, \hat{bty}*, \mathrm{public}, 1))$, where $\omega \cong_{\psi} \hat{\omega}$, $ty \cong \hat{ty}$, and $p = p$.*

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.17 ($(l) \nvDash \sigma \implies l = \hat{l}$). *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, memory block identifier $l, \hat{l}$, if $(l) \nvDash \sigma$, $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, and $l \cong_{\psi} \hat{l}$, then $l = \hat{l}$.*

Proof. Using case analysis over the semantics, we can see that public memory blocks are never swapped around (the only rule that triggers locations being swapped is Multiparty Free, which only ever operates over private memory blocks). □

Lemma 4.18. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, type $ty \in \{a\ bty, a\ \mathrm{const}\ bty*, \mathrm{public}\ bty*\}$, $\hat{ty}$, byte representation $\omega, \hat{\omega}$, number $n, \hat{n}$, and permission $p, \hat{p}$, if $\sigma_1 = \sigma_2[l \rightarrow (\omega, ty, n, \mathrm{PermL}(p, ty, a, n))]$, $(\gamma, \sigma_1) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$, and $l \cong_{\psi} \hat{l}$, then $\hat{\sigma}_1 = \hat{\sigma}_2[\hat{l} \rightarrow (\hat{\omega}, \hat{ty}, \hat{n}, \mathrm{PermL}(p, \hat{ty}, \mathrm{public}, \hat{n}))]$ such that $(\gamma, \sigma_2) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_2)$, $\omega \cong_{\psi} \hat{\omega}$, $n = \hat{n}$, $ty \cong \hat{ty}$, and $p = p$.*

Proof. Using Definition 4.16 and the structure of memory, we can perform case analysis of the semantics to show that, for all types except void∗ and private $bty*$, this holds. The interesting rules for this proof would be those modifying or adding to memory, showing that when they are first stored into memory this holds, and that there isn't anything that will break this property when memory is updated. □

Lemma 4.19. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, type $ty \in \{\text{private } bty*\}$, $\hat{ty} \in \{\hat{bty}*\}$, byte representation $\omega, \hat{\omega}$, number $n, \hat{n}$, and permission $p, \hat{p}$, if $\sigma_1 = \sigma_2[l \to (\omega, ty, n, \text{PermL\_Ptr}(p, ty, \text{private}, n))]$, $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, and $l \cong_\psi \hat{l}$, then $\hat{\sigma}_1 = \hat{\sigma}_2[\hat{l} \to (\hat{\omega}, \hat{ty}, 1, \text{PermL}(p, \hat{ty}, \text{public}, 1))]$ such that $(\gamma, \sigma_2) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$, $\omega \cong_\psi \hat{\omega}$, $ty \cong \hat{ty}$, and $p = p$.*

Proof. Using Definition 4.16 and the structure of memory, we can perform case analysis of the semantics to show that this holds for all private pointers. The interesting rules for this proof would be those modifying or adding to memory of private pointers, showing that when they are first stored into memory this holds, and that there isn't anything that will break this property when memory is updated. □

Lemma 4.20. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, memory block identifier $l, \hat{l}$, and size $n, \hat{n}$, if $\sigma_1 = \sigma\big[l \to (\omega,\ \text{void}*,\ n,\ \text{PermL}(\text{Freeable}, \text{void}*, \text{public}, n))\big]$, $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, and $l = \hat{l}$, then $\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega}, \text{void}*,\ \hat{n},\ \text{PermL}(\text{Freeable}, \text{void}*, \text{public}, \hat{n}))]$ such that $(\gamma,\ \sigma_1) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)$ and $n = \hat{n}$.*

Proof. Using Definition 4.16 and the structure of memory, we can perform case analysis of the semantics to show that this holds for all uncast public memory locations. The interesting rules for this proof would be those operating over uncast public memory (i.e., malloc and cast public location), showing that when they are first stored into memory this holds, and that there isn't anything that will break this property. □

Lemma 4.21. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, memory block identifier $l, \hat{l}$, type $ty, \hat{ty}$, and size $n, \hat{n}$, if $\sigma_1 = \sigma\big[l \to \big(\omega,\ \text{void}*,\ n,\ \text{PermL}(\text{Freeable}, \text{void}*, \text{private}, n)\big]$, $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, and $l \cong_\psi \hat{l}$, then $\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega}, \text{void}*,\ \hat{n},\ \text{PermL}(\text{Freeable}, \text{void}*, \text{public}, \hat{n}))]$ such that $(\gamma,\ \sigma_1) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)$, and $\exists ty, \hat{ty}$ such that $ty \cong \hat{ty}$ and $\frac{n}{\tau(ty)} = \frac{\hat{n}}{\tau(\hat{ty})}$.*

Proof. Using Definition 4.16 and the structure of memory, we can perform case analysis of the semantics to show that this holds for all uncast private memory locations. The interesting rules for this proof would be those operating over uncast private memory (i.e., pmalloc and cast private location), showing that when they are first stored into memory this holds, and that there isn't anything that will break this property. □

Lemma 4.22. *Given map ψ and type $ty \in \{\text{public } bty, \text{public } bty*\}$, $\hat{ty}$, if $ty \cong_\psi \hat{ty}$ then $\tau(ty) = \tau(\hat{ty})$.*

Proof. By definition of τ. □

Lemma 4.23. *Given map ψ, variable name $x, \hat{x}$ and input party number $n, \hat{n}$ such that the corresponding input files inp_n, $\hat{inp}\_\hat{n}$ are congruent, if $\text{InputValue}(x, n) = n_1$, $x = \hat{x}$, and $n = \hat{n}$, then $\text{InputValue}(\hat{x},\ \hat{n}) = \hat{n}_1$ such that $n_1 \cong_\psi \hat{n}_1$.*

Proof. By definition of algorithm InputValue and by Definition 4.27. □

Lemma 4.24. *Given map ψ, variable name $x, \hat{x}$, input party number $n, \hat{n}$ such that the corresponding input files inp_n, $\hat{inp}\_\hat{n}$ are congruent, and array length $n_1, \hat{n}_1$, if $\text{InputArray}(x, n, n_1) = [m_0, ..., m_{n_1}]$, $x = \hat{x}$, $n = \hat{n}$, and $n_1 = \hat{n}_1$, then $\text{InputArray}(\hat{x},\ \hat{n},\ \hat{n}_1) = [\hat{m}_0, ..., \hat{m}_{\hat{n}_1}]$ such that $[m_0, ..., m_{n_1}] \cong_\psi [\hat{m}_0, ..., \hat{m}_{\hat{n}_1}]$.*

Proof. By definition of algorithm InputArray and by Definition 4.27. □

Lemma 4.25. *Given map ψ, variable name $x, \hat{x}$ and input party number $n, \hat{n}$ such that the corresponding input files out_n, $\hat{out}_{\hat{n}}$ are congruent, if $\text{OutputValue}(x, n, n_1)$, $x = \hat{x}$, $n = \hat{n}$, and $n_1 \cong_\psi \hat{n}_1$, then $\text{OutputValue}(\hat{x}, \hat{n}, \hat{n}_1)$ such that $out_n \cong \hat{out}_{\hat{n}}$.*

Proof. By definition of algorithm OutputArray and by Definition 4.28. □

Lemma 4.26. *Given map ψ, variable name x, $\hat{x}$, input party number n, $\hat{n}$ such that the corresponding input files out_n, $\hat{out}_{\hat{n}}$ are congruent, and array $[m_0, ..., m_{n_1}]$, $[\hat{m}_0, ..., \hat{m}_{\hat{n}_1}]$, if* OutputArray$(x, n, [m_0, ..., m_{n_1}])$, $x = \hat{x}$, $n = \hat{n}$, *and* $[m_0, ..., m_{n_1}] \cong_\psi [\hat{m}_0, ..., \hat{m}_{\hat{n}_1}]$, *then* OutputArray$(\hat{x}, \hat{n}, [\hat{m}_0, ..., \hat{m}_{\hat{n}_1}])$ *such that* $out_n \cong \hat{out}_{\hat{n}}$.

Proof. By definition of algorithm OutputArray and by Definition 4.28. □

Lemma 4.27. *Given map ψ, pointer variable being read or dereferenced $x, \hat{x}$, and pointer data structure* $[\alpha,\ L,\ J,\ i]$, $[1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$, *if x refers to* $[\alpha,\ L,\ J,\ i]$, *$\hat{x}$ refers to* $[1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$, *and* $x = \hat{x}$, *then* $[\alpha,\ L,\ J,\ i] \cong_\psi (\hat{l}_1, \hat{\mu}_1)$ *as a resulting value.*

Proof. By case analysis over the semantics, we can see that for every SMC^2 rule that returns multiple locations or accepts multiple locations as a result from an evaluation, there is a congruent Vanilla C rule that has corresponding behavior over a single location, leading to the formation of congruent trees. □

Lemma 4.28. *Given map ψ and configuration* $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, s) \parallel ... \parallel\ (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, s))$, *environment $\hat{\gamma}$, memory $\hat{\sigma}$, and statement $\hat{s}$, if* $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma},\ \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$ *and* $s \cong_\psi \hat{s}$, *then* $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}))$ *such that* $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, s) \parallel ... \parallel\ (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, s)) \cong_\psi ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}))$.

Proof. By Definitions 4.20 and 4.22. □

Lemma 4.29. *Given map ψ, environment* $\{\gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, *memory* $\{\sigma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}$, *variable $x, \hat{x}$ such that* $\{x \in \gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$ *and* $\hat{x} \in \hat{\gamma}$, *if* $\{\gamma^\text{p}(x) = (l^\text{p}, ty)\}_{\text{p}=1}^{\text{q}}$, $\hat{x} = x$, *and* $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$, *then* $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{ty})$, *where* $\{l^\text{p} = \hat{l}\}_{\text{p}=1}^{\text{q}}$, $\{(l^\text{p}, 0) \cong_\psi (\hat{l}, 0)\}_{\text{p}=1}^{\text{q}}$, *and* $ty \cong \hat{ty}$.

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.30. *Given map ψ, environment* $\{\gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, *memory* $\{\sigma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}$, *and memory block identifier* $\{l^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{l}$ *such that* $\{l^\text{p} \in \sigma^\text{p}\}_{\text{p}=1}^{\text{q}}$ *and* $\hat{l} \in \hat{\sigma}$, *if* $\{\sigma^\text{p}(l^\text{p}) = (\omega^\text{p}, ty, n, \text{PermL}(p, ty, a, n))$ *such that* $ty \neq \text{private } bty*$, $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$, $\{l^\text{p} = \hat{l}\}_{\text{p}=1}^{\text{q}}$, *then* $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{ty}, \hat{n}, \text{PermL}(p, \hat{ty}, \text{public}, \hat{n}))$, *where* $\{\omega^\text{p} \cong_\psi \hat{\omega}\}_{\text{p}=1}^{\text{q}}$, $ty \cong \hat{ty}$, $n = \hat{n}$, *and* $p = p$.

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.31. *Given map ψ, environment* $\{\gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, *memory* $\{\sigma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}$, *and memory block identifier* $\{l^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{l}$ *such that* $\{l^\text{p} \in \sigma^\text{p}\}_{\text{p}=1}^{\text{q}}$ *and* $\hat{l} \in \hat{\sigma}$, *if* $\{\sigma^\text{p}(l^\text{p}) = (\omega^\text{p}, \text{private } bty*, n, \text{PermL}(p, \text{private } bty*, a, n))$, $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$, $\{l^\text{p} = \hat{l}\}_{\text{p}=1}^{\text{q}}$, *then* $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL}(p, \hat{bty}*, \text{public}, 1))$, *where* $\{\omega^\text{p} \cong_\psi \hat{\omega}\}_{\text{p}=1}^{\text{q}}$, $\text{private } bty* \cong \hat{bty}*$, *and* $p = p$.

Proof. This holds by Definition 4.16 and the definition of function Erase. □

Lemma 4.32. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, list of values* $[n_0, ..., n_{\alpha-1}]$, $[\hat{n}_0, ..., \hat{n}_{\hat{\alpha}-1}]$, *and type a bty, $\hat{bty}$, if* $\{\forall i \in \{0...\alpha-1\}\ \text{UpdateArr}(\sigma^\text{p}_{1+i}, (l^\text{p}_1, i), n^\text{p}_i, \text{private } bty) = \sigma^\text{p}_{2+i}\}_{\text{p}=1}^{\text{q}}$ $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, $\{l^\text{p} = \hat{l}\}_{\text{p}=1}^{\text{q}}$, $\alpha = \hat{\alpha}$, $\{\omega^\text{p} \cong_\psi \hat{\omega}\}_{\text{p}=1}^{\text{q}}$, $\{\forall i \in \{0...\alpha-1\}\ \text{DecodeArr}(\text{private } bty, i, \omega^\text{p}) = n^\text{p}_i\}_{\text{p}=1}^{\text{q}}$, $\{n'^\text{p}_{\hat{i}} \cong_{\psi_2} \hat{n}\}_{\text{p}=1}^{\text{q}}$, $\{\forall j \neq \hat{i} \in \{0...\alpha-1\} n^\text{p}_j = n'^\text{p}_j\}_{\text{p}=1}^{\text{q}}$, *and a $bty \cong_\psi \hat{bty}$, then* $\text{UpdateArr}(\hat{\sigma}_1,\ (\hat{l}, \hat{i}),\ \hat{n},\ \hat{bty}) = \sigma_2$ *such that* $(\gamma, \sigma_{2+\alpha-1}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$.

Proof. Given a $bty \cong_\psi \hat{bty}$, $\{\omega^\text{p} \cong_\psi \hat{\omega}\}_{\text{p}=1}^{\text{q}}$ and $\{\forall i \in \{0...\alpha-1\}\ \text{DecodeArr}(\text{private } bty, i, \omega^\text{p}) = n^\text{p}_i\}_{\text{p}=1}^{\text{q}}$, by Definition 4.17 and Lemma 4.46, we have that these SMC^2 values read from memory are ψ-congruent to the values stored in the array for Vanilla C.

Given updated list of values $\{[n'^\text{p}_0, ..., n'^\text{p}_{\alpha-1}]\}_{\text{p}=1}^{\text{q}}$ such that $\{n'^\text{p}_{\hat{i}} \cong_{\psi_2} \hat{n}\}_{\text{p}=1}^{\text{q}}$ and $\{\forall j \neq \hat{i} \in \{0...\alpha-1\} n^\text{p}_j = n'^\text{p}_j\}_{\text{p}=1}^{\text{q}}$, we have that only the value at index $\hat{i}$ is modified in the updated list of values. Given this, we are only storing an updated value in memory once, all other values will simply be overwritten with the same value.

By Lemma 4.52, we have that the environment and memory pair maintains ψ-congruency when updating the value that changed within the array for a single party, which in turn holds for all parties.

Given the above, we have the ending environment and memory pairs ψ-congruent, or $(\gamma, \sigma_{2+\alpha-1}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$. □

Lemma 4.33. *Given map ψ, type* private bty, $\hat{bty}$, *pointer data structure* $\{[\alpha, L^p, J^p, 1]\}_{p=1}^q, [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$, *environment* $\{\gamma^p\}_{p=1}^q, \hat{\gamma}$, *and memory* $\{\sigma^p\}_{p=1}^q, \hat{\sigma}$,
if $\{\text{Retrieve\_vals}(\alpha, L^p, \text{private } bty, \sigma^p) = ([n_0^p, ... n_{\alpha-1}^p], 1)\}_{p=1}^q$ $\text{MPC}_{dv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^q, ..., n_{\alpha-1}^q]], [J^1, ..., J^q]) = (n^1, ..., n^q)$, $\{[\alpha, L^p, J^p, 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{p=1}^q$, private $bty \cong_\psi \hat{bty}$, *and* $\{(\gamma^p, \sigma^p) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{p=1}^q$,
then $\text{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)$ *such that* $\{n^p \cong \hat{n}\}_{p=1}^q$.

Proof. By definition of Retrieve_vals, we have $\{[n_0^p, ... n_{\alpha-1}^p]\}_{p=1}^q$ such that each value n_j^p is the value stored at location j in L^p. Therefore, by Axiom 4.10 we have that $\{n^p\}_{p=1}^q$ is the value stored in the true location referred to by the private pointer.

Given $\{[\alpha, L^p, J^p, 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{p=1}^q$, private $bty \cong_\psi \hat{bty}$, and $\{(\gamma^p, \sigma^p) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{p=1}^q$, we have the Vanilla C call $\text{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)$ such that $\{n^p \cong \hat{n}\}_{p=1}^q$. □

Lemma 4.34. *Given map ψ, type* private $bty*$, $\hat{bty}*$, *pointer data structure* $\{[\alpha, L^p, J^p, 1]\}_{p=1}^q, [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$, *environment* $\{\gamma^p\}_{p=1}^q, \hat{\gamma}$, *and memory* $\{\sigma^p\}_{p=1}^q, \hat{\sigma}$,
if $\{\text{Retrieve\_vals}(\alpha, L^p, \text{ private } bty*, \sigma^p) = ([[\alpha_0, L_0^p, J_0^p, i-1], ..., [\alpha_{\alpha-1}, L_{\alpha-1}^p, J_{\alpha-1}^p, i-1]], 1)\}_{p=1}^q$, $\text{MPC}_{dp}([[[\alpha_0, L_0^1, J_0^1], ..., [\alpha_{\alpha-1}, L_{\alpha-1}^1, J_{\alpha-1}^1]], ..., [[\alpha_0, L_0^q, J_0^q], ..., [\alpha_{\alpha-1}, L_{\alpha-1}^q, J_{\alpha-1}^q]]], [J^1, ..., J^q]) = ([[\alpha_\alpha, L_\alpha^1, J_\alpha^1], ..., [\alpha_\alpha, L_\alpha^q, J_\alpha^q]])$, $\{[\alpha, L^p, J^p, 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{p=1}^q$, private $bty* \cong_\psi \hat{bty}*$, *and* $\{(\gamma^p, \sigma^p) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{p=1}^q$,
then $\text{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], 1)$ *such that* $\{[\alpha_\alpha, L_\alpha^q, J_\alpha^q, \hat{i}-1] \cong_\psi [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1]\}_{p=1}^q$.

Proof. By definition of Retrieve_vals, we have $\{\forall j \in \{0...\alpha-1\} [\alpha_j, L_j^p, J_j^p, i-1]\}_{p=1}^q$ such that each pointer data structure $[\alpha_j, L_j^p, J_j^p, i-1]$ is stored at location j in L^p. Therefore, by Axiom 4.11 we have that $\{[\alpha_\alpha, L_\alpha^1, J_\alpha^1]\}_{p=1}^q$ properly indicates the true location of the lower level private pointer that is the true location referred to by the higher level private pointer.

Given $\{[\alpha, L^p, J^p, 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{p=1}^q$, private $bty* \cong_\psi \hat{bty}*$, and $\{(\gamma^p, \sigma^p) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}_{p=1}^q$, we have the Vanilla C call $\text{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], 1)$ such that $\{[\alpha_\alpha, L_\alpha^q, J_\alpha^q, \hat{i}-1] \cong_\psi [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1]\}_{p=1}^q$. □

Lemma 4.35. *Given map ψ, type* private bty, $\hat{bty}$, *pointer data structure* $\{[\alpha, L^p, J^p, 1]\}_{p=1}^q, [1, [(\hat{l}, \hat{\mu})], [1], 1]$, *values* $\{n^p\}_{p=1}^q, \hat{n}$, *environment* $\{\gamma^p\}_{p=1}^q, \hat{\gamma}$, *and memory* $\{\sigma_1^p\}_{p=1}^q, \hat{\sigma}_1$,
if $\{\text{Retrieve\_vals}(\alpha, L^p, \text{private } bty, \sigma_1^p) = ([n_0^p, ... n_{\alpha-1}^p], 1)\}_{p=1}^q$, $\text{MPC}_{wdv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^q, ..., n_{\alpha-1}^q]], [n^1, ..., n^q], [J^1, ..., J^q]) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^q, ..., n_{\alpha-1}'^q])$, *and* $\{\text{UpdateDerefVals}(\alpha, L^p, [n_0'^p, ..., n_{\alpha-1}'^p], \text{private } bty, \sigma_1^p) = \sigma_2^p\}_{p=1}^q$, $\{[\alpha, L^p, J^p, 1] \cong_\psi [1, [(\hat{l}, \hat{\mu})], [1], 1]\}_{p=1}^q$, $\{n^p \cong_\psi \hat{n}\}_{p=1}^q$, private $bty \cong_\psi \hat{bty}$, *and* $\{(\gamma^p, \sigma_1^p) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)\}_{p=1}^q$,
then $\text{UpdateOffset}(\hat{\sigma}_1, (\hat{l}, \hat{\mu}), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, 1)$ *such that* $\{(\gamma^p, \sigma_2^p) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)\}_{p=1}^q$.

Proof. By definition of Retrieve_vals, we have $\{[n_0^p, ... n_{\alpha-1}^p]\}_{p=1}^q$ such that each value n_j^p is the value stored at location j in L^p. Therefore, by Axiom 4.12 we have that $\{n_j'^p = n^p\}_{p=1}^q$ and $\{\forall i \neq j \in \{0...\alpha-1\}$ $n_i'^p = n_i^p\}_{p=1}^q$.

Given $\{[\alpha, L^{\text{p}}, J^{\text{p}}, 1] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{\text{p}=1}^{\text{q}}$, private $bty \cong_{\psi} \hat{bty}$, and $\{(\gamma^{\text{p}}, \sigma^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$, we have the Vanilla C call UpdateOffset$(\hat{\sigma}_1, (\hat{l}, \hat{\mu}), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, 1)$ such that $\{(\gamma^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$. □

Lemma 4.36. *Given map ψ, type* private $bty*$, $\hat{bty}*$, *pointer data structure* $\{[\alpha, L^{\text{p}}, J^{\text{p}}, 1]\}_{\text{p}=1}^{\text{q}}$, $[1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$, *location* $\{(l_e^{\text{p}}, \mu_e^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $(\hat{l}_e, \hat{\mu}_e)$ *environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, *and memory* $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_1$,
if $\{$Retrieve_vals$(\alpha,\ L^{\text{p}},$ private $bty*,\ \sigma_1^{\text{p}}) = ([[\alpha_0,\ L_0^{\text{p}},\ J_0^{\text{p}}, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\text{p}}, J_{\alpha-1}^{\text{p}}, i-1]], 1)\}_{\text{p}=1}^{\text{q}}$, MPC$_{wdp}([[[1, [(l_e^1, \mu_e^1)], [1], i-1], [\alpha_0,\ L_0^1,\ J_0^1, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1, i-1]], ..., [[1, [(l_e^{\text{q}}, \mu_e^{\text{q}})], [1], i-1], [\alpha_0,\ L_0^{\text{q}},\ J_0^{\text{q}}, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\text{q}},\ J_{\alpha-1}^{\text{q}}, i-1]]], [J^1, ..., J^{\text{q}}]) = [[[\alpha_0',\ L_0'^1,\ J_0'^1, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^1,\ J_{\alpha-1}'^1, i-1]], ..., [[\alpha_0',\ L_0'^{\text{q}},\ J_0'^{\text{q}}, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^{\text{q}},\ J_{\alpha-1}'^{\text{q}}, i-1]]]$, $\{$UpdateDerefVals$(\alpha, L^{\text{p}}, [[\alpha_0', L_0'^{\text{p}}, J_0'^{\text{p}}, i-1], ..., [\alpha_{\alpha-1}', L_{\alpha-1}'^{\text{p}}, J_{\alpha-1}'^{\text{p}}, i-1]]$, private $bty*, \sigma_1^{\text{p}}) = \sigma_2^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{[\alpha, L^{\text{p}}, J^{\text{p}}, 1] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{\text{p}=1}^{\text{q}}$, $\{(l_e^{\text{p}}, \mu_e^{\text{p}}) \cong_{\psi_1} (\hat{l}_e, \hat{\mu}_e)\}_{\text{p}=1}^{\text{q}}$, private $bty* \cong_{\psi} \hat{bty}*$, *and* $\{(\gamma^{\text{p}}, \sigma_1^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)\}_{\text{p}=1}^{\text{q}}$,
then UpdatePtr$(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}-1], \hat{bty}*) = (\hat{\sigma}_2, 1)$ *such that* $\{(\gamma^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Retrieve_vals, we have $\{\forall j \in \{0...\alpha-1\}[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}, i-1]\}_{\text{p}=1}^{\text{q}}$ such that each pointer data structure $[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}, i-1]$ is stored at location j in L^{p}. Therefore, by Axiom 4.13 we have that $[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}]$ has the true location set as $(l_e^{\text{p}}, \mu_e^{\text{p}})$ and $\forall i \neq j \in \{0...\alpha-1\}[\alpha_i,\ L_i^{\text{p}},\ J_i^{\text{p}}]$, the true location remains the same as what it originally was.

Given $\{[\alpha,\ L^{\text{p}},\ J^{\text{p}}, 1] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{\text{p}=1}^{\text{q}}$, private $bty* \cong_{\psi} \hat{bty}*$, $\{(l_e^{\text{p}}, \mu_e^{\text{p}}) \cong_{\psi_1} (\hat{l}_e, \hat{\mu}_e)\}_{\text{p}=1}^{\text{q}}$ and $\{(\gamma^{\text{p}}, \sigma^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$, by definition of UpdatePtr, we have the Vanilla C call UpdatePtr$(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}-1], \hat{bty}*) = (\hat{\sigma}_2, 1)$ such that $\{(\gamma^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$. □

Lemma 4.37. *Given map ψ, pointer data structure* $\{[\alpha,\ L^{\text{p}},\ J^{\text{p}}, 1]\}_{\text{p}=1}^{\text{q}}$, $[1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$, *environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, *and memory* $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_1$,
if $\{\forall (l_m^{\text{p}}, 0) \in L^{\text{p}}.\ \sigma^{\text{p}}(l_m^{\text{p}}) = (\omega_m^{\text{p}}, ty, n,$ PermL(Freeable, ty, private, $n))\}_{\text{p}=1}^{\text{q}}$, MPC$_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^{\text{q}}, ..., \omega_{\alpha-1}^{\text{q}}]], [J^1, ...J^{\text{q}}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^{\text{q}}, ..., \omega_{\alpha-1}'^{\text{q}}]], [J'^1, ..., J'^{\text{q}}])$, $\{$UpdateBytesFree$(\sigma^{\text{p}}, L^{\text{p}}, [\omega_0'^{\text{p}}, ..., \omega_{\alpha-1}'^{\text{p}}]) = \sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{\sigma_2^{\text{p}} =$ UpdatePointerLocations$(\sigma_1^{\text{p}}, L^{\text{p}}[1 : \alpha-1], J^{\text{p}}[1 : \alpha-1], L^{\text{p}}[0], J^{\text{p}}[0])\}_{\text{p}=1}^{\text{q}}$, $\{[\alpha,\ L^{\text{p}}\ J^{\text{p}},\ i] \cong_{\psi} [1, [(\hat{l}_1, 0)], [1], \hat{i}]\}_{\text{p}=1}^{\text{q}}$, *and* $\{(\gamma^{\text{p}}, \sigma^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\text{p}=1}^{\text{q}}$,
then Free$(\hat{\sigma}, \hat{l}_1) = \hat{\sigma}_1$ *and* ψ_1 *such that* $\{(\gamma^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}_{\text{p}=1}^{\text{q}}$.

Proof. Proof Sketch:
Given $\{\forall (l_m^{\text{p}}, 0) \in L.\ \sigma^{\text{p}}(l_m^{\text{p}}) = (\omega_m^{\text{p}}, ty, n,$ PermL(Freeable, ty, private, $n))\}_{\text{p}=1}^{\text{q}}$, we have pulled all the byte representations for each location within L^{p}.

Given MPC$_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^{\text{q}}, ..., \omega_{\alpha-1}^{\text{q}}]], [J^1, ...J^{\text{q}}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^{\text{q}}, ..., \omega_{\alpha-1}'^{\text{q}}]], [J'^1, ..., J'^{\text{q}}])$, we have that either tag 0 was not the true location and therefore the byte representation for a location j was swapped with the byte representation for 0 and all others remain the same, or 0 was the true location and all byte representations remain constant.

If the locations were swapped, we obtain ψ_1 by add a mapping to ψ indicating that location 0 was swapped with location j. If the locations were not swapped, $\psi_1 = \psi$.

Given $\{$UpdateBytesFree$(\sigma^{\text{p}}, L^{\text{p}}, [\omega_0'^{\text{p}}, ..., \omega_{\alpha-1}'^{\text{p}}]) = \sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, by definition of UpdateBytesFree, we have that each of the updated byte representations are placed into memory at their corresponding locations, with the permissions at the first location marked as Freeable.

Given $\{\sigma_2^{\text{p}} =$ UpdatePointerLocations$(\sigma_1^{\text{p}}, L^{\text{p}}[1 : \alpha-1], J^{\text{p}}[1 : \alpha-1], L^{\text{p}}[0], J^{\text{p}}[0])\}_{\text{p}=1}^{\text{q}}$, by definition of UpdatePointerLocations we will iterate through and find all private pointers. If the private pointer had location 0 in its location list, we will appropriately update the location list to store the union of what it was and what the location list of the pointer we just freed was, merging the lists and updating the tags so that, if the location we freed was it's true location and we swapped the byte data to a new location, the pointer will now refer it's true location to the location j that we swapped the data to.

Once we have ensured all pointers that could have been affected by the swapping of locations are properly updated, we obtain $\{(\gamma^{p},\ \sigma_2^{p}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}_{p=1}^{q}$. □

Axiom 4.1. *Given a SMC² program of statement s and a ψ-congruent Vanilla C program of statement $\hat{s}$, in symbols $s \cong_{\psi} \hat{s}$, any time a new memory block identifier is obtained from the available pool in the SMC² program such that $l = \phi()$, an identical memory block identifier is obtained from the available pool in the Vanilla C program such that $\hat{l} = \phi()$ and $l = \hat{l}$ and $(l, 0) \cong_{\psi} (\hat{l}, 0)$.*

Lemma 4.38. *Given* $*, *$ *if* $\text{GetIndirection}(*) = i$ *and* $|*| = |*|$, *then* $\text{GetIndirection}(*) = \hat{i}$ *such that* $i = \hat{i}$.

Proof. Proof Sketch:
By definition of function Erase, when two types are congruent, their levels of indirection will be the same. Therefore, when we evaluate the level of indirection from the number of $*$, we will get the same number in both SMC² and Vanilla C. □

Lemma 4.39. *Given parameter list* $P, \hat{P}$ *and* ψ, *if* $\text{GetFunTypeList}(P) = tyL$ *and* $P \cong_{\psi} \hat{P}$, *then* $\text{GetFunTypeList}(\hat{P}) = \hat{tyL}$ *such that* $tyL \cong_{\psi} \hat{tyL}$.

Proof. Proof Sketch:
By the definition of Algorithm GetFunTypeList, GetFunTypeList, and function Erase. □

Lemma 4.40. *Given parameter list* $P, \hat{P}$ *and expression list* $E, \hat{E}$, *if* $\text{GetFunParamAssign}(P, E) = s_1$, $P \cong \hat{P}$, *and* $E \cong \hat{E}$, *then* $\text{GetFunParamAssign}(\hat{P}, \hat{E}) = \hat{s}_1$ *where* $s_1 \cong_{\psi} \hat{s}_1$.

Proof. By definition of GetFunParamAssign. □

Lemma 4.41. *Given map* ψ, *pointer type* $ty \in \{a\ \text{const}\ bty*,\ a\ bty*\}$, $\hat{ty} \in \{\text{const}\ \hat{bty}*,\ \hat{bty}*\}$, *and pointer data structure* $[1, [(l, \mu)], [1], i]$, $[1, [(\hat{l}, \hat{\mu})], [1], \hat{i}]$, *if* $\text{EncodePtr}(ty, [1, [(l, \mu)], [1], i]) = \omega$, $ty \cong \hat{ty}$, $(l, \mu) \cong_{\psi} (\hat{l}, \hat{\mu})$, *then* $\text{EncodePtr}(\hat{ty}, [1, [(\hat{l}, \hat{\mu})], [1], \hat{i}]) = \hat{\omega}$ *such that* $\omega \cong_{\psi} \hat{\omega}$.

Proof. By definition of Algorithm EncodePtr, EncodePtr, and function Erase. □

Lemma 4.42. *Given map* ψ, *type* $ty \in \{a\ bty\}$, $\hat{bty}$, *and value* n, $\hat{n}$, *if* $\text{EncodeVal}(ty, n) = \omega$, $n \cong \hat{n}$, $ty \cong \hat{bty}$, *then* $\text{EncodeVal}(\hat{bty}, \hat{n}) = \hat{\omega}$ *such that* $\omega \cong_{\psi} \hat{\omega}$.

Proof. By definition of Algorithm EncodeVal, EncodeVal, and definition of function Erase. □

Lemma 4.43. *Given map* ψ, *type* $ty \in \{a\ bty\}$, $\hat{bty}$, *value* n, $\hat{n}$, i_1, i_2, $\hat{i}_1, \hat{i}_2$, *if* $\text{EncodeArr}(ty, i_1, i_2, n) = \omega$, $n \cong_{\psi} \hat{n}$ $i_1 = \hat{i}_1$, $i_2 = \hat{i}_2$, *and* $ty \cong_{\psi} \hat{bty}$, *then* $\text{EncodeArr}(\hat{bty}, \hat{i}_1, \hat{i}_2, v) = \hat{\omega}$ *such that* $\omega \cong_{\psi} \hat{\omega}$.

Proof. By definition of Algorithm EncodeArr, EncodeArr, and the definition of function Erase. □

Lemma 4.44. *Given map* ψ, *statement* s, $\hat{s}$, *value* n, *and parameter list* $P, \hat{P}$, *if* $\text{EncodeFun}(s,\ n,\ P) = \omega$, $s \cong_{\psi} \hat{s}$, *and* $P \cong_{\psi} \hat{P}$, *then* $\text{EncodeFun}(\hat{s}, \square, \hat{P}) = \hat{\omega}$ *such that* $\omega \cong_{\psi} \hat{\omega}$.

Proof. By definition of Algorithm EncodeFun, EncodeFun, and the definition of Erase. □

Lemma 4.45. *Given map* ψ, *type* $a\ bty$, $\hat{bty}$, *and byte representation* $\omega, \hat{\omega}$, *if* $\text{DecodeVal}(a\ bty,\ \omega) = n$, $a\ bty \cong \hat{bty}$ *and* $\omega \cong_{\psi} \hat{\omega}$, *then* $\text{DecodeVal}(\hat{bty}, \hat{\omega}) = \hat{n}$ *and* $n \cong_{\psi} \hat{n}$.

Proof. By case analysis of the semantics, Lemma 4.42, definition of Algorithm DecodeVal, DecodeVal and function Erase. □

Lemma 4.46. *Given map* ψ, *type* $a\ bty$, $\hat{bty}$, *index* $i, \hat{i}$, *and byte representation* $\omega, \hat{\omega}$, *if* $\text{DecodeArr}(a\ bty,\ i\ \omega) = n$, $a\ bty \cong \hat{bty}$, $i \cong_{\psi} \hat{i}$, *and* $\omega \cong_{\psi} \hat{\omega}$, *then* $\text{DecodeArr}(\hat{bty}, \hat{i}, \hat{\omega}) = \hat{n}$ *and* $n \cong_{\psi} \hat{n}$.

Proof. By case analysis of the semantics, Lemma 4.43, definition of Algorithm DecodeArr, DecodeArr and function Erase. □

Lemma 4.47. *Given map ψ, type a $bty, \hat{bty}$, index $i \in \{0...\alpha - 1\}, \hat{i} \in \{0...\hat{\alpha} - 1\}$, and byte representation $\omega, \hat{\omega}$, if $\forall i \in \{0...\alpha - 1\}$ $\mathrm{DecodeArr}(a\ bty,\ i\ \omega) = n_i$, $a\ bty \cong \hat{bty}$, $\alpha = \hat{\alpha}$, and $\omega \cong_\psi \hat{\omega}$, then $\forall \hat{i} \in \{0...\hat{\alpha} - 1\}$ $\mathrm{DecodeArr}(\hat{bty}, \hat{i}, \hat{\omega}) = \hat{n}_{\hat{i}}$ such that $\forall i \in \{0...\alpha - 1\}$ $n_i \cong_\psi \hat{n}_i$.*

Proof. By case analysis of the semantics, Lemma 4.43, definition of Algorithm DecodeArr, DecodeArr and function Erase. □

Lemma 4.48. *Given map ψ, type a $bty*$, $\hat{bty}*$, number of locations $\alpha, 1$, and byte representation $\omega, \hat{\omega}$, if $\mathrm{DecodePtr}(a\ bty*,\ \alpha\ \omega) = [\alpha, L, J, i]$, $a\ bty* \cong \hat{bty}*$, and $\omega \cong_\psi \hat{\omega}$, then $\mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}, \hat{\mu})], [1], \hat{i}]$ such that $[\alpha, L, J, i] \cong_\psi [1, [(\hat{l}, \hat{\mu})], [1], \hat{i}]$.*

Proof. By case analysis of the semantics, Lemma 4.41, definition of Algorithm DecodePtr, DecodePtr and function Erase. □

Lemma 4.49. *Given map ψ, type a $\mathrm{const}\ bty*$, $\mathrm{const}\ \hat{bty}*$, number of locations $\alpha, 1$, and byte representation $\omega, \hat{\omega}$, if $\mathrm{DecodePtr}(a\ \mathrm{const}\ bty*,\ \alpha\ \omega) = [\alpha, L, J, i]$, $a\ \mathrm{const}\ bty* \cong \mathrm{const}\ \hat{bty}*$, and $\omega \cong_\psi \hat{\omega}$, then $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}, 0)], [1], \hat{i}]$ such that $[1, [(l, 0)], [1], i] \cong_\psi [1, [(\hat{l}, 0)], [1], \hat{i}]$ and $l = \hat{l}$.*

Proof. By case analysis of the semantics, Lemma 4.41, definition of Algorithm DecodePtr, DecodePtr and function Erase.

We obtain that $l = \hat{l}$ for a constant pointer (array) type by case analysis of the semantics, showing that the location that a constant pointer cannot be changed after it is declared. □

Lemma 4.50. *Given map ψ and byte representation $\omega, \hat{\omega}$, if $\mathrm{DecodeFun}(\omega) = (s, n, P)$, and $\omega \cong_\psi \omega$, then $\mathrm{DecodeFun}(\hat{\omega}) = (\hat{s}, \square, \hat{P})$, $s \cong_\psi \hat{s}$ and $P \cong_\psi \hat{P}$.*

Proof. By case analysis of the semantics, Lemma 4.44, definition of Algorithm DecodeFun and DecodeFun, and definition of function Erase. □

Lemma 4.51. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, value $n, \hat{n}$, and type a $bty, \hat{bty}$, if $\mathrm{UpdateVal}(\sigma_1, l, n, a\ bty) = \sigma_2$, $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, $l \cong_\psi \hat{l}$, $n \cong_\psi \hat{n}$, and $a\ bty \cong \hat{bty}$, then $\mathrm{UpdateVal}(\hat{\sigma}_1, \hat{l}, \hat{n}, \hat{bty}) = \hat{\sigma}_2$ such that $(\gamma, \sigma_2) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$.*

Proof. By definition of Algorithms UpdateVal, UpdateVal, and Erase. □

Lemma 4.52. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, value $n, \hat{n}$, index $i, \hat{i} \in \{0...\alpha - 1\}$, and type a $bty, \hat{bty}$, if $\mathrm{UpdateArr}(\sigma_1,\ (l, i),\ n,\ a\ bty) = \sigma_2$, $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, $l = \hat{l}$, $i = \hat{i}$, $n \cong_\psi \hat{n}$, and $a\ bty \cong_\psi \hat{bty}$, then $\mathrm{UpdateArr}(\hat{\sigma}_1,\ (\hat{l}, \hat{i}),\ \hat{n},\ \hat{bty}) = \sigma_2$ such that $(\gamma, \sigma_2) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$.*

Proof. By definition of Algorithms UpdateArr, UpdateArr, and Erase. □

Lemma 4.53. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, list of values $[n_0, ..., n_{\alpha-1}]$, $[\hat{n}_0, ..., \hat{n}_{\hat{\alpha}-1}]$, and type a $bty, \hat{bty}$, if $\forall i \in \{0...\alpha - 1\}$ $\mathrm{UpdateArr}(\sigma_{1+i},\ (l, i),\ n_i,\ a\ bty) = \sigma_{2+i}$, $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, $l = \hat{l}$, $\alpha = \hat{\alpha}$, $[n_0, ..., n_{\alpha-1}] \cong_\psi [\hat{n}_0, ..., \hat{n}_{\hat{\alpha}-1}]$, and $a\ bty \cong_\psi \hat{bty}$, then $\forall \hat{i} \in \{0...\hat{\alpha} - 1\}$ $\mathrm{UpdateArr}(\hat{\sigma}_{1+\hat{i}},\ (\hat{l}, \hat{i}),\ \hat{n}_{\hat{i}},\ \hat{bty}) = \sigma_{2+\hat{i}}$ such that $(\gamma, \sigma_{2+i}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_{2+\hat{i}})$.*

Proof. By definition of Algorithms UpdateArr, UpdateArr, and Erase, and Lemma 4.52.

Lemma 4.52 gives us that this holds when updating a single value within an array. Given that we have α values and are updating each of them sequentially, we have that each intermediate step i maintains $(\gamma, \sigma_{2+i}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_{2+i})$, and therefore the final memory maintains $(\gamma, \sigma_{2+\alpha-1}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_{2+\alpha-1})$. □

Lemma 4.54. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, location $(l, \mu), (\hat{l}, \hat{\mu})$, pointer data structure $[\alpha, L, J, i]$, $[1,\ [(\hat{l}_1,\ \hat{\mu}_1)],\ [1],\ \hat{i}]$, and type a $bty*, \hat{bty}*$, if $\mathrm{UpdatePtr}(\sigma,\ (l, \mu),\ [\alpha,\ L,\ J,\ i],\ a\ bty*) = (\sigma_1, \eta)$, $a\ bty* \cong \hat{bty}*$, $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, $(l, \mu) \cong_\psi (\hat{l}, \hat{\mu})$, and $[\alpha, L, J, i] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$, then $\mathrm{UpdatePtr}(\hat{\sigma}, (\hat{l}, \hat{\mu}),$ $[1,\ [(\hat{l}_1, \hat{\mu}_1)],\ [1],\ i],\ \hat{bty}*) = (\hat{\sigma}_1, \hat{\eta})$ such that $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$ and $\eta = \hat{\eta}$.*

Proof. By definition of UpdatePtr, UpdatePtr, and Erase, as well as Definition 4.15, 4.14, and 4.4. □

Lemma 4.55. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, memory block identifier $l, \hat{l}$, type $a\ bty, \hat{bty}$, and array index $i, \hat{i}$ and size $n, \hat{n}$, if* $\text{ReadOOB}(i,\ n,\ l,\ a\ bty,\ \sigma) = (n,\ \eta, (l_1, \mu))$, $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, $i = \hat{i}$, $n = \hat{n}$, $l = \hat{l}$, *and* $a\ bty \cong \hat{bty}$, *then* $\text{ReadOOB}(\hat{i},\ \hat{n},\ \hat{l},\ \hat{bty},\ \hat{\sigma}) = (\hat{v},\ \hat{\eta})$ *such that* $n \cong_\psi \hat{n}$ *and* $\eta = \hat{\eta}$.

Proof. By definition of ReadOOB, if the number returned with the updated memory is 1, then the out of bounds access was *well-aligned* by Definition 4.3. Therefore, when we iterate over the ψ*-congruent* Vanilla C memory, the resulting out of bounds access will also be *well-aligned*. We use the definition of ReadOOB, ReadOOB, and Erase to help prove this. □

Lemma 4.56. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma_1, \hat{\sigma}_1$, memory block identifier $l, \hat{l}$, type $a\ bty, \hat{bty}$, value $n, \hat{n}$, array index $i, \hat{i}$ and size $\alpha, \hat{\alpha}$, if* $\text{WriteOOB}(n, i, \alpha, l, a\ bty, \sigma_1) = (\sigma_2, \eta, (l_2, \mu))$, $n \cong_\psi \hat{n}$, $i = \hat{i}$, $\alpha = \hat{\alpha}$, $l = \hat{l}$, $a\ bty \cong_\psi \hat{bty}$, *and* $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$, *then* $\text{WriteOOB}(\hat{n},\ \hat{i},\ \hat{\alpha},\ \hat{l},\ \hat{bty},\ \hat{\sigma}_1) = (\hat{\sigma}_2,\ \hat{\eta})$ *such that* $(\gamma, \sigma_2) \cong_\psi (\hat{\gamma}, \hat{\sigma}_2)$ *and* $\eta \cong \hat{\eta}$.

Proof. Proof Idea:
By definition of WriteOOB, if the number returned with the updated memory is 1, then the out of bounds access was *well-aligned* by Definition 4.3. Therefore, when we iterate over the ψ*-congruent* Vanilla C memory, the resulting out of bounds access will also be *well-aligned*. We use the definition of WriteOOB, WriteOOB, and Erase to help prove this. □

Lemma 4.57. *Given map ψ, location $(l_1, \mu_1), (\hat{l}_1, \hat{\mu}_1)$, type $ty, \hat{ty}$, number $n, \hat{n}$, environment $\gamma, \hat{\gamma}$, and memory $\sigma, \hat{\sigma}$, if* $\text{GetLocation}((l_1,\ \mu_1),\ n, \sigma) = ((l_2, \mu_2), \eta)$, $(l_1, \mu_1) \cong_\psi (\hat{l}_1, \hat{\mu}_1)$, $ty \cong \hat{ty}$, $\tau(ty) = n$, $\tau(\hat{ty}) = \hat{n}$, *and* $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, *then* $\text{GetLocation}((\hat{l}_1,\ \hat{\mu}_1),\ \hat{n}, \hat{\sigma}) = ((\hat{l}_2,\ \hat{\mu}_2), \hat{\eta})$ *such that* $(l_2, \mu_2) \cong_\psi (\hat{l}_2, \hat{\mu}_2)$ *and* $\eta \cong \hat{\eta}$.

Proof. By definition of algorithms GetLocation and Erase and Definition 4.14. □

Lemma 4.58. *Given location list L, location $(\hat{l}, \hat{\mu})$, type $ty, \hat{ty}$, number $n, \hat{n}$, map ψ, environment $\gamma, \hat{\gamma}$, and memory $\sigma, \hat{\sigma}$, if* $\text{IncrementList}(L, n, \sigma) = (L', \eta)$, $\text{DeclassifyPtr}([\alpha,\ L,\ J,\ i], ty) = (l_1, \mu_1)$ *such that* $(l_1, \mu_1) \cong_\psi (\hat{l}_1, \hat{\mu}_1)$, $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, $ty \cong \hat{ty}$, $\tau(ty) = n$, $\tau(\hat{ty}) = \hat{n}$, *and* $\text{DeclassifyPtr}([\alpha,\ L',\ J,\ i], \text{private } bty*) = (l_2, \mu_2)$, *then* $\text{GetLocation}((\hat{l}_1, \hat{\mu}_1), \hat{n}, \hat{\sigma}) = ((\hat{l}_2, \hat{\mu}_2), \hat{\eta})$ *such that* $(l_2, \mu_2) \cong_\psi (\hat{l}_2, \hat{\mu}_2)$ *and* $\eta \cong \hat{\eta}$.

Proof. By definition of algorithms IncrementList, GetLocation, and Erase and Definitions 4.14 and Def: ptr list cong. □

Lemma 4.59. *Given map ψ, memory $\sigma, \hat{\sigma}$ and environment $\gamma, \hat{\gamma}$ such that $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, and memory block identifier $l, \hat{l}$, if* $\text{Free}(\sigma, l, \gamma) = \sigma_1$ *and* $l \cong_\psi \hat{l}$, *then* $\text{Free}(\hat{\sigma},\ \hat{l}, \hat{\gamma}) = \hat{\sigma}_1$ *such that* $(\gamma, \sigma_1) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)$.

Proof. By definition of Free, the ψ-congruent location will be marked as deallocated. □

Axiom 4.2. *Given a SMC[2] private pointer data structure $[\alpha, L, J, i]$ stored at memory block l and ψ-*congruent *Vanilla C pointer data structure $[1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ stored at ψ-*congruent *memory block $\hat{l}$, we consider $l, \hat{l}$ to be* equally freeable *if either:*

- *both* $\text{CheckFreeable}(\gamma,\ L,\ J) = 1$ *and* $\text{CheckFreeable}(\hat{\gamma},\ [(\hat{l}_1, \hat{\mu}_1)],\ [1]) = 1$, *or*
- *both* $\text{CheckFreeable}(\gamma,\ L,\ J) = 0$ *and* $\text{CheckFreeable}(\hat{\gamma},\ [(\hat{l}_1, \hat{\mu}_1)],\ [1]) = 0$.

Lemma 4.60. *Given type $ty, \hat{ty}$ and value $n, \hat{n}$, if* $n_1 = \text{Cast}(\text{public}, ty, n)$, $ty \cong_\psi \hat{ty}$, *and* $n = \hat{n}$ *then* $\hat{n}_1 = \text{Cast}(\text{public},\ \hat{ty},\ \hat{n})$ *such that* $n_1 = \hat{n}_1$.

Proof. By definition of algorithm Cast and Cast. □

Lemma 4.61. *Given map ψ, type $ty, \hat{ty}$ and number $n, \hat{n}$, if* $n_1 = \text{Cast}(\text{private},\ ty,\ n)$, $ty \cong_\psi \hat{ty}$, *and* $n \cong_\psi \hat{n}$ *then* $\hat{n}_1 = \text{Cast}(\text{public},\ \hat{ty},\ \hat{n})$ *such that* $n_1 \cong_\psi \hat{n}_1$.

Proof. By definition of algorithms Cast and Cast and function Erase. □

Lemma 4.62. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, type $a\ bty$, $\hat{bty}$, and location $(l_1, \mu_1), (\hat{l}_1, \hat{\mu}_1)$, if* $\text{DerefPtr}(\sigma, a\ bty, (l_1, \mu_1)) = (n, \eta)$, $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, $a\ bty \cong \hat{bty}$, *and* $(l_1, \mu_1) \cong_{\psi} (\hat{l}_1, \hat{\mu}_1)$, *then* $(\hat{n}, \hat{\eta}) = \text{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1))$ *such that* $n \cong_{\psi} \hat{n}$ *and* $\eta \cong \hat{\eta}$.

Proof. By definition of Algorithms DerefPtr, DerefPtr, and Erase. ◻

Lemma 4.63. *Given map ψ, environment $\gamma, \hat{\gamma}$, memory $\sigma, \hat{\sigma}$, type* public $bty*$, $\hat{bty}*$, *and location* (l_1, μ_1), $(\hat{l}_1, \hat{\mu}_1)$, *if* $\text{DerefPtrHLI}(\sigma, a\ bty*, (l_1, \mu_1)) = ([\alpha, L, J, i-1], \eta)$, $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, $a\ bty* \cong \hat{bty}*$, *and* $(l_1, \mu_1) \cong_{\psi} (\hat{l}_1, \hat{\mu}_1)$, *then* $([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], \hat{\eta}) = \text{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1))$ *such that* $[\alpha, L, J, i-1] \cong_{\psi} [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1]$ *and* $\eta \cong \hat{\eta}$.

Proof. By definition of Algorithms DerefPtrHLI, DerefPtrHLI, and Erase. ◻

Lemma 4.64. *Given map ψ, environment γ, $\hat{\gamma}$, memory σ_1, $\hat{\sigma}_1$, location (l, μ), $(\hat{l}, \hat{\mu})$, value n, $\hat{n}$, and type $a\ bty$, $\hat{bty}$, if* $\text{UpdateOffset}(\sigma_1, (l, \mu), n, a\ bty) = (\sigma_2, \eta)$, $(\gamma, \sigma_1) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$, $(l, \mu) \cong_{\psi} (\hat{l}, \hat{\mu})$, $n \cong_{\psi} \hat{n}$, *and* $a\ bty \cong_{\psi} \hat{bty}$, *then* $\text{UpdateOffset}(\hat{\sigma}_1, (\hat{l}, \hat{\mu}), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, \hat{\eta})$ *such that* $(\gamma, \sigma_2) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_2)$ *and* $\eta \cong \eta'$.

Proof. By definition of Algorithm UpdateOffset, UpdateOffset, and Erase, as well as Definition 4.14 and 4.4. ◻

Lemma 4.65. *Given map ψ, memory $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_1$, environment $\{\gamma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}_1$, variable list x_{list}, value $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and accumulator* acc, *if* $\{\text{InitializeVariables}(x_{list}, \gamma_1^{\text{p}}, \sigma_1^{\text{p}}, n^{\text{p}}, \text{acc} + 1) = (\gamma_2^{\text{p}}, \sigma_2^{\text{p}}, L_2^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{(\gamma_1^{\text{p}},\ \sigma_1^{\text{p}}) \cong_{\psi} (\hat{\gamma}_1,\ \hat{\sigma}_1)\}_{\text{p}=1}^{\text{q}}$, *then* $\{\gamma_2^{\text{p}} = \gamma_1^{\text{p}} :: \gamma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{\sigma_2^{\text{p}} = \sigma_1^{\text{p}} :: \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and* $\{(\gamma_2^{\text{p}},\ \sigma_2^{\text{p}}) \cong_{\psi} (\hat{\gamma}_1,\ \hat{\sigma}_1)\}_{\text{p}=1}^{\text{q}}$.

Proof. By analysis of Algorithm InitializeVariables, we can see that we do not modify any elements currently in the environment of memory, we only add new mappings for our temporary variables used to for tracking and resolution. Given this, that the original SMC^2 environment and memory pairs were ψ-congruent to the Vanilla C pair, and the definition of Algorithm Erase, we have that the updated SMC^2 environment and memory pair that is returned from Algorithm InitializeVariables is still ψ-congruent to the Vanilla C pair. ◻

Lemma 4.66. *Given statements s, if $s_1 \in s$ modifies memory at a* constant location, *then that location is dictated by a given variable x.*

Proof. Proof by case analysis of the semantics and Definitions 4.30 and 4.29, we can show that all modifications to memory that are at a *constant location* are able to be found and tracked using the variable x that refers to that location. ◻

Lemma 4.67. *Given statement s, if there exists a possible evaluation of s that results an update to memory that at a* non-constant location, *then s is found by a case in Algorithm* Extract *and the tag j returned by Algorithm* Extract *is returned as 1.*

Proof. By Definition 4.29 and case analysis of our semantics, we have statements $*x = e$ and $x[e_1] = e_2$ where $(e_1) \nvdash \gamma$ as the only statements that could possibly lead to updating memory at a *non-constant location.*
By definition of Algorithm Extract, we can see that such statements will always be found and result in the tag being set to 1. We can also show that once the tag is set to 1, it cannot be set back to 0, and therefore will be returned as 1. ◻

Lemma 4.68. *Given statement s, if any possible evaluation of $s_1 \in s$ results in an update to memory, then s_1 is found by a case in Algorithm* Extract *and either*

- s_1 *results in an update to a* non-constant location *and so the tag is set as 1,*
- s_1 *results in an update to a* constant location *dictated by x that is local, or*
- s_1 *results in an update to a* constant location *dictated by x and x is added to the variable list x_{list}.*

Proof. Proof by contradiction showing there does not exist a statement that can be evaluated via any of our rules and result in a modification in memory that is not found by one of the cases in Algorithm Extract.
By Lemma 4.67, we have bullet 1. By Lemma 4.66 and analysis of Algorithm Extract, we have bullets 2 and 3. ◻

Lemma 4.69. *Given statements s_1, s_2, and environment γ, if* $\{\text{Extract}(s_1,\ s_2, \gamma^{\text{p}}) = (x_{list}, 0)\}_{\text{p}=1}^{\text{q}}$ *then the evaluation of s_1 and s_2 can only result in updates to memory at constant locations, each dictated by variable x such that $x \in x_{list}$.*

Proof. By Lemma 4.68, we can see that as long as the tag is not returned as 1, this holds and therefore there are no updates in memory to non-local variables that will occur in either branch that cannot be caught by variable tracking. □

Lemma 4.70. *Given variable list x_{list}, environment $\{\gamma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, memory $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, value $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and accumulator* acc*, if all updates to memory in either branch will be caught by variables $x \in x_{list}$ and* $\{\text{InitializeVariables}(x_{list}, \gamma_1^{\text{p}}, \sigma_1^{\text{p}}, n^{\text{p}}, \text{acc}) = (\gamma_2^{\text{p}}, \sigma_2^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, then* $\{\forall x \in x_{list}, (\gamma_1^{\text{p}}, \sigma_1^{\text{p}}) \models (x \equiv v\_x\_orig^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall x \in x_{list}\ (\gamma_2^{\text{p}}, \sigma_2^{\text{p}}) \models (x\_else\_\text{acc} \equiv v\_x\_orig^{\text{p}})\}_{\text{p}=1}^{\text{q}}$.

Proof. By Lemma 4.69 we have all updates to memory in either branch will be caught by variables $x \in x_{list}$. By Definition 4.34 and given when Algorithm InitializeVariables is called, the current values of each x will be the original values for the variable. By definition of Algorithm InitializeVariables, we have that original values of x are stored in the temporary `else` variable corresponding to x. □

Lemma 4.71. *Given evaluation* $((1, \gamma_1^1, \sigma_{start}^1, \Delta_1^1, \text{acc}+1, s) \parallel \ldots \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_{start}^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}+1, s)) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((1, \gamma_2^1, \sigma_{end}^1, \Delta_2^1, \text{acc}+1, \text{skip}) \parallel \ldots \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_{end}^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}+1, \text{skip}))$*, if* $\{\sigma_{start}^{\text{p}} = \sigma_1^{\text{p}} :: \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *is the portion containing the temporary variables for this level of nesting designated by* acc*, then* $\{\sigma_{end}^{\text{p}} = \sigma_2^{\text{p}} :: \sigma_{temp}^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\sigma_{temp}^{\prime\text{p}} = \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Proof. Using case analysis of the semantics, it is clear that the temporary variables given used by the Private If Else rules can only be modified during execution of a Private If Else rule. It is also clear that each level of nesting will increase the accumulator acc, and given this is appended to each of the temporary variables, it is clear that there can be no overlap of temporary variable names between levels of nesting, and so the only rule that can modify the temporary variables is the one of the level at which they were created. Therefore, we have that given the execution of one of the branches, the temporary variables used for tracking remain unchanged in the execution of that branch, or $\{\sigma_{end}^{\text{p}} = \sigma_2^{\text{p}} :: \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ such that $\{\sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ remains unchanged. □

Lemma 4.72. *Given environment $\{\gamma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$,* then *branch memory $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_1$, original memory $\{\sigma_{orig}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_{orig}$, variable list x_{list}, and accumulator* acc*, if all updates to memory in either branch will be caught by variables $x \in x_{list}$,* $\{\text{RestoreVariables}(x_{list}, \gamma_1^{\text{p}}, \sigma_1^{\text{p}}, \text{acc}) = (\sigma_2^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{\forall x \in x_{list}, (\gamma_1^{\text{p}}, \sigma_1^{\text{p}}) \models (x\_else\_\text{acc} \equiv v\_x\_orig^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $(\gamma_1^{\text{p}}, \sigma_1^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$*, and* $(\gamma_1^{\text{p}}, \sigma_{orig}^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_{orig})$,
then $\{\forall x \in x_{list}, (\gamma_1^{\text{p}}, \sigma_1^{\text{p}}) \models (x \equiv v\_x\_then^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{\forall x \in x_{list}\ (\gamma_2^{\text{p}}, \sigma_2^{\text{p}}) \models (x\_then\_\text{acc} \equiv v\_x\_then^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall x \in x_{list}, (\gamma_2^{\text{p}}, \sigma_2^{\text{p}}) \models (x \equiv v\_x\_orig^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\sigma_2^{\text{p}} = \sigma_{orig}^{\text{p}} :: \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{(\gamma_1^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_{orig})\}_{\text{p}=1}^{\text{q}}$.

Proof. By Lemma 4.69 we have that all variables x that will be modified are contained in the variable list x_{list}. By Lemma 4.70, we have that all variables x within variable list x_{list} will have a `then` and `else` temporary created, and the `else` temporary stores the original value of x. By Lemma 4.71, we have that the temporary variables will remain unchanged throughout the execution of the `then` branch statement, and therefore the `else` temporary still stores the original values. Given $(\gamma_1^{\text{p}}, \sigma_{orig}^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, we have that the original memories are ψ-congruent.

By Definition 4.34 and given when Algorithm RestoreVariables is called, the current values of each x will be the `then` values for the variable. By definition of Algorithm RestoreVariables, we will store the `then` values into the `then` temporaries, and then restore the original values (stored in the `else` temporaries) back into memory for x. We will then have the resulting memory as the original memory plus our temporaries ($\{\sigma_2^{\text{p}} = \sigma_{orig}^{\text{p}} :: \sigma_{temp}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$). By definition of Algorithm Erase, we will therefore have the resulting SMC^2 environment and memory pair ψ-congruent to the original Vanilla C environment and memory pair. □

Lemma 4.73. *Given the evaluation of a Private If Else rule, if* $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n^{\text{q}}))$, $\{\text{ResolveVariables\_Retrieve}(x_{list}, \text{ acc} + 1, \gamma_1^{\text{p}}, \sigma_5^{\text{p}}) = ([(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})], n'^{\text{p}}, L_6^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, and* $\{n^{\text{p}} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$ *then* $\{n^{\text{p}} = n'\text{p}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{n'^{\text{p}} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm InitializeVariables, the results $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ from the evaluation of the private conditional e will be stored in temporary variables based on the level of nesting indicated by the accumulator acc. By Lemma 4.71, we have that these temporaries cannot be modified by the evaluation of either branch statements s_1, s_2. By definition of Algorithm RestoreVariables, we have that these temporaries cannot be modified during the evaluation of Algorithm RestoreVariables. Therefore, when we retrieve these values from memory using Algorithm ResolveVariables_Retrieve, they will be identical to the values we stored into memory using Algorithm InitializeVariables. □

Lemma 4.74. *Given environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, `else` *branch memory* $\{\sigma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$*, variable list* x_{list}*, and accumulator* acc*, if all updates to memory in either branch will be caught by variables* $x \in x_{list}$ *and* $\{\text{ResolveVariables\_Retrieve}(x_{list}, \text{ acc}+1, \gamma^{\text{p}}, \sigma^{\text{p}}) = ([(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})], n^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, and* $\forall x \in x_{list}, \text{p} \in \{1...\text{q}\}$, $(\gamma^{\text{p}}, \sigma^{\text{p}}) \models (x\_then\_\text{acc} \equiv v\_x\_then^{\text{p}})$*, then* $\{\forall x_i \in x_{list}, (\gamma^{\text{p}}, \sigma^{\text{p}}) \models (x_i \equiv v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, and* $\{\forall x_i \in x_{list}, (\gamma^{\text{p}}, \sigma^{\text{p}}) \models (x_i\_then\_\text{acc} \equiv v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$.

Proof. Given $\forall x \in x_{list}, \text{p} \in \{1...\text{q}\}, (\gamma_1^{\text{p}}, \sigma_5^{\text{p}}) \models (x\_then\_\text{acc} \equiv v\_x\_then^{\text{p}})$ by Lemma 4.71, we have that all of the `then` temporary variables currently store the result of the then branch. By definition of Algorithm ResolveVariables_Retrieve, this is what is returned for each variable x_i in value v_{ti}^{p}.

Given that we are executing Algorithm ResolveVariables_Retrieve with the resulting memory from the `else` branch, by definition of Algorithm ResolveVariables_Retrieve this is what is returned for each variable x_i in value v_{ei}^{p}. □

Lemma 4.75. *Given variable list* x_{list}*, accumulator* acc*, environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}, \hat{\gamma}$, `else` *branch memory* $\{\sigma_e^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_e$*, and values* $\{[v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}], [v_{e1}^{\text{p}}, ..., v_{em}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$*, if all updates to memory in either branch will be caught by variables* $x \in x_{list}$, $\{\text{ResolveVariables\_Store}(x_{list}, \text{ acc}, \gamma^{\text{p}}, \sigma_e^{\text{p}}, [v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}]) = (\sigma_f^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{(\gamma^{\text{p}}, \sigma_e^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_e)\}_{\text{p}=1}^{\text{q}}$, $\{\forall x_i \in x_{list}, (\gamma^{\text{p}}, \sigma_e^{\text{p}}) \models (x_i \equiv v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, and* $\{\forall i \in \{1...m\}, v_{fi}^{\text{p}} = v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, then* $\{\forall x \in x_{list}, (\gamma^{\text{p}}, \sigma_f^{\text{p}}) \models (x \equiv v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{(\gamma^{\text{p}}, \sigma_f^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_e)\}_{\text{p}=1}^{\text{q}}$.

Proof. Given that all changes were caught by variables in the variable list and that the final list of values given matches the `else` values, by definition of Algorithm ResolveVariables_Store we will iterate through the list and properly store all final values into memory for their respective variables.

Given the `else` environment and memory pairs we ψ-congruent, and that we are placing the `else` values into memory, we will have the resulting SMC[2] memory ψ-congruent to the `else` Vanilla C memory. □

Lemma 4.76. *Given variable list* x_{list}*, accumulator* acc*, environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}, \hat{\gamma}$, `else` *branch memory* $\{\sigma_e^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ `then` *branch memory* $\{\sigma_t^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_t$*, and values* $\{[v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}], [v_{t1}^{\text{p}}, ..., v_{tm}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$*, if all updates to memory in either branch will be caught by variables* $x \in x_{list}$ *and* $\{\text{ResolveVariables\_Store}(x_{list}, \text{ acc}, \gamma^{\text{p}}, \sigma_e^{\text{p}}, [v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}]) = (\sigma_f^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{(\gamma^{\text{p}}, \sigma_t^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_t)\}_{\text{p}=1}^{\text{q}}$, $\{\forall x_i \in x_{list}, (\gamma^{\text{p}}, \sigma_t^{\text{p}}) \models (x_i \equiv v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall i \in \{1...m\}, v_{fi}^{\text{p}} = v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$*, then* $\{\forall x \in x_{list}, (\gamma^{\text{p}}, \sigma_f^{\text{p}}) \models (x \equiv v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{(\gamma^{\text{p}}, \sigma_f^{\text{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_t)\}_{\text{p}=1}^{\text{q}}$.

Proof. Given that all changes were caught by variables in the variable list and that the final list of values given matches the `then` values, by definition of Algorithm ResolveVariables_Store we will iterate through the list and properly store all final values into memory for their respective variables.

Given the `then` environment and memory pairs we ψ-congruent, and that we are placing the `then` values into memory, we will have the resulting SMC[2] memory ψ-congruent to the `then` Vanilla C memory. □

Lemma 4.77. *Given statement* s_1, s_2*, environment* $\{\gamma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$*, memory* $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$*, value* $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$*, location map* $\{\Delta_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$*, and accumulator* acc*, if* $\{\text{Extract}(s_1, s_2, \gamma_1^{\text{p}}) = (x_{list}, 1)\}_{\text{p}=1}^{\text{q}}$ *and* $\{\text{Initialize}(\Delta_1^{\text{p}}, x_{list}, \gamma_1^{\text{p}}, \sigma_1^{\text{p}}, n^{\text{p}}, \text{acc})$

$= (\gamma_2^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, L^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, *then all updates to a* constant location *dictated by variable* x *will have their original value stored within location map* Δ, $\{(\gamma_2^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \models (res\_acc \equiv n^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and* $\{\sigma_2^{\mathrm{p}} = \sigma_1^{\mathrm{p}} :: \sigma_{temp1}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Proof. By Definition 4.29 and case analysis of our semantics, we have statements $*x = e$ and $x[e_1] = e_2$ where $(e_1) \nvDash \gamma$ as the only statements that could possibly lead to updating memory at a *non-constant location*. By Definition 4.30, Lemma 4.66, Lemma 4.68, and the definition of Algorithm Extract, we can see that all updates made in other semantic rules would be dictated by a variable x and added to x_{list}. By definition of Algorithm Initialize, we can see that all variables in x_{list} will have initial mappings of their location, original value, and type stored into location map $\{\Delta_1^{\mathrm{p}}[\mathrm{acc}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, as well as added the mappings to store the result of the private condition within the temporary variable *res*_acc. □

Lemma 4.78. *Given variable list* x_{list}, *location map* $\{\Delta_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, *environment* $\{\gamma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\hat{\gamma}$, *memory* $\{\sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\hat{\sigma}$, *value* $\{n^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and accumulator* acc, *if* $\{\mathrm{Initialize}(\Delta_1^{\mathrm{p}}, x_{list}, \gamma_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, n^{\mathrm{p}}, \mathrm{acc}) = (\gamma_2^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, L^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ *and* $\{(\gamma_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$, *then* $\{(\gamma_2^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Proof. By definition of Algorithm Initialize and Erase. Initialize adds a mapping for a temporary variable to store the result of the private condition, and therefore maintains ψ-congruency with the Vanilla C environment and memory pair. □

Lemma 4.79. *Given configuration* $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C)$, *if an update is made at a* non-constant location (l, μ) *during the execution of a statement* s *within a private-conditioned branch, then* $(l, \mu) \in \Delta[\mathrm{acc}]$ *such that* $\Delta[\mathrm{acc}](l, \mu) = (v_{orig}, v_{then}, j, ty)$ *and* $\Delta[\mathrm{acc}]$ *is* complete.

Proof. By Definition 4.29 and case analysis of our semantics, we have statements $*x = e$ and $x[e_1] = e_2$ where $(e_1) \nvDash \gamma$ as the only statements that could possibly lead to updating memory at a *non-constant location*. In each such rule, either DynamicUpdate is called before the update or WriteOOB is called to perform the update, and will perform the appropriate checks and add to Δ if necessary before performing the update in memory. By definitions of Algorithms DynamicUpdate and WriteOOB, we can see that we have the following cases:

- acc = 0, and we are not inside a private-conditioned branch and therefore do not need to track anything,
- the location already exists in $\Delta[\mathrm{acc}]$, and therefore already has the initial value stored and no modification of the entry will occur within Δ, or
- the location does not exist in $\Delta[\mathrm{acc}]$, and we add it with its current value as the initial value, a null `then` value, tag 0, and it's expected type, then proceed to ensure it is also tracked in outer levels of nesting (if applicable).

Given these three cases, we can see that while inside a private-conditioned branch, we are either already tracking the location or we will initialize a mapping for the location, and therefore the modification will be properly tracked within Δ. By Definition 4.31, we have that $\Delta[\mathrm{acc}]$ is *complete*. □

Lemma 4.80. *Given environment* $\{\gamma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\hat{\gamma}$, `then` *branch memory* $\{\sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\hat{\sigma}_1$, *original memory* $\{\sigma_{orig}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\hat{\sigma}_{orig}$, *location map* Δ_1, *and accumulator* acc, *if* $\{\Delta_1^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ *is* complete, $\{\mathrm{Restore}(\sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \mathrm{acc}) = (\sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, L^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, $(\gamma_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)$, *and* $(\gamma_1^{\mathrm{p}}, \sigma_{orig}^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_{orig})$, *then* $\{\Delta_2^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ *is* then-complete *and* $\{(\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}orig)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Proof. Given $\{\Delta_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *complete*, we can see that by Definition 4.31 and definition of Algorithm Restore, we will iterate through all non-local locations that were modified within `then` branch, storing the `then` value from the `then` branch memory and resetting the value in memory to be that of the original. We will set the tag to be 1 as we store each `then` value in $\{\Delta_2^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, indicating that these locations were modified within the `then` branch and ensuring that all non-local locations will be able to be properly resolved after evaluation of the `else` branch. By Definition 4.32, we have that $\{\Delta_2^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *then-complete*. □

Lemma 4.81. *Given the evaluation of a Private If Else rule, if* $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n^{\text{q}}))$, $\{\text{Extract}(s_1, s_2, \gamma^{\text{p}}) = (x_{list}, 1)\}_{\text{p}=1}^{\text{q}}$, $\{\text{Initialize}(\Delta_1^{\text{p}}, x_{list}, \gamma^{\text{p}}, \sigma_1^{\text{p}}, n^{\text{p}}, \text{acc}+1) = (\gamma_1^{\text{p}}, \sigma_2^{\text{p}}, \Delta_2^{\text{p}}, L_2^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $((1, \gamma_1^1, \sigma_2^1, \Delta_2^1, \text{acc}+1, s_1) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}+1, s_1)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_3} ((1, \gamma_2^1, \sigma_3^1, \Delta_3^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_3^{\text{q}}, \Delta_3^{\text{q}}, \text{acc} + 1, \text{skip}))$, $\{\text{Restore}(\sigma_3^{\text{p}}, \Delta_3^{\text{p}}, \text{acc} + 1) = (\sigma_4^{\text{p}}, \Delta_4^{\text{p}}, L_4^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ $((1, \gamma_1^1, \sigma_4^1, \Delta_4^1, \text{acc}+1, s_2) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_4^{\text{q}}, \Delta_4^{\text{q}}, \text{acc}+1, s_2)) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_5} ((1, \gamma_3^1, \sigma_5^1, \Delta_5^1, \text{acc}+1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_3^{\text{q}}, \sigma_5^{\text{q}}, \Delta_5^{\text{q}}, \text{acc} + 1, \text{skip}))$, $\{\text{Resolve\_Retrieve}(\gamma_1^{\text{p}}, \sigma_5^{\text{p}}, \Delta_5^{\text{p}}, \text{acc} + 1) = ([(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})], n'^{\text{p}}, L_6^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *and* $\{n^{\text{p}} \cong_\psi \hat{n}\}_{\text{p}=1}^{\text{q}}$ *then* $\{n^{\text{p}} = n'\text{p}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{n'^{\text{p}} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm Initialize, the results $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ from the evaluation of the private conditional e will be stored in a temporary variable based on the level of nesting indicated by the accumulator acc. By definition of Algorithm Restore, this temporary does not get modified. By Lemma 4.71, we have that this temporary cannot be modified by the evaluation of either branch statements s_1, s_2. Therefore, when we retrieve these values from memory using Algorithm Resolve_Retrieve, they will be identical to the values we stored into memory using Algorithm Initialize, and therefore maintain ψ-congruency with the Vanilla C value. By Definition 4.19, given these values are not locations, we have that they are congruent, $\{n'^{\text{p}} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$. □

Lemma 4.82. *Given environment* $\{\gamma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *statement* s, *memory* $\{\sigma_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *accumulator* acc, *and location map* $\{\Delta_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *if* $\{\Delta_1^{\text{p}}[\text{acc} + 1]\}_{\text{p}=1}^{\text{q}}$ *is* then-complete, $((1, \gamma_1^1, \sigma_1^1, \Delta_2^1, \text{acc} + 1, s) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_1^{\text{q}}, \Delta_2^{\text{q}}, \text{acc} + 1, s)) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((1, \gamma_2^1, \sigma_2^1, \Delta_2^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc} + 1, \text{skip}))$, *and* $\{\Delta_2^{\text{p}}[\text{acc} + 1]\}_{\text{p}=1}^{\text{q}}$ *is* complete, *then* $\{\Delta_2^{\text{p}}[\text{acc} + 1]\}_{\text{p}=1}^{\text{q}}$ *is* else-complete

Proof. This holds by Definition 4.33. □

Lemma 4.83. *Given environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *location map* $\{\Delta^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *accumulator* acc, `then` *memory* $\{\sigma_t^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and* `else` *memory* $\{\sigma_e^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *if* $\{\Delta^{\text{p}}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ *is* else-complete *and* $\{\text{Resolve\_Retrieve}(\gamma^{\text{p}}, \sigma_e^{\text{p}}, \Delta^{\text{p}}, \text{acc}) = ([(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})], n^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *then* $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, 1, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_t^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty} v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{\forall (l_i, \mu_i) = (v_{ti}^{\text{p}}, \text{NULL}, 0, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_t^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *and* $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, j, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_e^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm Resolve_Retrieve, we can see that we pull the `then` value from the location map at nesting level acc based on the tag, and the `else` value from the given `else` memory. Given $\{\Delta^{\text{p}}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ is *else-complete*, we have that all `original` and `then` values have been properly added into $\{\Delta^{\text{p}}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$. By Definitions 4.35 and 4.33, this gives us $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, 1, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_t^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty} v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ and $\{\forall (l_i, \mu_i) = (v_{ti}^{\text{p}}, \text{NULL}, 0, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_t^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$. Given we are pulling the `else` value from the given `else` memory, we have $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, j, ty_i) \in \Delta^{\text{p}}[\text{acc}], (\sigma_e^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$. This gives us that the all of our `then` values are those from the end of the `then` branch, and all of our `else` values are those from the end of the `else` branch. □

Lemma 4.84. *Given location map* $\{\Delta_1^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *accumulator* acc, *environment* $\{\gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, `else` *branch memory* $\{\sigma_e^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_e$, *and values* $\{[v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}], [v_{e1}^{\text{p}}, ..., v_{em}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$, *if* $\{\Delta_1^{\text{p}}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ *is* else-complete, $\{\text{Resolve\_Store}(\Delta_1^{\text{p}}, \sigma_e^{\text{p}}, \text{acc}, [v_{f1}^{\text{p}}, ..., v_{fm}^{\text{p}}]) = (\sigma_f^{\text{p}}, \Delta_2^{\text{p}}, L^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, $\{(\gamma^{\text{p}}, \sigma_e^{\text{p}}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_e)\}_{\text{p}=1}^{\text{q}}$, $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, j, ty_i) \in \Delta_1^{\text{p}}[\text{acc}], (\sigma_e^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *and* $\{\forall i \in \{1...m\}, v_{fi}^{\text{p}} = v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *then* $\{(\gamma^{\text{p}}, \sigma_f^{\text{p}}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_e)\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall (l_i, \mu_i) = (v_{oi}^{\text{p}}, v_{ti}^{\text{p}}, j, ty_i) \in \Delta_1^{\text{p}}[\text{acc}], (\sigma_f^{\text{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$.

Proof. Given that $\{\Delta_1^{\text{p}}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ is *else-complete*, by definition of Algorithm ResolveVariables_Store we will iterate through the list of locations and properly store all final values into memory at their respective locations.

Given the `else` environment and memory pairs we ψ-congruent, and that we are placing the `else` values into memory, we will have the resulting SMC^2 memory ψ-congruent to the `else` Vanilla C memory. □

Lemma 4.85. *Given location map* $\{\Delta_1^\text{p}\}_{\text{p}=1}^{\text{q}}$, *accumulator* acc, *environment* $\{\gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\gamma}$, `else` *branch memory* $\{\sigma_e^\text{p}\}_{\text{p}=1}^{\text{q}}$ `then` *branch memory* $\{\sigma_t^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{\sigma}_t$, *and values* $\{[v_{f1}^\text{p}, ..., v_{fm}^\text{p}], [v_{t1}^\text{p}, ..., v_{tm}^\text{p}]\}_{\text{p}=1}^{\text{q}}$, *if* $\{\Delta_1^\text{p}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ *is* else-complete, $\{\text{Resolve\_Store}(\Delta_1^\text{p}, \sigma_e^\text{p}, \text{acc}, [v_{f1}^\text{p}, ..., v_{fm}^\text{p}]) = (\sigma_f^\text{p}, \Delta_2^\text{p}, L^\text{p})\}_{\text{p}=1}^{\text{q}}$, $\{(\gamma^\text{p}, \sigma_t^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_t)\}_{\text{p}=1}^{\text{q}}$, $\{\forall (l_i, \mu_i) = (v_{oi}^\text{p}, v_{ti}^\text{p}, 1, ty_i) \in \Delta_5^\text{p}[\text{acc}], (\sigma_3^\text{p}) \models_l ((l_i, \mu_i) \equiv_{ty} v_{ti}^\text{p})\}_{\text{p}=1}^{\text{q}}$, $\{\forall (l_i, \mu_i) = (v_{ti}^\text{p}, \text{NULL}, 0, ty_i) \in \Delta_1^\text{p}[\text{acc}], (\sigma_t^\text{p}) \models_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^\text{p})\}_{\text{p}=1}^{\text{q}}$, *and* $\{\forall i \in \{1...m\}, v_{fi}^\text{p} = v_{ti}^\text{p})\}_{\text{p}=1}^{\text{q}}$, *then* $\{(\gamma^\text{p}, \sigma_f^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma}_t)\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall (l_i, \mu_i) = (v_{oi}^\text{p}, v_{ti}^\text{p}, 1, ty_i) \in \Delta_1^\text{p}[\text{acc}], (\sigma_f^\text{p}) \models_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^\text{p})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall (l_i, \mu_i) = (v_{ti}^\text{p}, \text{NULL}, 0, ty_i) \in \Delta_1^\text{p}[\text{acc}], (\sigma_f^\text{p}) \models_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^\text{p})\}_{\text{p}=1}^{\text{q}}$.

Proof. Given that $\{\Delta_1^\text{p}[\text{acc}]\}_{\text{p}=1}^{\text{q}}$ is *else-complete*, by definition of Algorithm ResolveVariables_Store we will iterate through the list of locations and properly store all final values into memory at their respective locations.

Given the `then` environment and memory pairs we ψ-congruent, and that we are placing the `then` values into memory, we will have the resulting SMC^2 memory ψ-congruent to the `then` Vanilla C memory. □

4.5 Correctness: Multiparty Computation Axioms

Axiom 4.3 (MPC$_b$). *Given bop* $\in \{+, -, \cdot, \div\}$, *values* $\{n_1^\text{p}, n_2^\text{p}, \hat{n}_1, \hat{n}_2\}_{\text{p}=1}^{\text{q}} \in \mathbb{N}$, *if* $\text{MPC}_b(bop, [n_1^1, ..., n_1^\text{q}], [n_2^1, ..., n_2^\text{q}]) = (n_3^1, ..., n_3^\text{q})$, $\{n_1^\text{p} \cong \hat{n}_1\}_{\text{p}=1}^{\text{q}}$, *and* $\{n_2^\text{p} \cong \hat{n}_2\}_{\text{p}=1}^{\text{q}}$, *then* $\{n_3^\text{p} \cong \hat{n}_3^\text{p}\}_{\text{p}=1}^{\text{q}}$ *such that* $\hat{n}_1$ *bop* $\hat{n}_2 = \hat{n}_3$.

Axiom 4.4 (MPC$_{cmp}$). *Given bop* $\in \{==, !=, <\}$, *values* $\{n_1^\text{p}, n_2^\text{p}, \hat{n}_1, \hat{n}_2\}_{\text{p}=1}^{\text{q}} \in \mathbb{N}$, *if* $\text{MPC}_{cmp}(bop, [n_1^1, ..., n_1^\text{q}], [n_2^1, ..., n_2^\text{q}]) = (n_3^1, ..., n_3^\text{q})$, $\{n_1^\text{p} \cong \hat{n}_1\}_{\text{p}=1}^{\text{q}}$, *and* $\{n_2^\text{p} \cong \hat{n}_2\}_{\text{p}=1}^{\text{q}}$, *then* $\{n_3^\text{p} \cong \hat{n}_3\}_{\text{p}=1}^{\text{q}}$ *such that* $\hat{n}_1$ *bop* $\hat{n}_2 = \hat{n}_3$.

Axiom 4.5 (MPC$_{resolve}$ False Conditional). *Given conditional result values* $\{n^\text{p}\}_{\text{p}=1}^{\text{q}}$ *and branch result values* $\{[(v_{t1}^\text{p}, v_{e1}^\text{p}), ..., (v_{tm}^\text{p}, v_{em}^\text{p})]\}_{\text{p}=1}^{\text{q}}$,
if $\text{MPC}_{resolve}([n^1, ..., n^\text{q}], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^\text{q}, v_{e1}^\text{q}), ..., (v_{tm}^\text{q}, v_{em}^\text{q})]]) = [[v_1^1, ..., v_m^1], ..., [v_1^\text{q}, ..., v_m^\text{q}]]$, *and* $\{n^\text{p} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$ *such that* $\hat{n} = 0$, *then* $\{\forall i \in \{1...m\}, v_i^\text{p} = v_{ei}^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Axiom 4.6 (MPC$_{resolve}$ True Conditional). *Given conditional result values* $\{n^\text{p}\}_{\text{p}=1}^{\text{q}}$ *and branch result values* $\{[(v_{t1}^\text{p}, v_{e1}^\text{p}), ..., (v_{tm}^\text{p}, v_{em}^\text{p})]\}_{\text{p}=1}^{\text{q}}$,
if $\text{MPC}_{resolve}([n^1, ..., n^\text{q}], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^\text{q}, v_{e1}^\text{q}), ..., (v_{tm}^\text{q}, v_{em}^\text{q})]]) = [[v_1^1, ..., v_m^1], ..., [v_1^\text{q}, ..., v_m^\text{q}]]$, *and* $\{n^\text{p} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$ *such that* $\hat{n} \neq 0$, *then* $\{\forall i \in \{1...m\}, v_i^\text{p} = v_{ti}^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Axiom 4.7 (MPC$_{ar}$). *Given array size* $\alpha, \hat{\alpha}$, *values* $\{[n_0^\text{p}, ..., n_{\alpha-1}^\text{p}]\}_{\text{p}=1}^{\text{q}}$, $\hat{n}_{\hat{i}}$, *and indices* $\{i^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{i}$, *if* $\text{MPC}_{ar}((i^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^\text{q}, [n_0^\text{q}, ..., n_{\alpha-1}^\text{q}])) = (n^1, ..., n^\text{q})$, $0 \leq \hat{i} < \hat{\alpha}$, $\alpha = \hat{\alpha}$, $\{i^\text{p} \cong \hat{i}\}_{\text{p}=1}^{\text{q}}$, *and* $\{n_{\hat{i}}^\text{p} \cong \hat{n}_{\hat{i}}\}_{\text{p}=1}^{\text{q}}$, *then* $\{n^\text{p} \cong \hat{n}_{\hat{i}}\}_{\text{p}=1}^{\text{q}}$.

Axiom 4.8 (MPC$_{aw}$). *Given array size* $\alpha, \hat{\alpha}$, *values* $\{[n_0^\text{p}, ..., n_{\alpha-1}^\text{p}]\}_{\text{p}=1}^{\text{q}}$, $\hat{n}_{\hat{i}}$, *and indices* $\{i^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{i}$, *if* $\text{MPC}_{aw}((i^1, n^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^\text{q}, n^\text{q}, [n_0^\text{q}, ..., n_{\alpha-1}^\text{q}])) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^\text{q}, ..., n_{\alpha-1}'^\text{q}])$, $0 \leq \hat{i} < \hat{\alpha}$, $\alpha = \hat{\alpha}$, $\{i^\text{p} \cong \hat{i}\}_{\text{p}=1}^{\text{q}}$, *and* $\{n_{\hat{i}}^\text{p} \cong \hat{n}_{\hat{i}}\}_{\text{p}=1}^{\text{q}}$, *then* $\{n_{\hat{i}}'^\text{p} \cong \hat{n}_{\hat{i}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall j \neq \hat{i} \in \{0...\alpha-1\} n_j^\text{p} = n_j'^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Axiom 4.9 (MPC$_u$). *Given array size* $\alpha, \hat{\alpha}$, *unary operator uop* $\in \{++\}$, *and values* $\{n_1^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\hat{n}_1$, *if* $\text{MPC}_u(++, n_1^1, ..., n_1^\text{q}) = (n_2^1, ..., n_2^\text{q})$ *and* $\{n_1^\text{p} \cong \hat{n}_1\}_{\text{p}=1}^{\text{q}}$, *then* $\{n_2^\text{p} \cong \hat{n}_2\}_{\text{p}=1}^{\text{q}}$ *such that* $\hat{n}_2 = \hat{n}_1 + 1$.

Axiom 4.10 (MPC$_{dv}$). *Given map ψ, tag lists $\{J^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and values stored at each location referred to by the given private pointer $\{[n_0^{\text{p}}, ..., n_{\alpha-1}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$,*
if $\text{MPC}_{dv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^{\text{q}}, ..., n_{\alpha-1}^{\text{q}}]], [J^1, ..., J^{\text{q}}]) = (n^1, ..., n^{\text{q}})$,
then $\{n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ is the value stored in the true location referred to by the private pointer.

Axiom 4.11 (MPC$_{dp}$). *Given map ψ, number of location α, tag lists $\{J^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and pointer data structures stored at each of the α location referred to by the given higher level private pointer $\{[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}, i-1]\}_{\text{p}=1}^{\text{q}}$,*
if $\text{MPC}_{dp}([[[\alpha_0,\ L_0^1,\ J_0^1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1]], ..., [[\alpha_0,\ L_0^{\text{q}},\ J_0^{\text{q}}], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\text{q}},\ J_{\alpha-1}^{\text{q}}]]], [J^1, ..., J^{\text{q}}]) = ([[\alpha_\alpha,\ L_\alpha^1,\ J_\alpha^1], ..., [\alpha_\alpha,\ L_\alpha^{\text{q}},\ J_\alpha^{\text{q}}]])$,
then $\{[\alpha_\alpha,\ L_\alpha^{\text{p}},\ J_\alpha^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$ properly indicates the true location of the lower level private pointer that is the true location referred to by the higher level private pointer.

Axiom 4.12 (MPC$_{wdv}$). *Given map ψ, tag lists $\{J^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and values stored at each location referred to by the given private pointer $\{[n_0^{\text{p}}, ..., n_{\alpha-1}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$,*
if $\text{MPC}_{wdv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^{\text{q}}, ..., n_{\alpha-1}^{\text{q}}]], [n^1, ..., n^{\text{q}}], [J^1, ..., J^{\text{q}}]) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^{\text{q}}, ..., n_{\alpha-1}'^{\text{q}}])$ *and* $\{J^{\text{p}}[j] = \text{encrypt}(1)\}_{\text{p}=1}^{\text{q}}$, *then* $\{n_j'^{\text{p}} = n^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall i \neq j \in \{0...\alpha-1\}\ n_i'^{\text{p}} = n_i^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Axiom 4.13 (MPC$_{wdp}$). *Given map ψ, number of location α, tag lists $\{J^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and pointer data structures stored at each of the α location referred to by the given higher level private pointer $\{[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}, i-1]\}_{\text{p}=1}^{\text{q}}$,*
if $\text{MPC}_{wdp}([[1, [(l_e^1, \mu_e^1)], [1], i-1], [\alpha_0,\ L_0^1,\ J_0^1, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1, i-1]], ..., [[1, [(l_e^{\text{q}}, \mu_e^{\text{q}})], [1], i-1], [\alpha_0,\ L_0^{\text{q}},\ J_0^{\text{q}}, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\text{q}},\ J_{\alpha-1}^{\text{q}}, i-1]], [J^1, ..., J^{\text{q}}]) = [[[\alpha_0',\ L_0'^1,\ J_0'^1, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^1,\ J_{\alpha-1}'^1, i-1]], ..., [\alpha_0',\ L_0'^{\text{q}},\ J_0'^{\text{q}}, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^{\text{q}},\ J_{\alpha-1}'^{\text{q}}, i-1]]$ *and* $\{J^{\text{p}}[j] = \text{encrypt}(1)\}_{\text{p}=1}^{\text{q}}$,
then $[\alpha_j,\ L_j^{\text{p}},\ J_j^{\text{p}}]$ has the true location set as $(l_e^{\text{p}}, \mu_e^{\text{p}})$ and $\forall i \neq j \in \{0...\alpha-1\}[\alpha_i,\ L_i^{\text{p}},\ J_i^{\text{p}}]$, the true location remains the same as what it originally was.

Axiom 4.14 (MPC$_{free}$). *Given map ψ, byte representations $\{[\omega_0^{\text{p}}, ..., \omega_{\alpha-1}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$ and tag lists $\{J^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ such that* $\text{MPC}_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^{\text{q}}, ..., \omega_{\alpha-1}^{\text{q}}]], [J^1, ...J^{\text{q}}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^{\text{q}}, ..., \omega_{\alpha-1}'^{\text{q}}]], [J'^1, ..., J'^{\text{q}}])$,
if $\{J'^{\text{p}}[0] = \text{encrypt}(1)\}_{\text{p}=1}^{\text{q}}$, $\{J'^{\text{p}}[j] = \text{encrypt}(1)\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall i \neq j \in \{1...\alpha-1\} J'^{\text{p}}[i] = \text{encrypt}(0)\}_{\text{p}=1}^{\text{q}}$
then $\{\omega_0^{\text{p}} = \omega_j'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{\omega_j^{\text{p}} = \omega_0'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and* $\{\forall i \neq j \in \{1...\alpha-1\}, \omega_i^{\text{p}} = \omega_i'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$
otherwise if $\{J'^{\text{p}}[0] = \text{encrypt}(1)\}_{\text{p}=1}^{\text{q}}$ *and* $\{\forall i \in \{1...\alpha-1\} J'^{\text{p}}[i] = \text{encrypt}(0)\}_{\text{p}=1}^{\text{q}}$
then $\{\forall i \in \{0...\alpha-1\}, \omega_i^{\text{p}} = \omega_i'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

4.6 Confluence

Definition 4.36 ($v^1 \sim v^2$). Two values are *corresponding*, in symbols $v^1 \sim v^2$, if and only if either both v^1, v^2 are public (including locations) and $v^1 = v^2$, or v^1, v^2 are private and $\text{Erase}(v^1) = \text{Erase}(v^2)$.

Definition 4.37 ($\gamma^1 \sim \gamma^2$). Two environments are *corresponding*, in symbols $\gamma^1 \sim \gamma^2$, if and only if $\gamma^1 = \gamma^2$.

Definition 4.38 ($\omega^1 \sim \omega^2$). Two bytes are *corresponding*, in symbols $\omega^1 \sim \omega^2$, if and only if they are of the same type, and when decoded to values, $v^1 \sim v^2$.

Definition 4.39 ($\sigma^1 \sim \sigma^2$). Two memories are *corresponding*, in symbols $\sigma^1 \sim \sigma^2$, if and only if $\forall l_1 \notin \sigma^1, l_1 \notin \sigma^2$, and $\forall l \in \sigma^1$ such that $\sigma^1(l) = (\omega^1, ty^1, \alpha^1, \text{PermL}^1)$, $l \in \sigma^2$ such that $\sigma^2(l) = (\omega^2, ty^2, \alpha^2, \text{PermL}^2)$ and $\omega^1 \sim \omega^2$, $ty^1 = ty^2$, $\alpha^1 = \alpha^2$, and $\text{PermL}^1 = \text{PermL}^2$.

Definition 4.40 ($\Delta^1 \sim \Delta^2$). Two location maps are *corresponding*, in symbols $\Delta^1 \sim \Delta^2$, if and only if $\forall (l_1, \mu_1) \notin \Delta^1, (l_1, \mu_1) \notin \Delta^2$, and $\forall (l, \mu) \in \Delta^1$ such that $(l, \mu) \to (v_1^1, v_2^1, j^1, ty^1)$, $(l, \mu) \in \Delta^2$ such that $(l, \mu) \to (v_1^2, v_2^2, j^2, ty^2)$ and $v_1^1 \sim v_1^2, v_2^1 \sim v_2^2, j^1 \sim j^2$, and $ty^1 \sim ty^2$.

Definition 4.41 ($\text{acc}^1 \sim \text{acc}^2$). Two accumulators are *corresponding*, in symbols $\text{acc}^1 \sim \text{acc}^2$, if and only if $\text{acc}^1 = \text{acc}^2$.

Definition 4.42 ($C^1 \sim C^2$). Two configurations are *corresponding*, in symbols $C^1 \sim C^2$ or $(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s^1) \sim (2, \gamma^2, \sigma^2, \Delta^2, \mathrm{acc}^2, s^2)$, if and only if $\gamma^1 = \gamma^2$, $\sigma^1 \sim \sigma^2$, $\Delta^1 \sim \Delta^2$, $\mathrm{acc}^1 = \mathrm{acc}^2$, and $s^1 = s^2$.

Lemma 4.86 ($C^1 \sim C^2 \implies C^1 \cong_\psi \hat{C} \wedge C^2 \cong_\psi \hat{C}$). *Given two configurations C^1, C^2 such that $C^1 = (1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s^1)$ and $C^2 = (2, \gamma^2, \sigma^2, \Delta^2, \mathrm{acc}^2, s^2)$ and ψ, if $C^1 \sim C^2$ then $\{C^\mathrm{p} \cong_\psi (\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, s)\}_{\mathrm{p}=1}^2$.*

Proof.

Proof Sketch:

Using the definition of Erase and Definition 4.42, there is only one possible Vanilla C configuration $\hat{C}$ (modulo party ID) that can be obtained from both Erase(C^1) and Erase(C^2). □

Lemma 4.87 (Unique party-wise transitions). *Given $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C)$ if $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma_1, \sigma_1, \Delta_1, \mathrm{acc}, v) \parallel C_1)$ then there exists no other rule by which $(\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s)$ can step.*

Proof.

Proof Sketch:

By induction on $(\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, s)$. We verify that for every configuration, given s, acc, and stored type information, there is only one corresponding semantic rule. □

Theorem 4.1 (Confluence). *Given $C^1 \parallel ... \parallel C^\mathrm{q}$ such that $\{C^1 \sim C^\mathrm{p}\}_{\mathrm{p}=1}^{\mathrm{q}}$*
if $(C^1 \parallel ... \parallel C^\mathrm{q}) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} (C_1^1 \parallel ... \parallel C_1^\mathrm{q})$ such that $\exists \mathrm{p} \in \{1...\mathrm{q}\} C_1^1 \nsim C_1^\mathrm{p}$,
then $\exists\, (C_1^1 \parallel ... \parallel C_1^\mathrm{q}) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} (C_2^1 \parallel ... \parallel C_2^\mathrm{q})$
such that $\{C_2^1 \sim C_2^\mathrm{p}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{(\mathcal{L}_1^1 :: \mathcal{L}_2^1) = (\mathcal{L}_1^\mathrm{p} :: \mathcal{L}_2^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$, and $\{(\mathcal{D}_1^1 :: \mathcal{D}_2^1) = (\mathcal{D}_1^\mathrm{p} :: \mathcal{D}_2^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Proof.

Proof Sketch:

By Lemma 4.87, we have that there is only one possible execution trace for any given party based on the starting configuration.

By definition of $\{C^1 \sim C^\mathrm{p}\}_{\mathrm{p}=1}^{\mathrm{q}}$, we have that the starting states of all parties are corresponding, with identical statements.

Therefore, all parties must follow the same execution trace and will eventually reach another set of corresponding states. □

4.7 Multiparty Correctness Theorem

Axiom 4.15. *For purposes of correctness, we assume all parties are executing a program s from initial state $(\mathrm{p}, [\,], [\,], [\,], 0, s)$ with congruent input data. We assume that s does not contain hard-coded locations, has* well-aligned *out-of-bounds memory accesses where private indices are not used and no out-of-bounds accesses where private indices are used, and type-casts for private locations match the intended type that the location was allocated for.*

Theorem 4.2 (Semantic Correctness).
For every configuration $\{(\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}^\mathrm{p}, s^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{(\mathrm{p}, \hat{\gamma}^\mathrm{p}, \hat{\sigma}^\mathrm{p}, \square, \square, \hat{s}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and map ψ
such that $\{(\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}^\mathrm{p}, s^\mathrm{p}) \cong_\psi (\mathrm{p}, \hat{\gamma}^\mathrm{p}, \hat{\sigma}^\mathrm{p}, \square, \square, \hat{s}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$,
if $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s^1) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}^\mathrm{q}, s^\mathrm{q}))$
$\Downarrow_{\mathcal{D}}^{\mathcal{L}} ((1, \gamma_1^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}_1^1, v^1) \parallel ... \parallel (\mathrm{q}, \gamma_1^\mathrm{q}, \sigma_1^\mathrm{q}, \Delta_1^\mathrm{q}, \mathrm{acc}_1^\mathrm{q}, v^\mathrm{q}))$
for codes $\mathcal{D} \in SmcC$, then there exists a derivation
$\Sigma \triangleright ((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \hat{s}^1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}^\mathrm{q}, \hat{\sigma}^\mathrm{q}, \square, \square, \hat{s}^\mathrm{q}))$
$\Downarrow_{\hat{\mathcal{D}}} ((1, \hat{\gamma}_1^1, \hat{\sigma}_1^1, \square, \square, \hat{v}^1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}_1^\mathrm{q}, \hat{\sigma}_1^\mathrm{q}, \square, \square, \hat{v}^\mathrm{q}))$
for codes $\hat{\mathcal{D}} \in VanC$ and a map ψ_1 such that
$\mathcal{D} \cong \hat{\mathcal{D}}$, $\{(\mathrm{p}, \gamma_1^\mathrm{p}, \sigma_1^\mathrm{p}, \Delta_1^\mathrm{p}, \mathrm{acc}_1^\mathrm{p}, v^\mathrm{p}) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma}_1^\mathrm{p}, \hat{\sigma}_1^\mathrm{p}, \square, \square, \hat{v}^\mathrm{p})\}_{\mathrm{p}=1}^{\mathrm{q}}$, and $\Pi \cong_{\psi_1} \Sigma$.

Proof. *Proof Sketch:* By induction over all SMC^2 semantic rules.

The bulk of the complexity of this proof lies with rules pertaining to Private If Else, handling of pointers, and freeing of memory. We first provide a brief overview of the intuition for the simpler cases and then dive deeper into the details for the more complex cases. Full proofs are available in our artifact submission.

For the rules evaluating over public data, correctness follows simply as the Vanilla C and SMC^2 rules for public data are nearly identical. For all the semantic rules that use general helper algorithms (i.e., algorithms in common to both Vanilla C and SMC^2), we also reason about the correctness of the helper algorithms, comparing the Vanilla C version and the SMC^{2}version. Correctness over such algorithms is easily proven, as these algorithms are nearly identical, differing on privacy labels as we do not have private data in Vanilla C.

For all SMC^2 multiparty semantic rules, we relate them to the multiparty versions of the Vanilla C rules. To reason about the multiparty protocols, we leverage Axioms, such as Axiom 4.3, to prove these rules correct. These Axioms should be proven correct by a library developer to ensure the completeness of the formal model. The correctness of most multiparty semantic rules follows easily, with Multiparty Private Free being an exception. For this rule, we also must reason about our helper algorithms that are specific to the SMC^2 semantics (e.g., UpdateBytesFree, UpdatePointerLocations). We leverage the correctness of the behavior of the multiparty protocol MPC$_{free}$, to show that correctness of these algorithms follows due to the deterministic definitions of the algorithms. In this case, we must also show that the locations that are swapped within this rule (which is done to hide the true location) are deterministic based on our memory model definition. We use ψ to map the swapped locations, enabling us to show that, if these swaps were reversed, we would once again have memories that are directly congruent. This concept of locations being ψ-congruent is particularly necessary when reasoning about pointers in other rule cases. For all the rules using private pointers, we will rely upon the pointer data structure containing a set of locations and their associated tags, only one of which being the true location. With this proven to be the case, it is then clear that the true location indicated within the private pointer's data structure in SMC^2 will be ψ-congruent with the location given by the pointer data structure in Vanilla C. In our proof, we make the assumption that locations are not hard-coded, as hard-coded locations would lead to potentially differing results between Vanilla C and SMC^2 execution due to the behavior of `pfree`. Additionally, given the distributed nature of the SMC^2, it would not make sense to allow hard-coded locations, as a single program will be executed on several different machines.

For rule Private Malloc, we must relate this rule to the sequence of Vanilla C rules for Malloc, Multiplication, and Size Of Type. This is due to the definition of `pmalloc` as a helper that allows the user to write programs without knowing the size of private types. This case follows from the definition of translating the SMC^2 program to a Vanilla C program, $\textbf{Erase}(\text{pmalloc}(e,\ ty) = (\text{malloc}(\text{sizeof}(\textbf{Erase}(ty)) \cdot \textbf{Erase}(e))))$.

For the Private If Else rules, we must reason that our end results in memory after executing both branches and resolving correctly match the end result of having only executed the intended branch. The cases for both of these rules will have two subcases - one for the conditional being true, and the other for false. To obtain correctness, we use multiparty versions of the if else true and false rules that execute both branches - this allows us to reason that both branches will evaluate properly, and that we will obtain the correct ending state once completed. For both rules, we must first show that Extract will correctly find all non-local variables that are modified within both branches, including non-assignment modifications such as use of the pre-increment operator $++x$, and that all such modified variables will be added to the list (excluding pointers modified exclusively by pointer dereference write statements). We must also show that it will correctly find and tag if a pointer dereference write statement was found. These properties follow deterministically from the definition of the algorithm.

For rule Private If Else Variable Tracking, we will leverage the correctness of Extract, and that if Extract returns the tag 0, no pointer dereference writes were found. We then reason that InitializeVariables will correctly create the assignment statements for our temporary variables, and that the original values for each of the modified variables will be stored into the `else` temporary variables. The temporaries being stored into memory correctly through the evaluation of these statements follows by induction. Next we have the evaluation of the `then` branch, which will result in the values that are correct for if the condition had been true - this holds by induction. We then proceed to reason that RestoreVariables will properly create the statements to store the ending results of the `then` branch into the `then` temporary variables, and restore all of the original values from the `else` variables (the original values being correctly stored follows from InitializeVariables and the evaluation of it's statements). The correct evaluation of the this set of statements follows by induction. Next we have the evaluation of the `else` branch, which will result in the values that are correct for if the condition had been false - this holds by induction and the values having been restored to the original values properly. We will then reason about the correctness of the statements created by ResolveVariables. These statements must be set up to correctly take the information from the then temporary variable, the temporary variable for the condition for the branch, and the ending result for all variables from the else branch. For the resolution of pointers, we insert a call for a resolution function (resolve), because the resolution of pointer data is more involved. The evaluation of this function is shown in rule Multiparty Resolve Pointer Locations. By proving that this rule will correctly resolve the true locations for pointers, we will then have that the statements created by ResolveVariables will appropriately resolve all pointers.

For rule Private If Else Location Tracking, the structure of the case is similar to the rule for variable tracking, but with a few differences we will discuss here. For this rule, we will need to reason about DynamicUpdate, and that we will catch all modifications by pointer dereference writes and properly add them to Δ if the location being modified is not already tracked. If a new mapping is added, we store the current value in v_{orig} (as this location has not yet been modified) and the tag has to be set to 0. This behavior will be used to ensure the correctness during resolution. For Initialize, we must reason that we correctly initialize the map Δ with all of the locations we found within Extract to be modified by means other than pointer dereference writes and store their original values in v_{orig}. Then we can evaluate the then branch, which will result in the values that are correct for if the condition had been true - this holds by induction. For Restore, we reason that we properly store the results of the then branch, and update the tag for the location to signify that we should use v_{then} instead of v_{orig}. We will then restore the original values, leveraging the correctness of Initialize to prove this will happen correctly. Then we can evaluate the else branch, which will result in the values that are correct for if the condition had been false - this holds by induction. For Resolve, we reason that we will create the appropriate resolution statements to be executed. For the then result, these statements must use the value stored in v_{orig} if the tag is set to 0 (this occurs if the first modification to the location was a pointer dereference write within the else branch), and the value stored in v_{then} if the tag is set to 1. We prove this to be the correct then result through the correctness of DynamicUpdate and Restore. The else result must use the current value for that location in memory, which is proven to be the correct else result through the correctness of Initialize and Resolve. In this way, we can prove the correctness the contents of the statements created by Resolve, and then the correctness of the evaluation of the statements created by Restore will hold as we discussed for with those created by ResolveVariables for Private If Else Variable tracking. □

Proof.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1\ bop\ e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e_1\ bop\ e_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\text{ALL}, [mpb])}$
$((1, \gamma^1, \sigma^1_2, \Delta^1_2, \text{acc}, n^1_3) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_2, \Delta^\text{q}_2, \text{acc}, n^\text{q}_3))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1\ bop\ e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e_1\ bop\ e_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\text{ALL}, [mpb])}$ $((1, \gamma^1, \sigma^1_2, \Delta^1_2, \text{acc}, n^1_3) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_2, \Delta^\text{q}_2, \text{acc}, n^\text{q}_3))$, by SMC^2 rule Multiparty Binary Operation we have $\{(e_1, e_2) \vdash \gamma^\text{p}\}^\text{q}_{\text{p}=1}$, (B) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e_1)) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, n^1_1) \parallel ... \parallel$ $(\text{q}, \gamma^\text{q}, \sigma^\text{q}_1, \Delta^\text{q}_1, \text{acc}, n^\text{q}_1))$, (C) $((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_1, \Delta^\text{q}_1, \text{acc}, e_2)) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((1, \gamma^1, \sigma^1_2, \Delta^1_2, \text{acc}, n^1_2)$ $\parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_2, \Delta^\text{q}_2, \text{acc}, n^\text{q}_2))$, (D) $\text{MPC}_b(bop, [n^1_1, ..., n^\text{q}_1], [n^1_2, ..., n^\text{q}_2]) = (n^1_3, ..., n^\text{q}_3)$, and (E) $bop \in \{\cdot, +, -, \div\}$.

Given (A), $((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \hat{e}_1\, bop\, \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}^\text{q}, \hat{\sigma}^\text{q}, \square, \square, \hat{e}_1\, bop\, \hat{e}_2))$ and ψ such that $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc},$ $e_1\ bop\ e_2) \cong_\psi (\text{p}, \hat{\gamma}^\text{p}, \hat{\sigma}^\text{p}, \square, \square, \hat{e}_1\, bop\, \hat{e}_2)\}^\text{q}_{\text{p}=1}$, by Definition 4.22 we have $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}^\text{p}, \hat{\sigma}^\text{p})\}^\text{q}_{\text{p}=1}$ and (F) $e_1\, bop\, e_2 \cong_\psi \hat{e}_1\, bop\, \hat{e}_2$. By Definition 4.20 we have $bop = bop$, (G) $e_1 \cong_\psi \hat{e}_1$ and (H) $e_2 \cong_\psi \hat{e}_2$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1\ bop\ e_2) \sim (\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, e_1\ bop\ e_2)\}^\text{q}_{\text{p}=1}$. By Lemma 4.86, we have $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, e_1\ bop\ e_2) \cong_\psi (\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1\, bop\, \hat{e}_2)\}^\text{q}_{\text{p}=1}$. and therefore (I) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1\, bop\, \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1\, bop\, \hat{e}_2))$. By Definition 4.22 we have (J) $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}^\text{q}_{\text{p}=1}$.

Given (B), (J), (G), and ψ, by Lemma 4.28 we have $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1))$ such that $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, e_1) \cong_\psi (\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1)\}^\text{q}_{\text{p}=1}$. By the inductive hypothesis, we have (K) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1)$

$\| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1)) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1))$ and ψ_1 such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \mathrm{acc}, n_1^{\mathrm{p}}) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. By Definition 4.22 we have (L) $\{(\gamma^{\mathrm{p}},\ \sigma_1^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and $\{n_1^{\mathrm{p}} \cong_{\psi_1} \hat{n}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Definition 4.19 we have (M) $\{n_1^{\mathrm{p}} \cong \hat{n}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (H), by Lemma 4.7 we have $e_2 \cong_{\psi_1} \hat{e}_2$. Therefore, given (C), (L), and ψ_1, by Lemma 4.28 we have $((1, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{e}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{e}_2))$ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \mathrm{acc}, e_2) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{e}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By the inductive hypothesis, we have (N) $((1, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{e}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2))$ and ψ_2 such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, \mathrm{acc}, n_2^{\mathrm{p}}) \cong_{\psi_2} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22 we have (O) $\{(\gamma^{\mathrm{p}},\ \sigma_2^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and $\{n_2^{\mathrm{p}} \cong_{\psi_2} \hat{n}_2\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Definition 4.19 we have (P) $\{n_2^{\mathrm{p}} \cong \hat{n}_2\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (D), (M), and (P), by Axiom 4.3 we have (Q) $\{n_3^{\mathrm{p}} \cong \hat{n}_3\}_{\mathrm{p}=1}^{\mathrm{q}}$ such that (R) $\hat{n}_1\ bop\ \hat{n}_2 = \hat{n}_3$.

Given (I), (K), (N), (R), (E) and $bop = bop$, we have $\Sigma \triangleright ((1, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1\ bop\ \hat{e}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1\ bop\ \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\mathrm{ALL}, [\hat{mpb}])]} ((1, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_3) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_3))$ by Vanilla C rule Multiparty Binary Operation.

Given (O) and (Q), by Definition 4.22 we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, \mathrm{acc}, n_3^{\mathrm{p}}) \cong_{\psi_2} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_2,\ \square,\square,\ \hat{n}_3)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Definition 4.23 we have $mpb \cong \hat{mpb}$. Given $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$, $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$, $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpb])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\mathrm{ALL}, [\hat{mpb}])]$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpb]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\mathrm{ALL}, [\hat{mpb}])]$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, e_1\ bop\ e_2) \| \dots \| \; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, e_1\ bop\ e_2)) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpcmp])} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}^1, n_3^1) \| \dots \| \; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, n_3^{\mathrm{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, e_1\ bop\ e_2) \| \dots \| \; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, e_1\ bop\ e_2)) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpcmp])} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}^1, n_3^1) \| \dots \| \; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, n_3^{\mathrm{q}}))$ by SMC^2 rule Multiparty Comparison Operation, we have $\{(e_1, e_2) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e_1) \| \dots \| (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e_1)) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n_1^1) \| \dots \| (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n_1^{\mathrm{q}}))$, (C) $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, e_2) \| \dots \| (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, e_2)) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, n_2^1) \| \dots \| \; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, n_2^{\mathrm{q}}))$, (D) $\mathrm{MPC}_{cmp}(bop, [n_1^1, \dots, n_1^{\mathrm{q}}], [n_2^1, \dots, n_2^{\mathrm{q}}]) = (n_3^1, \dots, n_3^{\mathrm{q}})$, and (E) $bop \in \{==, !=, <\}$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma}^{\mathrm{q}},\ \hat{\sigma}^{\mathrm{q}},\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e_1\ bop\ e_2) \cong_{\psi} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}},\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (F) $e_1 bop\ e_2 \cong_{\psi} \hat{e}_1 bop\ \hat{e}_2$. Given (F), by Definition 4.20 we have (G) $e_1 \cong_{\psi} \hat{e}_1$, (H) $e_2 \cong_{\psi} \hat{e}_2$, and $bop = bop$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e_1\ bop\ e_2) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e_1\ bop\ e_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e_1\ bop\ e_2) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and therefore (I) $((1, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1 bop\ \hat{e}_2))$. By Definition 4.22 we have (J) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B), (J), (G), and ψ, by Lemma 4.28 we have $((1, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1))$ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e_1) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By the inductive hypothesis, we have (K) $((1, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square,\square,\ \hat{e}_1)) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1) \| \dots \| \; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_1,\ \square,\square,\ \hat{n}_1))$ and ψ_1 such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}},$

acc, $n_1^{\text{p}}) \cong_{\psi_1} (\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1)\}_{\text{p}=1}^{\text{q}}$ and $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. By Definition 4.22 we have (L) $\{(\gamma^{\text{p}}, \sigma_1^{\text{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}_{\text{p}=1}^{\text{q}}$ and $\{n_1^{\text{p}} \cong_{\psi_1} \hat{n}_1\}_{\text{p}=1}^{\text{q}}$. By Definition 4.19 we have (M) $\{n_1^{\text{p}} \cong \hat{n}_1\}_{\text{p}=1}^{\text{q}}$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (H), by Lemma 4.7 we have $e_2 \cong_{\psi_1} \hat{e}_2$. Therefore, given (C), (E), (L), and ψ_1, by Lemma 4.28 we have $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2))$ such that $\{(\text{p}, \gamma^{\text{p}}, \sigma_1^{\text{p}}, \Delta_1^{\text{p}}, \text{acc}, e_2) \cong_{\psi} (\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2)\}_{\text{p}=1}^{\text{q}}$. By the inductive hypothesis, we have (N) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2))$ and ψ_2 such that $\{(\text{p}, \gamma^{\text{p}}, \sigma_2^{\text{p}}, \Delta_2^{\text{p}}, \text{acc}, n_2^{\text{p}}) \cong_{\psi_2} (\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2)\}_{\text{p}=1}^{\text{q}}$ and $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22 we have (O) $\{(\gamma^{\text{p}}, \sigma_2^{\text{p}}) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$ and $\{n_2^{\text{p}} \cong_{\psi_2} \hat{n}_2\}_{\text{p}=1}^{\text{q}}$. By Definition 4.19 we have (P) $\{n_2^{\text{p}} \cong \hat{n}_2\}_{\text{p}=1}^{\text{q}}$.

Given (D), (M), and (P), by Axiom 4.4 we have (Q) $\{n_3^{\text{p}} \cong \hat{n}_3\}_{\text{p}=1}^{\text{q}}$ such that (R) $(\hat{n}_1 \; bop \; \hat{n}_2) = \hat{n}_3$.

Subcase (S1) $\hat{n}_3 = 1$

Given (I), (K), (N), (R), (S1), (E), and $bop = bop$, we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 \; bop \; \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 \; bop \; \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::[(\text{ALL}, [\hat{mpcmpt}])]} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3))$ by Vanilla C rule Multiparty Comparison True Operation.

Given (O) and (Q), by Definition 4.22 we have $\{(\text{p}, \gamma^{\text{p}}, \sigma_2^{\text{p}}, \Delta_2^{\text{p}}, \text{acc}, n_3^{\text{p}}) \cong_{\psi} (\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3)\}_{\text{p}=1}^{\text{q}}$.
By Definition 4.23 we have $mpcmp \cong \hat{mpcmpt}$.
Given $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$, $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$, $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpcmp])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\text{ALL}, [\hat{mpcmpt}])]$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpcmp]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\text{ALL}, [\hat{mpcmpt}])]$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Subcase (S2) $\hat{n}_3 = 0$

Given (I), (K), (N), (R), (S2), (E), and $bop = bop$, we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 \; bop \; \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 \; bop \; \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::[(\text{ALL}, [\hat{mpcmpf}])]} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3))$ by Vanilla C rule Multiparty Comparison False Operation.

Given (O) and (Q), by Definition 4.22 we have $\{(\text{p}, \gamma^{\text{p}}, \sigma_2^{\text{p}}, \Delta_2^{\text{p}}, \text{acc}, n_3^{\text{p}}) \cong_{\psi} (\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3)\}_{\text{p}=1}^{\text{q}}$.
By Definition 4.23 we have $mpcmp \cong \hat{mpcmpf}$.
Given $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$, $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$, $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpcmp])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\text{ALL}, [\hat{mpcmpf}])]$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpcmp]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: [(\text{ALL}, [\hat{mpcmpf}])]$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{if } (e) \; s_1 \text{ else } s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$

Given $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{if } (e) \; s_1 \text{ else } s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$ by SMC[2] rule Public If Else True, we have $(e) \nvdash \gamma$, (A) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (B) $n \neq 0$, and (C) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, s_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_1, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$.

Given $(\square, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{if}(\hat{e}) \; \hat{s}_1 \text{ else } \hat{s}_2)$ and ψ such that $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{if } (e) \; s_1 \text{ else } s_2) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

if$(\hat{e})$ $\hat{s}_1$ else $\hat{s}_2) \parallel \hat{C})$, by Definition 4.22 we have (D) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$ and if (e) s_1 else $s_2 \cong_{\psi}$ if $(\hat{e})$ $\hat{s}_1$ else $\hat{s}_2$ and (E) $C \cong_{\psi} \hat{C}$. By Definition 4.20, we have (F) $e \cong_{\psi} \hat{e}$, (G) $s_1 \cong_{\psi} \hat{s}_1$, and (H) $s_2 \cong_{\psi} \hat{s}_2$.

Given (D), ψ, (E), and (F), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (A), by the inductive hypothesis we have (I) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and ψ_1 such that $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (J) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. By Definition 4.22 we have (K) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, (L) $C_1 \cong_{\psi_1} \hat{C}_1$, and $n \cong_{\psi_1} \hat{n}$. By Definition 4.19 we have $n \cong \hat{n}$.

Given $(e) \nvdash \gamma$, we have $(n) \nvdash \gamma$ and therefore $n = \hat{n}$. Given (B), we have (M) $\hat{n}\ \neq\ 0$.

Given Axiom 4.15, we have $(l, \mu) \notin s_1$. Given (G), by Lemma 4.7 we have $s_1 \cong_{\psi_1} \hat{s}_1$. Therefore, given (K), ψ_1, and (L), by Lemma 4.2 we have $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ s_1) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel \hat{C}_1)$. Given (C), by the inductive hypothesis, we have (N) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and ψ_2 such that $((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and (O) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22, we have $(\gamma_1, \sigma_2) \cong_{\psi_2} (\hat{\gamma}_1,\ \hat{\sigma}_2)$ and (P) $C_2 \cong_{\psi_2} \hat{C}_2$. By Lemma 4.9, we have (Q) $(\gamma,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel \hat{C})$ and (I), (M), and (N), we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{iet}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ by Vanilla C rule If Else True.

Given (P) and (Q), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. By Definition 4.23 we have $iet \cong \hat{iet}$. Given (J), (O), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [iet])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{iet}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [iet]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{iet}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p},\ \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ief])} ((\mathrm{p},\ \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [iet])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \Downarrow^{\mathcal{L}}_{\mathcal{D} :: (\mathrm{p}, [wle])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \Downarrow^{\mathcal{L}}_{\mathcal{D} :: (\mathrm{p}, [wle])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC^2 rule While End, we have $(e) \nvdash \gamma$, (A) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C), \Downarrow^{\mathcal{L}}_{\mathcal{D}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$, (B) $n\ ==\ 0$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\ \hat{s}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

while$(\hat{e})\ \hat{s}) \parallel \hat{C})$, by Definition 4.22 we have (C) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (D) $C \cong_{\psi} \hat{C}$ and while $(e)\ s \cong_{\psi}$ while $(\hat{e})\ \hat{s}$. By Definition 4.20 we have (E) $e \cong_{\psi} \hat{e}$ and $s \cong_{\psi} \hat{s}$

Given (C), (D), (E), and ψ, by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. By the inductive hypothesis, we have (F) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C})$ such that $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \cong_{\psi_1} ((\square, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (G) $\mathcal{D} \cong \hat{\mathcal{D}}$.

By Definition 4.22 we have (H) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$ and $n \cong_{\psi_1} \hat{n}$. By Definition 4.19 we have $n \cong \hat{n}$. Given $(e) \nvDash \gamma$, we have $(n) \nvDash \gamma$ and therefore (I) $n = \hat{n}$.

Given (B) and (I), we have (J) $\hat{n}\ =\ 0$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\ \hat{s}) \parallel \hat{C})$, (F), and (J), we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\ \hat{s}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p}, [\widehat{wle}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$ by Vanilla C rule While End.

Given (H), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $wle \cong \widehat{wle}$. Given (G), $\mathcal{D}_1 :: (\mathrm{p}, [wle])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{wle}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [wle]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\widehat{wle}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [wlc])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C_2)$

Given $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [wlc])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C_2)$ by SMC^2 rule While Continue, we have $(e) \nvDash \gamma$, (A) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)$, (B) $n\ \neq\ 0$, and (C) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ s) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$.

Given (D) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\ \hat{s}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{while}\ (e)\ s) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\ \hat{s}) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, $C \cong_{\psi} \hat{C}$, and (F) while $(e)\ s \cong_{\psi}$ while $(\hat{e})\ \hat{s}$. By Definition 4.20 we have (G) $e \cong_{\psi} \hat{e}$ and (H) $s \cong_{\psi} \hat{s}$.

Given (D), ψ, (E), $C \cong_{\psi} \hat{C}$, and (H), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (A), by the inductive hypothesis we have (I) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and ψ_1 such that $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (J) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

By Definition 4.22 we have (K) $C_1 \cong_{\psi_1} \hat{C}_1$, (L) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$ and $n \cong_{\psi_1} \hat{n}$. By Definition 4.19 we have $n \cong \hat{n}$. Given $(e) \nvDash \gamma$, we have $(n) \nvDash \gamma$ and therefore (M) $n = \hat{n}$.

Given (B) and (M), we have (N) $\hat{n}\ \neq\ 0$.

Given Axiom 4.15, we have $(l, \mu) \notin s$. Given (H), by Lemma 4.7 we have $s \cong_{\psi_1} \hat{s}$. Therefore, given (L), ψ_1, and (K), by Lemma 4.2 we have $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ s) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}) \parallel \hat{C}_1)$. Given (C), by the inductive hypothesis, we have (O) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and ψ_2 such that

$((\mathrm{p}, \gamma_1, \ \sigma_2, \ \Delta_2, \ \mathrm{acc}, \ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and (P) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22, we have $(\gamma_1, \sigma_2) \cong_{\psi_2} (\hat{\gamma}_1, \ \hat{\sigma}_2)$ and (Q) $C_2 \cong_{\psi_2} \hat{C}_2$. By Lemma 4.9, we have (R) $(\gamma, \ \sigma_2) \cong_{\psi_2} (\hat{\gamma}, \ \hat{\sigma}_2)$.

Given (D), (I), (N), and (O), we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{while}(\hat{e})\hat{s}) \parallel \hat{C})$ $\Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{wlc}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{while}(\hat{e})\hat{s}) \parallel \hat{C}_2)$ by Vanilla C rule While Continue.

Given Axiom 4.15, we have $(l, \mu) \notin$ while (e) s. Therefore, given (F), by Lemma 4.7 we have (S) while (e) $s \cong_{\psi_2}$ while $(\hat{e})$ $\hat{s}$.

Given (S), (R), and (Q), by Definition 4.22 we have $((\mathrm{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \mathrm{acc}, \ \mathrm{while}\ (e)\ s) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square,$ while $(\hat{e})$ $\hat{s}) \parallel \hat{C}_2)$.
By Definition 4.23 we have $wlc \cong \hat{wlc}$. Given (J) and (O), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wlc])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wlc}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wlc]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wlc}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ ty\ x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0)])}_{(\mathrm{p},[dp])} ((\mathrm{p}, \gamma_1, \ \sigma_1, \ \Delta, \ \mathrm{acc}, \ \mathrm{skip}) \parallel C)$

Given $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ ty\ x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0)])}_{(\mathrm{p},[dp])} ((\mathrm{p}, \gamma_1, \ \sigma_1, \ \Delta, \ \mathrm{acc}, \ \mathrm{skip}) \parallel C)$ by SMC^2 rule Public Pointer Declaration, we have (A) $(ty = \mathrm{public}\ bty*)$, acc $= 0$, (B) $l = \phi()$, (C) GetIndirection$(*) = i$, (D) $\omega = \mathrm{EncodePtr}(\mathrm{public}\ bty*, [1, [(l_{default}, 0)], [1], i])$, (E) $\gamma_1 = \gamma[x \ \rightarrow \ (l, \ \mathrm{public}\ bty*)]$, and (F) $\sigma_1 = \sigma[l \ \rightarrow$ $(\omega, \ \mathrm{public}\ bty*, \ 1, \ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public}\ bty*, \mathrm{public}, 1))]$.

Given (G) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ ty\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma, \ \sigma) \cong_{\psi} (\hat{\gamma}, \ \hat{\sigma})$, $C \cong_{\psi} \hat{C}$, and (I) $ty\ x \cong_{\psi} \hat{ty}\ \hat{x}$. By Definition 4.20 we have (J) $ty \cong_{\psi} \hat{ty}$ such that (K) $* = *$ and $x \cong_{\psi} \hat{x}$. Therefore, we have (L) $x = \hat{x}$.

Given (C) and (K), by Lemma 4.38 we have (M) GetIndirection$(*) = \hat{i}$ such that (N) $i = \hat{i}$.

Given (B), by Axiom 4.1 we have (O) $\hat{l} = \phi()$ and (P) $l = \hat{l}$.

Given (D), (J), (N), and $[1, [(l_{default}, 0)], [1], i] \cong_{\psi} [1, [(\hat{l}_{default}, 0)], [1], \hat{i}]$ by Definition 4.15, by Lemma 4.41 we have (Q) $\hat{\omega} = \mathrm{EncodePtr}(\hat{bty}*, [1, [(\hat{l}_{default}, 0)], [1], \hat{i}])$ such that (R) $\omega \cong_{\psi} \hat{\omega}$.

Given (E), (L), (P), (H), (A), and (I), by Lemma 4.12 we have (S) $\hat{\gamma}_1 = \hat{\gamma}[\hat{x} \rightarrow (\hat{l}, \ \hat{ty})]$ such that (T) $(\gamma_1, \sigma) \cong_{\psi}$ $(\hat{\gamma}_1, \ \hat{\sigma})$.

Given (F), (T), (P), (R), and (J), by Lemma 4.13 we have (U) $\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \rightarrow (\hat{\omega}, \ \hat{ty}, \ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable},$ $\hat{ty}, \mathrm{public}, 1))]$ such that (V) $(\gamma_1, \sigma_1) \cong_{\psi} (\hat{\gamma}_1, \ \hat{\sigma}_1)$.

Given (A) and (J), by Definition 4.8 we have (W) $(\hat{ty} = \hat{bty}*)$.

Given (G), (M), (O), (Q), (S), (U), and (W) we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\hat{dp}])} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C})$ by Vanilla C rule Pointer Declaration.

Given (U) and $C \cong_{\psi} \hat{C}$, by Definition 4.22 we have $((\mathrm{p}, \gamma_1, \ \sigma_1, \ \Delta, \ \mathrm{acc}, \ \mathrm{skip}) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C})$.

By Definition 4.23 we have $dp \cong \hat{dp}$ and by Definition 4.25 we have $(\mathrm{p}, [dp]) \cong (\mathrm{p}, [\hat{dp}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[dp1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[dp])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2)$

Given $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2)$ by SMC^2 rule Public Addition, we have (A) $(e_1, e_2) \nvdash \gamma$, (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$, (C) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_2) \parallel C_2)$, and (D) $n_1 + n_2 = n_3$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 + \hat{e}_2) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 + e_2) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 + \hat{e}_2) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$ and (F) $C \cong_{\psi} \hat{C}$. $e_1 + e_2 \cong_{\psi} \hat{e}_1 + \hat{e}_2$. By Definition 4.20 we have (G) $e_1 \cong_{\psi} \hat{e}_1$ and (H) $e_2 \cong_{\psi} \hat{e}_2$.

Given (E), ψ, (F), and (G), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C})$. Given (B), by the inductive hypothesis we have (I) $((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$ and ψ_1 such that $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1) \cong_{\psi_1} ((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$ and (J) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. By Definition 4.22 we have (K) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, $n_1 \cong_{\psi_1} \hat{n}_1$, and (L) $C_1 \cong_{\psi_1} \hat{C}_1$. Given (A), we have $(n_1) \nvdash \gamma$ and therefore by Definition 4.19 (M) $n_1 = \hat{n}_1$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (H), by Lemma 4.7 we have $e_2 \cong_{\psi_1} \hat{e}_2$. Therefore, given (K), ψ_1, and (L), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, e_2) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C})$. Given (C), by the inductive hypothesis we have (N) $((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2) \parallel \hat{C}_2)$ and ψ_2 such that $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_2) \parallel C_2) \cong_{\psi_2} ((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2) \parallel \hat{C}_2)$ and (O) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22 we have (P) $(\gamma,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)$, (Q) $C_2 \cong_{\psi_2} \hat{C}_2$ and $n_2 \cong_{\psi_2} \hat{n}_2$. Given (A), we have $(n_2) \nvdash \gamma$ and therefore by Definition 4.19 (R) $n_2 = \hat{n}_2$.

Given (D), (M), and (R), we have (S) $\hat{n}_1 + \hat{n}_2 = \hat{n}_3$ such that $n_3 = \hat{n}_3$ and therefore by Definition 4.19 (T) $n_3 \cong_{\psi_2} \hat{n}_3$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 + \hat{e}_2) \parallel \hat{C})$, (I), (N), and (S), by Vanilla C rule Addition we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square, \hat{e}_1 + \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{bp}])} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n}_3) \parallel \hat{C}_2)$.

Given (P), (Q), and (T), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2) \cong_{\psi_2} ((\mathrm{p},\ \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_3) \parallel \hat{C}_2)$. By Definition 4.23 we have $bp \cong \hat{bp}$. Given (J), (O), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [bp])$ and

$\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bp}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [bp]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{bp}])$. Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 - e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bs])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 + e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bp])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 \cdot e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bm])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 + e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bp])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 \div e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bd])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1 + e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[bp])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$

Given $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$ by SMC[2] rule Public Less Than True, we have (A) $(e_1, e_2) \nvdash \gamma$, (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$, (C) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_2) \parallel C_2)$, and (D) $(n_1 < n_2) = 1$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 < \hat{e}_2) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 < \hat{e}_2) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (F) $C \cong_\psi \hat{C}$ and $e_1 < e_2 \cong_\psi \hat{e}_1 < \hat{e}_2$. By Definition 4.20 we have (G) $e_1 \cong_\psi \hat{e}_1$ and (H) $e_2 \cong_\psi \hat{e}_2$.

Given (E), ψ, (F), and (G), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C})$. Given (B), by the inductive hypothesis we have (I) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$ and ψ_1 such that $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$ and (J) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. (K) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, (L) $C_1 \cong_{\psi_1} \hat{C}_1$, and $n_1 \cong_\psi \hat{n}_1$. Given (A), we have $(n_1) \nvdash \gamma$ and therefore by Definition 4.19 we have (M) $n_1 = \hat{n}_1$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (H), by Lemma 4.7 we have $e_2 \cong_{\psi_1} \hat{e}_2$. Therefore, given (K), ψ, and (L), by Lemma 4.2 we have $((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, e_2) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C})$. Given (C), by the inductive hypothesis we have (N) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2) \parallel \hat{C}_2)$ and ψ_2 such that $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n_2) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}_2) \parallel \hat{C}_2)$ and (O) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. By Definition 4.22 we have (P) $(\gamma,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)$, (Q) $C_2 \cong_{\psi_2} \hat{C}_2$, and $n_2 \cong_{\psi_2} \hat{n}_2$. Given (A), we have $(n_2) \nvdash \gamma$ and therefore by Definition 4.19 we have (R) $n_2 = \hat{n}_2$.

Given (D), (M), and (R), we have (S) $(\hat{n}_1 < \hat{n}_2) = 1$.

Given $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1 < \hat{e}_2) \parallel \hat{C})$, (I), (N), and (S), by Vanilla C rule Less Than True we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square, \hat{e}_1 < \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{p},[\hat{ltt}])} ((\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ 1) \parallel \hat{C}_2)$.

Given (P), (Q), and $1 = 1$, by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, 1) \parallel \hat{C}_2)$. By Definition 4.23 we have $ltt \cong \hat{ltt}$. Given (J), (O), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ltt])$ and

$\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{ltt}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ltt]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{ltt}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[eqt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[eqf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 != e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[net])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 != e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[nef])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\mathrm{p},[fd])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\mathrm{p},[fd])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$ by SMC^2 rule Function Definition, we have $\mathrm{acc} = 0$, (B) $x \notin \gamma$, (C) $l = \phi()$, (D) $\mathrm{GetFunTypeList}(P) = tyL$, (E) $\gamma_1 = \gamma[x \to (l,\ tyL \to ty)]$, (F) $\mathrm{CheckPublicEffects}(s,\ x,\ \gamma, \sigma) = n$, (G) $\mathrm{EncodeFun}(s,\ n,\ P) = \omega$, and (H) $\sigma_1 = \sigma[l \to (\omega,\ tyL \to ty,\ 1,\ \mathrm{PermL\_Fun}(\mathrm{public}))]$.

Given (I) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x(P)\{s\}) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

$\hat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}) \parallel \hat{C})$, by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (K) $C \cong_\psi \hat{C}$ and $ty\ x(P)\{s\} \cong_\psi \hat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}$. By Definition 4.20 we have (L) $ty \cong_\psi \hat{ty}$, $x \cong_\psi \hat{x}$ and therefore (M) $x = \hat{x}$, (N) $P \cong_\psi \hat{P}$, and (O) $s \cong_\psi \hat{s}$.

Given (B), (M), and (J), by Lemma 4.11 we have (P) $\hat{x} \notin \hat{\gamma}$.

Given (C) by Axiom 4.1 we have (Q) $\hat{l} = \phi()$ such that (R) $l = \hat{l}$.

Given (D) and (N), by Lemma 4.39 we have (S) GetFunTypeList$(\hat{P}) = \hat{tyL}$ such that (T) $tyL \cong_\psi \hat{tyL}$. Given (L) and (T), by Definition 4.7 we have (U) $tyL \rightarrow ty \cong_\psi \hat{tyL} \rightarrow \hat{ty}$.

Given (E), (J), (M), (R), and (U), by Lemma 4.12 we have (V) $\hat{\gamma}_1 = \hat{\gamma}[\hat{x} \rightarrow (\hat{l}, \hat{tyL} \rightarrow \hat{ty})]$ such that (W) $(\gamma_1,\ \sigma) \cong_\psi (\hat{\gamma}_1,\ \hat{\sigma})$.

Given (G), (N), and (O), by Lemma 4.44 we have (X) EncodeFun$(\hat{s}, \square, \hat{P}) = \hat{\omega}$ such that (Y) $\omega \cong_\psi \hat{\omega}$.

Given (H), (W), (R), (Y), and (U), by Lemma 4.13 we have (Z) $\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \rightarrow (\hat{\omega}, \hat{tyL} \rightarrow \hat{ty}, 1, \text{PermL\_Fun(public)})]$ such that (A1) $(\gamma_1,\ \sigma_1) \cong_\psi (\hat{\gamma}_1,\ \hat{\sigma}_1)$.

Given (I), (P), (Q), (S), (V), (X), and (Z), by Vanilla C rule Function Definition we have $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x}(\hat{P})\{\hat{s}\}) \parallel \hat{C}) \Downarrow'_{(\text{p},[\hat{fd}])} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C})$.

Given (A1) and (K), by Definition 4.22 we have $((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel C)$. By Definition 4.23 we have $fd \cong \hat{fd}$. Given $(\text{p}, [fd])$ and $(\text{p}, [\hat{fd}])$, by Definition 4.25 we have $(\text{p}, [fd]) \cong (\text{p}, [\hat{fd}])$. Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)) \parallel C) \Downarrow_{(\text{p},[df])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$. The main difference is that we are creating the function data as a NULL placeholder, to be defined later.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fpd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$. The main difference is that we taking out the NULL placeholder data and replacing it with the function data.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$ by SMC^2 rule Function Call With Public Side Effects, we have (B) $\gamma(x) = (l,\ tyL \rightarrow ty)$, (C) $\sigma(l) = (\omega,\ tyL \rightarrow ty,\ 1,\ \text{PermL\_Fun(public)})$, (D) DecodeFun$(\omega) = (s,\ n,\ P)$, (E) GetFunParamAssign$(P, E) = s_1$, (F) $\text{acc} = 0$, (G) $((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc}, s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)$, (H) $n = 1$, and (I) $((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$.

Given (J) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}(\hat{E})) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x(E)) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}(\hat{E})) \parallel \hat{C})$, by Definition 4.22 we have (K) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (L) $C \cong_\psi \hat{C}$, and (M) $x(E) \cong_\psi \hat{x}(\hat{E})$. By Definition 4.20 we have (N) $E \cong_\psi \hat{E}$ and $x \cong_\psi \hat{x}$. Therefore we have (O) $x = \hat{x}$.

Given (B), (K), and (O), by Lemma 4.14 we have (P) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{tyL} \to \hat{ty})$ such that (Q) $tyL \to ty \cong_\psi \hat{tyL} \to \hat{ty}$ and (R) $l = \hat{l}$.

Given (C), (K), and (R), by Lemma 4.15 we have (S) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{tyL} \to \hat{ty}, 1, \text{PermL\_Fun(public)})$ such that (T) $\omega \cong_\psi \hat{\omega}$.

Given (D) and (T), by Lemma 4.50 we have (U) $\text{DecodeFun}(\hat{\omega}) = (\hat{s}, \square, \hat{P})$ such that (V) $s \cong_\psi \hat{s}$ and (W) $P \cong_\psi \hat{P}$.

Given (E), (W), and (N), by Lemma 4.40 we have (X) $\text{GetFunParamAssign}(\hat{P}, \hat{E}) = \hat{s}_1$ such that (Y) $s_1 \cong_\psi \hat{s}_1$.

Given (G), (K), (L), and (Y), by Lemma 4.2 we have (Z) $((\text{p},\ \gamma,\ \sigma,\ \Delta,\ \text{acc}, s_1) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square, \hat{s}_1) \parallel \hat{C})$. Given (Z), by the inductive hypothesis we have (A1) $((\text{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square, \hat{s}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C}_1)$ and ψ_1 such that (B1) $((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C}_1)$ and (C1) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (B1), by Definition 4.22 we have (D1) $(\gamma_1, \sigma_1) \cong_{\psi_1} (\hat{\gamma}_1, \hat{\sigma}_1)$ and (E1) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given Axiom 4.15, we have $(l, \mu) \notin s$. Therefore, given (V), by Lemma 4.7 we have (F1) $s \cong_{\psi_2} \hat{s}$.

Given (I), (D1), (E1), and (F1), by Lemma 4.2 we have (G1) $((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, s) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{s}) \parallel \hat{C}_1)$. Given (G1), by the inductive hypothesis we have (H1) $((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{s}) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_2)$ and ψ_2 such that (I1) $((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2) \cong_{\psi_2} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_2)$ and (J1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. Given (I1), by Definition 4.22 we have (K1) $(\gamma_2, \sigma_2) \cong_{\psi_2} (\hat{\gamma}_2, \hat{\sigma}_2)$ and (L1) $C_2 \cong_{\psi_2} \hat{C}_2$.

Given (J1), by Lemma 4.9, we have (M1) $(\gamma,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)$.

Given (J), (P), (S), (U), (X), (A1), and (H1), by Vanilla C rule Function Call we have $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}(\hat{E})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{fc}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_2)$.

Given (M1) and (L1), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2) \cong_{\psi_2} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_2)$.
By Definition 4.23 we have $fc \cong \hat{fc}$. Given (E1) and (J1), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [fc])$ and
$\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{fc}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [fc]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{fc}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x(E)) \parallel C) \Downarrow^{(\text{p}, [(l,0)]) :: \mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [fc1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x(E)) \parallel C) \Downarrow^{(\text{p}, [(l,0)]) :: \mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [fc])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l,0)])}_{\mathcal{D}_1 :: (\text{p}, [w])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l,0)])}_{\mathcal{D}_1 :: (\text{p}, [w])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Write Public Variable, we have (B) $(e) \nvdash \gamma$, (C) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (D) $\gamma(x) = (l,\ \text{public } bty)$, and (E) $\text{UpdateVal}(\sigma_1,\ l,\ n,\ \text{public } bty) = \sigma_2$.

Given (F) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e})$

$\| \hat{C})$, by Definition 4.22 we have (G) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (H) $C \cong_\psi \hat{C}$, and (I) $x = e \cong_\psi \hat{x} = \hat{e}$. Given (I), by Definition 4.20 we have (J) $e \cong_\psi \hat{e}$ and $x \cong_\psi \hat{x}$. Therefore we have (K) $x = \hat{x}$.

Given (C), (G), (H), by Lemma 4.2 we have (L) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \| \hat{C})$ Given (L), by the inductive hypothesis we have (M) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \| \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \| \hat{C}_1)$ and ψ_1 such that (N) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \| C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \| \hat{C}_1)$ and (O) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (N), by Definition 4.22 we have (P) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (Q) $n \cong_{\psi_1} \hat{n}$ and (R) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (B), (C) and (Q), by Definition 4.19 we have (S) $n = \hat{n}$.

Given (D), (P), and (K), by Lemma 4.14 we have (T) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (U) public $bty \cong_{\psi_1} \hat{bty}$ and (V) $l = \hat{l}$.

Given (E), (P), (V), (Q), and (U), by Lemma 4.51 we have (W) $\text{UpdateVal}(\hat{\sigma}_1, \hat{l}, \hat{n}, \hat{bty}) = \hat{\sigma}_2$ such that (X) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (F), (M), (T), and (W), by Vanilla C rule Write we have $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \| \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{w}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \| \hat{C}_1)$.

Given (X) and (R), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \| C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \| \hat{C}_1)$.
By Definition 4.23 we have $w \cong \hat{w}$.
Given (O), $\mathcal{D}_1 :: (\text{p}, [w])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{w}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [w]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{w}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \| C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\text{p}, [w1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \| C_1)$

This case is similar to Case $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \| C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\text{p}, [w])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \| C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \| C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\text{p}, [w2])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \| C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \| C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\text{p}, [w2])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \| C_1)$ by SMC^2 rule Write Private Variable Public Value, we have (B) $(e) \nvdash \gamma$, (C) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \| C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \| C_1)$, (D) $\gamma(x) = (l,\ \text{private}\ bty)$, and (E) $\text{UpdateVal}(\sigma_1,\ l,\ \text{encrypt}(n),\ \text{private}\ bty) = \sigma_2$.

Given (F) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \| \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \| \hat{C})$, by Definition 4.22 we have (G) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (H) $C \cong_\psi \hat{C}$, and (I) $x = e \cong_\psi \hat{x} = \hat{e}$. Given (I), by Definition 4.20 we have (J) $e \cong_\psi \hat{e}$ and $x \cong_\psi \hat{x}$. Therefore we have (K) $x = \hat{x}$.

Given (C), (G), and (H), by Lemma 4.2 we have (L) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \| \hat{C})$ Given (C) and (L), by the inductive hypothesis we have (M) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \| \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \| \hat{C}_1)$ and ψ_1 such that (N) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (O) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (N), by Definition 4.22 we have (P) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (Q) $n \cong_{\psi_1} \hat{n}$ and (R) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (B), (C) and (Q), by Definition 4.19 we have $n = \hat{n}$ and therefore (S) $\mathrm{encrypt}(n) \cong_{\psi_1} \hat{n}$.

Given (D), (P), and (K), by Lemma 4.14 we have (T) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (U) private $bty \cong_{\psi_1} \hat{bty}$ and (V) $l = \hat{l}$.

Given (E), (P), (V), (S), and (U), by Lemma 4.51 we have (W) $\mathrm{UpdateVal}(\hat{\sigma}_1, \hat{l}, \hat{n}, \hat{bty}) = \hat{\sigma}_2$ such that (X) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (F), (M), (T), and (W), by Vanilla C rule Write we have $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{w}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$.

Given (X) and (R), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$.
By Definition 4.23 we have $w2 \cong \hat{w}$.
Given (O), $\mathcal{D}_1 :: (\mathrm{p}, [w2])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{w}])$ by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [w2]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{w}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow^{(\mathrm{p}, [(l,0)])}_{(\mathrm{p}, [r1])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow^{(\mathrm{p}, [(l,0)])}_{(\mathrm{p}, [r1])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$ by SMC^2 rule Read Private Variable, we have (B) $\gamma(x) = (l,\ \mathrm{private}\ bty)$, (C) $\sigma(l) = (\omega,\ \mathrm{private}\ bty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private}\ bty, \mathrm{private}, 1))$, and (D) $\mathrm{DecodeVal}(\mathrm{private}\ bty,\ \omega) = n$.

Given (E) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$ by Definition 4.22 we have (F) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (G) $C \cong_{\psi} \hat{C}$, and (H) $x \cong_{\psi} \hat{x}$. Given (H), by Definition 4.20 we have (I) $x = \hat{x}$.

Given (B), (F), and (I), by Lemma 4.14 we have (J) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (K) private $bty \cong_{\psi_1} \hat{bty}$ and (L) $l = \hat{l}$.

Given (C), (F), and (L), by Lemma 4.15 we have (M) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}, 1, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, 1))$ such that (N) $\omega \cong_{\psi} \hat{\omega}$.

Given (D), (K), and (N), by Lemma 4.45 we have (O) $\mathrm{DecodeVal}(\hat{bty}, \hat{\omega}) = \hat{n}$ such that (P) $n \cong_{\psi} \hat{n}$.

Given (E), (J), (M), and (O), by Vanilla C rule Read we have $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p}, [\hat{r}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

Given (F), (G), and (P), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

By Definition 4.23 we have $r1 \cong \hat{r}$, and by Definition 4.25 we have $(\mathrm{p}, [r1]) \cong (\mathrm{p}, [\hat{r}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[r])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[r1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[dv])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[dv])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$ by SMC^2 rule Public Declaration, we have (B) $(ty = \mathrm{public}\ bty)$, $\mathrm{acc} = 0$, (C) $l = \phi()$, (D) $\gamma_1 = \gamma[x \to (l,\ ty)]$, (E) $\omega = \mathrm{EncodeVal}(ty, \mathrm{NULL})$, and (F) $\sigma_1 = \sigma[l \to (\omega,\ ty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, ty, \mathrm{public}, 1))]$.

Given (G) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$ and (I) $ty\ x \cong_{\psi} \hat{bty}\ \hat{x}$. Given (B) and (I), by Definition 4.20 we have (J) public $bty \cong_{\psi} \hat{bty}$ such that (K) $bty = \hat{bty}$ and $x \cong_{\psi} \hat{x}$ such that (L) $x = \hat{x}$.

Given (C), by Axiom 4.1 we have (M) $\hat{l} = \phi()$ and (N) $l = \hat{l}$.

Given (D), (H), (L), (N), and (I), by Lemma 4.12 we have (O) $\hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l}, \hat{bty})]$ such that (P) $(\gamma_1, \sigma) \cong_{\psi} (\hat{\gamma}_1,\ \hat{\sigma})$.

Given (E) and (I), by Lemma 4.42 we have (Q) $\hat{\omega} = \mathrm{EncodeVal}(\hat{bty}, \mathrm{NULL})$ such that (R) $\omega \cong_{\psi} \hat{\omega}$.

Given (F), (N), (R), (I), and (P), by Lemma 4.13 we have (S) $\hat{\sigma}_1 = \hat{\sigma}[\hat{l} \to (\hat{\omega}, \hat{bty}, 1, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, 1))]$ such that (T) $(\gamma_1, \sigma_1) \cong_{\psi} (\hat{\gamma}_1,\ \hat{\sigma}_1)$.

Given (G), (M), (O), (Q), and (S), by Vanilla C rule Declaration we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\hat{dv}])} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C})$.

Given (T), by Definition 4.22 we have $((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C})$. By Definition 4.23 we have $dv \cong \hat{dv}$, and by Definition 4.25 we have $(\mathrm{p}, [dv]) \cong (\mathrm{p}, [\hat{dv}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[d1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{(\mathrm{p},[dv])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s_1;\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ss])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_2,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ v) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s_1;\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ss])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_2,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ v) \parallel C_2)$ by SMC^2 rule Statement Sequencing, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ v_1) \parallel C_1)$ and (C) $((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ s_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_2,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ v_2) \parallel C_2)$.

Given (D) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1; \hat{s}_2) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s_1;\ s_2) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1; \hat{s}_2) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (F) $C \cong_{\psi} \hat{C}$ and (G) $s_1;\ s_2 \cong_{\psi} \hat{s}_1; \hat{s}_2$. By Definition 4.12 we have (H) $s_1 \cong_{\psi} \hat{s}_1$ and (I) $s_2 \cong_{\psi} \hat{s}_2$.

Given ψ, (E), (F), and (H), by Lemma 4.2 we have (J) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ s_1) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1) \parallel \hat{C})$. Given (B) and (J), by the inductive hypothesis we have (K) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{v}_1) \parallel \hat{C}_1)$ and ψ_1 such that (L) $((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v_1) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{v}_1) \parallel \hat{C}_1)$ and (M) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (L), by Definition 4.22 we have (N) $(\gamma_1,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma}_1,\ \hat{\sigma}_1)$, (O) $v_1 \cong_{\psi_1} \hat{v}_1$, and (P) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given Axiom 4.15, we have $(l, \mu) \notin s_2$. Therefore, given (I), by Lemma 4.7 we have (Q) $s_2 \cong_{\psi_1} \hat{s}_2$.

Given ψ_1, (N), (P), and (Q), by Lemma 4.2 we have (R) $((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ s_2) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel \hat{C}_1)$. Given (C) and (R), by the inductive hypothesis we have (S) $((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \hat{v}_2) \parallel \hat{C}_2)$ and ψ_2 such that (T) $((\text{p}, \gamma_2,\ \sigma_2,\ \Delta_2,\ \text{acc},\ v_2) \parallel C_2) \cong_{\psi_2} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \hat{v}_2) \parallel \hat{C}_2)$ and (U) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (T), by Definition 4.22 we have (V) $(\gamma_2,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma}_2,\ \hat{\sigma}_2)$, (W) $v_2 \cong_{\psi_2} \hat{v}_2$, and (X) $C_2 \cong_{\psi_2} \hat{C}_2$.

Given (D), (K), and (S), by Vanilla C rule Statement Sequencing we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1; \hat{s}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{ss}])} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \hat{v}_2) \parallel \hat{C}_2)$.

Given (V), (W), and (X), by Definition 4.22 we have $((\text{p}, \gamma_2,\ \sigma_2,\ \Delta_2,\ \text{acc},\ v_2) \parallel C_2) \cong_{\psi_2} ((\text{p}, \hat{\gamma}_2, \hat{\sigma}_2, \square, \square, \hat{v}_2) \parallel \hat{C}_2)$. By Definition 4.23 we have $ss \cong \hat{ss}$. Given (M), (U), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{ss}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\text{p}, [\hat{ss}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \{s\}) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [sb])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \{s\}) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [sb])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Statement Block, we have (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ s) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1)$.

Given (C) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}_1; \hat{s}_2) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \{s\}) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \{\hat{s}\}) \parallel \hat{C})$, by Definition 4.22 we have (D) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (E) $C \cong_{\psi} \hat{C}$ and (F) $\{s\} \cong_{\psi} \{\hat{s}\}$. Given (F), by Definition 4.20 we have (G) $s \cong_{\psi} \hat{s}$.

Given ψ, (D), (E), and (G), by Lemma 4.2 we have (H) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ s) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C})$. Given (B) and (H), by the inductive hypothesis we have (I) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{s}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$ and ψ_1 such that (J) $((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$ and (K) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (J), by Definition 4.22 we have (L) $(\gamma_1,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma}_1,\ \hat{\sigma}_1)$, (M) $v \cong_{\psi_1} \hat{v}$, and (N) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (B), (I), and (J), by Lemma 4.9 we have (O) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$

Given (C) and (I), by Vanilla C rule Statement Block we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \{\hat{s}\}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{sb}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C}_1)$.

Given (O) and (N), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $sb \cong \hat{sb}$. Given (K), $\mathcal{D}_1 :: (\text{p}, [sb])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{sb}])$, by Lemma 4.10 we have

$\mathcal{D}_1 :: (\text{p}, [sb]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{sb}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (e)) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1::(\text{p},[ep])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (e)) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1::(\text{p},[ep])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1)$ by SMC^2 rule Parentheses, we have (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1)$.

Given (C) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{e})) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (e)) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{e})) \parallel \hat{C})$, by Definition 4.22 we have (D) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (E) $C \cong_{\psi} \hat{C}$ and (F) $(e) \cong_{\psi} (\hat{e})$. Given (F), by Definition 4.20 we have (G) $e \cong \hat{e}$.

Given ψ, (D), (E), and (G), by Lemma 4.2 we have (H) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (B) and (H), by the inductive hypothesis we have (I) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$ and ψ_1 such that (J) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$ and (K) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (J), by Definition 4.22 we have (L) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, (M) $v \cong_{\psi_1} \hat{v}$, and (N) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (C) and (I), by Vanilla C rule Parentheses we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\text{p},[\hat{ep}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$.

Given (L), (M), and (N), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ v) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{v}) \parallel \hat{C}_1)$. By Definition 4.23 we have $ep \cong \hat{ep}$. Given (L), $\mathcal{D}_1 :: (\text{p}, [ep])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{ep}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [ep]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{ep}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\, x = e) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ds])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\, x = e) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ds])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_2)$ by SMC^2 rule Declaration Assignment, we have (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\, x) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ and (C) $((\text{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \text{acc},\ x = e) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\text{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$.

Given (D) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\, \hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\, x = e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\, \hat{x} = \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (F) $C \cong_{\psi} \hat{C}$ and (G) $ty\, x = e \cong_{\psi} \hat{ty}\, \hat{x} = \hat{e}$. By Definition 4.20 we have (H) $ty \cong_{\psi} \hat{ty}$, (I) $x \cong_{\psi} \hat{x}$, such that (J) $x = \hat{x}$, and (K) $e \cong_{\psi} \hat{e}$.

Given (H) and (J), by Definition 4.20 we have (L) $ty\, x \cong_{\psi} \hat{ty}\, \hat{x}$.

Given ψ, (E), (F), and (L), by Lemma 4.2 we have (M) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\, x) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\, x) \parallel \hat{C})$.

Given (B) and (M), by the inductive hypothesis we have (N) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ x) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$ and ψ_1 such that (O) $((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$ and (P) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (O), by Definition 4.22 we have (Q) $(\gamma_1,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma}_1,\ \hat{\sigma}_1)$, and (R) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given Axiom 4.15, we have $(l, \mu) \notin e$. Therefore, given (K), by Lemma 4.7 we have (S) $e \cong_{\psi_1} \hat{e}$.

Given (J) and (S), by Definition 4.20 we have (T) $x = e \cong_{\psi_1} \hat{x} = \hat{e}$.

Given ψ_1, (Q), (T), and (R), by Lemma 4.2 we have (U) $((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ x = e) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C}_1)$. Given (C) and (U), by the inductive hypothesis we have (V) $((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_1, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and ψ_2 such that (W) $((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and (X) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (W), by Definition 4.22 we have (Y) $(\gamma_1,\ \sigma_2) \cong_{\psi_2} (\hat{\gamma}_1,\ \hat{\sigma}_2)$ and (Z) $C_2 \cong_{\psi_2} \hat{C}_2$.

Given (D), (N), and (V), by Vanilla C rule Declaration Assignment we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{ty}\ \hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{ds}])} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$.

Given (Y) and (Z), by Definition 4.22 we have $((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. By Definition 4.23 we have $ds \cong \hat{ds}$. Given (P), (X), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ds])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{ds}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ds]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{ds}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e_1] = e_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [das])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ds])} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0), (l_1, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [da1])} ((\mathrm{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0), (l_1, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [da1])} ((\mathrm{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC^2 rule Private Array Declaration, we have (B) $(e) \nvdash \gamma$, (C) $((ty = \mathrm{private}\ bty) \vee (ty = bty)) \wedge ((bty = \mathrm{int}) \vee (bty = \mathrm{float}))$, (D) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \alpha) \parallel C_1)$, (E) $\alpha > 0$, (F) $l = \phi()$, (G) $l_1 = \phi()$, (H) $\gamma_1 = \gamma[x \rightarrow (l,\ \mathrm{private\ const}\ bty*)]$, (I) $\omega = \mathrm{EncodePtr}(\mathrm{private\ const}\ bty*,\ [1,\ [(l_1, 0)],\ [1],\ 1])$, (J) $\omega_1 = \mathrm{EncodeArr}(\mathrm{private}\ bty, 0, \alpha, \mathrm{NULL})$, (K) $\sigma_2 = \sigma_1[l \rightarrow (\omega,\ \mathrm{private\ const}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private\ const}\ bty*, \mathrm{private}, 1))]$, and (L) $\sigma_3 = \sigma_2[l_1 \rightarrow (\omega_1,\ \mathrm{private}\ bty,\ \alpha, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private}\ bty, \mathrm{private}, \alpha))]$.

Given (M) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}[\hat{e}]) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e]) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}[\hat{e}]) \parallel \hat{C})$, by Definition 4.22 we have (N) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (O) $C \cong_{\psi} \hat{C}$ and (P) $ty\ x[e] \cong_{\psi} \hat{bty}\ \hat{x}[\hat{e}]$.

Given (P), by Definition 4.20 we have (Q) $ty \cong_{\psi} \hat{bty}$, (R) $x \cong_{\psi} \hat{x}$ such that (S) $x = \hat{x}$ and (T) $e \cong_{\psi} \hat{e}$. Given (C) and (Q), by Definition 4.8 we have (U) $bty = \hat{bty}$.

Given ψ, (N), (O), and (T), by Lemma 4.2 we have (V) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given

(D) and (V), by the inductive hypothesis we have (W) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{\alpha}) \parallel \hat{C}_1)$ and ψ_1 such that (X) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ \alpha) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{\alpha}) \parallel \hat{C}_1)$ and (Y) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (X), by Definition 4.22 we have (Z) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, (A1) $\alpha \cong_{\psi_1} \hat{\alpha}$, and (B1) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (F) and (G), by Axiom 4.1 we have (C1) $\hat{l} = \phi()$, (D1) $l = \hat{l}$, (E1) $\hat{l}_1 = \phi()$, and (F1) $l_1 = \hat{l}_1$.

Given (C), (Q), and (U), by Definition 4.8 we have (G1) private const $bty* \cong_\psi$ const $\hat{bty}*$ Given (H), (Z), (D1), and (G1), by Lemma 4.12 we have (H1) $\hat{\gamma}_1 = \hat{\gamma}[\hat{x} \to (\hat{l}, \mathrm{const}\ \hat{bty}*)]$ such that (I1) $(\gamma_1, \sigma_1) \cong_{\psi_1} (\hat{\gamma}_1, \hat{\sigma}_1)$.

Given (F1), by Definition 4.15 we have (J1) $[1,\ [(l_1, 0)],\ [1],\ 1] \cong_{\psi_1} [1, [(\hat{l}_1, 0)], [1], 1]$. Given (I), (G1), and (J1), by Lemma 4.41 we have (K1) $\hat{\omega} = \mathrm{EncodePtr}(\mathrm{const}\ \hat{bty}*, [1, [(\hat{l}_1, 0)], [1], 1])$ such that (L1) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (C), (Q), and (U), by Definition 4.8 we have (M1) private $bty \cong_\psi \hat{bty}$ Given (J), (M1), and (A1), by Lemma 4.43 we have (N1) $\hat{\omega}_1 = \mathrm{EncodeArr}(\hat{bty}, 0, \hat{\alpha}, \mathrm{NULL})$ such that (O1) $\omega_1 \cong_{\psi_1} \hat{\omega}_1$.

Given (A1) and (B), by Lemma 4.3 we have (P1) $\alpha = \hat{\alpha}$. Given (E) and (P1), we have (Q1) $\hat{\alpha} > 0$.

Given (K), (I1), (C1), (K1), and (G1), by Lemma 4.13 we have (R1) $\hat{\sigma}_2 = \hat{\sigma}_1[\hat{l} \to (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1,$ $\mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1))]$ such that (S1) $(\gamma_1, \sigma_2) \cong_{\psi_1} (\hat{\gamma}_1, \hat{\sigma}_2)$.

Given (L), (S1), (F1), (O1), (P1), and (M1), by Lemma 4.13 we have (T1) $\hat{\sigma}_3 = \hat{\sigma}_2[\hat{l}_1 \to (\hat{\omega}_1, \hat{bty}, \alpha, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \alpha))]$ such that (U1) $(\gamma_1, \sigma_3) \cong_{\psi_1} (\hat{\gamma}_1, \hat{\sigma}_3)$.

Given (M), (W), (C1), (E1), (H1), (K1), (N1), (Q1), (R1), and (T1), by Vanilla C rule Array Declaration we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{bty}\ \hat{x}[\hat{e}]) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{da}])} ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$.

Given (U1) and (B1), by Definition 4.22 we have $((\mathrm{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1) \cong_\psi ((\mathrm{p}, \hat{\gamma}_1, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $da1 \cong \hat{da}$. Given (Y), $\mathcal{D}_1 :: (\mathrm{p}, [da1])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{da}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [da1]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{da}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0),(l_1,0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [da])} ((\mathrm{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0),(l_1,0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [da1])} ((\mathrm{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1 :: (\mathrm{p}, [ra])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1 :: (\mathrm{p}, [ra])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$ by SMC[2] rule Public Array Read Public Index, we have (B) $(e) \nvdash \gamma$, (C) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1)$, (D) $\gamma(x) = (l,\ \mathrm{public\ const}\ bty*)$, (E) $\sigma_1(l) = (\omega,\ \mathrm{public\ const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public\ const}\ bty*, \mathrm{public}, 1))$, (F) $\mathrm{DecodePtr}(\mathrm{public\ const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (G) $\sigma_1(l_1) = (\omega_1, \mathrm{public}\ bty, \alpha, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{public}\ bty, \mathrm{public}, \alpha))$, (H) $0 \le i \le \alpha - 1$, and (I) $\mathrm{DecodeArr}(\mathrm{public}\ bty,\ i,\ \omega_1) = n_i$.

Given (J) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C})$, by

Definition 4.22 we have (K) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, (L) $x[e] \cong_{\psi} \hat{x}[\hat{e}]$, and (M) $C \cong_{\psi} \hat{C}$. Given (L), by Definition 4.20 we have (N) $e \cong_{\psi} \hat{e}$ and $x \cong_{\psi} \hat{x}$ such that (O) $x = \hat{x}$.

Given ψ, (K), (N), and (M), by Lemma 4.2 we have (P) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (C) and (P), by the inductive hypothesis we have (Q) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$ and ψ_1 such that $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$. By Definition 4.22 we have (R) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$ (S) $i \cong_{\psi_1} \hat{i}$, (T) $C_1 \cong_{\psi_1} \hat{C}_1$, and (U) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (D), (R), and (O), by Lemma 4.14 we have (V) $\hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*)$ such that (W) $\mathrm{public\ const}\ bty* \cong_{\psi_1} \mathrm{const}\ \hat{bty}*$ and (X) $l = \hat{l}$.

Given (E), (R), and (X), by Lemma 4.15 we have (Y) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1))$ such that (Z) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (F), (W), and (Z), by Lemma 4.49 we have (A1) $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (B1) $l_1 = \hat{l}_1$.

Given (G), (R), and (B1), by Lemma 4.15 we have (C1) $\hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha}))$ such that (D1) $\omega_1 \cong_{\psi_1} \hat{\omega}_1$, (E1) $\alpha = \hat{\alpha}$, and (F1) $\mathrm{public}\ bty \cong_{\psi_1} \hat{bty}$.

Given (S) and (B), by Lemma 4.3 we have (G1) $i = \hat{i}$. Given (H), (G1), and (E1), we have (H1) $0 \leq \hat{i} \leq \hat{\alpha} - 1$.

Given (I), (F1), (G1), and (D1), by Lemma 4.46 we have (I1) $\mathrm{DecodeArr}(\hat{bty},\ \hat{i},\ \hat{\omega}_1) = \hat{n}_{\hat{i}}$ such that (J1) $n_i \cong_{\psi_1} \hat{n}_{\hat{i}}$.

Given (J), (Q), (V), (Y), (A1), (C1), (H1), and (I1), by Vanilla C rule Array Read we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p}, [\hat{ra}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}) \parallel \hat{C}_1)$.

Given (R), (J1), and (T), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}) \parallel \hat{C}_1)$. By Definition 4.23 we have $ra \cong \hat{ra}$. Given (U), $\mathcal{D}_1 :: (\mathrm{p}, [ra])$, and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{ra}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [ra]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{ra}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1::(\mathrm{p}, [ra1])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1::(\mathrm{p}, [ra])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [wa2])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_1,i)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [wa2])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$ by SMC[2] rule Private Array Write Private Value Public Index, we have (B) $(e_1) \nvdash \gamma$, (C) $(e_2) \vdash \gamma$, (D) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e_1) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, i) \parallel C_1)$, (E) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n) \parallel C_2)$, (F) $\gamma(x) = (l,\ \mathrm{private\ const}\ bty*)$, (G) $\sigma_2(l) = (\omega, \mathrm{private\ const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private\ const}\ bty*, \mathrm{private}, 1))$, (H) $\mathrm{DecodePtr}(\mathrm{private\ const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (I) $\sigma_2(l_1) = (\omega_1, \mathrm{private}\ bty, \alpha, \mathrm{PermL}(\mathrm{Freeable},\ \mathrm{private}\ bty, \mathrm{private}, \alpha))$, (J) $0\ \leq\ i\ \leq\ \alpha - 1$, (K) $\mathrm{DynamicUpdate}(\Delta_2, \sigma_2, [(l_1, i)],\ \mathrm{acc}, \mathrm{private}\ bty) = \Delta_3$, and (L) $\mathrm{UpdateArr}(\sigma_2,\ (l_1, i),\ n,\ \mathrm{private}\ bty) = \sigma_3$.

Given (M) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

$\hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C})$, by Definition 4.22 we have (N) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (O) $x[e_1] = e_2 \cong_\psi \hat{x}[\hat{e}_1] = \hat{e}_2$, and (P) $C \cong_\psi \hat{C}$. By Definition 4.20 we have (Q) $e_1 \cong_\psi \hat{e}_1$, (R) $e_2 \cong_\psi \hat{e}_2$, and $x \cong_\psi \hat{x}$ such that (S) $x = \hat{x}$.

Given ψ, (N), (Q), and (P), by Lemma 4.2 we have (T) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e_1) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, \hat{e}_1) \parallel \hat{C})$ Given (D) and (T), by the inductive hypothesis we have (U) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$ and ψ_1 such that (V) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$. Given (V), by Definition 4.22 we have (W) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (X) $i \cong_{\psi_1} \hat{i}$, (Y) $C_1 \cong_{\psi_1} \hat{C}_1$, and (Z) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (B) and (X), by Lemma 4.3 we have (A1) $i = \hat{i}$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (R), by Lemma 4.7 we have (B1) $e_2 \cong_{\psi_1} \hat{e}_2$.

Given ψ_1, (W), (B1), and (Y), by Lemma 4.2 we have (C1) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1)$. Given (E) and (C1), by the inductive hypothesis we have (D1) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \hat{C}_2)$ and ψ_2 such that (E1) $((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \hat{C}_2)$. Given (E1), by Definition 4.22 we have (F1) $(\gamma, \sigma_2) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)$, (G1) $n \cong_{\psi_2} \hat{n}$, and (H1) $C_2 \cong_{\psi_2} \hat{C}_2$, and (I1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (F), (F1), and (S), by Lemma 4.14 we have (J1) $\hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*)$ such that (K1) $l = \hat{l}$ and (L1) private const $bty* \cong_{\psi_2}$ const $\hat{bty}*$. Given (L1), by Definition 4.8 we have (M1) private $bty \cong_{\psi_2} \hat{bty}$.

Given (G), (F1), and (K1), by Lemma 4.15 we have (N1) $\hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1))$ such that (O1) $\omega \cong_{\psi_2} \hat{\omega}$.

Given (H), (L1), and (O1), by Lemma 4.49 we have (P1) $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (Q1) $l_1 = \hat{l}_1$.

Given (I), (Q1), and (F1), by Lemma 4.15 we have (R1) $\hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha}))$ such that (S1) $\omega_1 \cong_{\psi_2} \hat{\omega}_1$, (T1) $\alpha = \hat{\alpha}$.

Given (J), (A1), and (T1), we have (U1) $0 \leq \hat{i} \leq \hat{\alpha} - 1$.

Given (L), (E1), (O1), (Z), (R1), and (K1), by Lemma 4.52 we have (V1) $\mathrm{UpdateArr}(\hat{\sigma}_2,\ (\hat{l}_1, \hat{i}),\ \hat{n},\ \hat{bty}) = \hat{\sigma}_3$ such that (W1) $(\gamma, \sigma_3) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_3)$.

Given (M), (U), (D1), (J1), (N1), (P1), (R1), (U1), and (V1), by Vanilla C rule Array Write we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wa}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$.

Given (W1) and (H1), by Definition 4.22 we have $((\mathrm{p}, \gamma, \sigma_3, \Delta_3, \mathrm{acc}, \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. By Definition 4.23 we have $wa2 \cong \hat{wa}$. Given (Z), (I1), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa2])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wa}])$, by

Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa2]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wa}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa1])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l,0), (l_1, i)])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa2])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l,0), (l_1, i)])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$. Given $n = \hat{n}$, we use Definition 4.19 to prove that $\mathrm{encrypt}(n) \cong \hat{n}$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l,0), (l_1, i)])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wa2])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l,0), (l_1, i)])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p}, [real])}^{(\mathrm{p}, [(l,0), (l_1, 0), \ldots, (l_1, \alpha-1)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n_0, ..., n_{\alpha-1}]) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p}, [real])}^{(\mathrm{p}, [(l,0), (l_1, 0), \ldots, (l_1, \alpha-1)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n_0, ..., n_{\alpha-1}]) \parallel C)$ by SMC^2 rule Read Entire Array, we have (B) $\gamma(x) = (l, a\ \mathrm{const}\ bty*)$, (C) $\sigma(l) = (\omega, a\ \mathrm{const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a\ \mathrm{const}\ bty*, a, 1))$, (D) $\mathrm{DecodePtr}(a\ \mathrm{const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (E) $\sigma(l_1) = (\omega_1, a\ bty, \alpha, \mathrm{PermL}(\mathrm{Freeable}, a\ bty, a, \alpha))$, and (F) $\forall i \in \{0...\alpha-1\}\ \mathrm{DecodeArr}(a\ bty, i, \omega_1) = n_i$.

Given (G) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, (I) $x \cong_\psi \hat{x}$, and (J) $C \cong_\psi \hat{C}$. Given (I), by Definition 4.20 we have (K) $x = \hat{x}$.

Given (B), (H), and (K), by Lemma 4.14 we have (L) $\hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*)$ such that (M) $l = \hat{l}$ and (N) $a\ \mathrm{const}\ bty* \cong_\psi \mathrm{const}\ \hat{bty}*$.

Given (C), (H), and (M), by Lemma 4.15 we have (O) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1))$ such that (P) $\omega \cong_\psi \hat{\omega}$.

Given (D), (N), and (P), by Lemma 4.49 we have (Q) $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (R) $l_1 = \hat{l}_1$.

Given (E), (H), (R), by Lemma 4.15 we have (T) $\hat{\sigma}(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, bty, \mathrm{public}, \hat{\alpha}))$ such that (U) $\omega_1 \cong_\psi \hat{\omega}_1$, (V) $bty \cong_\psi \hat{bty}$, and (W) $\alpha = \hat{\alpha}$.

Given (F) and (W), we have (X) $i = \hat{i}$. Given (F), (X), (W), (V), and (U), by Lemma 4.47 we have (Y) $\forall \hat{i} \in \{0...\hat{\alpha}-1\}$ $\mathrm{DecodeArr}(\hat{bty}, \hat{i}, \hat{\omega}_1) = \hat{n}_{\hat{i}}$ such that (Z) $\forall i \in \{0...\alpha-1\}\ n_i \cong_\psi \hat{n}_i$.

Given (G), (L), (O), (Q), (T), and (Y), by Vanilla C rule Read Entire Array we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p}, [\hat{real}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, [\hat{n}_0, ..., \hat{n}_{\hat{\alpha}-1}]) \parallel \hat{C})$.

Given (H), (J), (W), and (Z), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n_0, ..., n_{\alpha-1}]) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, [\hat{n}_0, ..., \hat{n}_{\hat{\alpha}-1}]) \parallel \hat{C})$.

By Definition 4.23 we have $rea \cong \hat{rea}$, and by Definition 4.25 we have $(\mathrm{p}, [rea]) \cong (\mathrm{p}, [\hat{rea}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea1])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea1])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC[2] rule Write Entire Private Array, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ [n_0, \ldots, n_{\alpha_e-1}]) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{private const } bty*)$, (D) $(e) \vdash \gamma$, (E) $\sigma_1(l) = (\omega, \text{private const } bty*, 1,$ $\mathrm{PermL\_Ptr}(\mathrm{Freeable}, \text{private const } bty*, \text{private}, 1))$, (F) $\mathrm{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (G) $\sigma_1(l_1) = (\omega_1, \text{private } bty, \alpha, \mathrm{PermL}(\mathrm{Freeable}, \text{private } bty, \text{private}, \alpha))$, (H) $\alpha_e = \alpha$, and (I) $\forall i \in \{0 \ldots \alpha-1\}$ $\mathrm{UpdateArr}(\sigma_{1+i},\ (l_1, i),\ n_i,\ \text{private } bty) = \sigma_{2+i}$.

Given (J) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (K) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (L) $C \cong_{\psi} \hat{C}$, and (M) $x = e \cong_{\psi} \hat{x} = \hat{e}$. Given (M), by Definition 4.20 we have (N) $e \cong_{\psi} \hat{e}$ and $x \cong_{\psi} \hat{x}$ such that (O) $x = \hat{x}$.

Given ψ, (K), (L), and (N), by Lemma 4.2 we have (P) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$ Given (B) and (P), by the inductive hypothesis we have (Q) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, [\hat{n}_0, \ldots, \hat{n}_{\hat{\alpha}_e-1}]) \parallel \hat{C}_1)$ and ψ_1 such that (R) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, [n_0, \ldots, n_{\alpha_e-1}]) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, [\hat{n}_0, \ldots, \hat{n}_{\hat{\alpha}_e-1}]) \parallel \hat{C}_1)$ and (S) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (R), by Definition 4.22 we have (T) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (U) $[n_0, \ldots, n_{\alpha_e-1}] \cong_{\psi_1} [\hat{n}_0, \ldots, \hat{n}_{\hat{\alpha}_e-1}]$, and (V) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (C), (T), and (O), by Lemma 4.14 we have (W) $\hat{\gamma}(\hat{x}) = (\hat{l}, \text{const } \hat{bty}*)$ such that (X) $l = \hat{l}$ and (Y) private const $bty* \cong_{\psi_1}$ const $\hat{bty}*$. By Definition 4.8 we have (Z) $bty = \hat{bty}$.

Given (E), (T), and (X), by Lemma 4.15 we have (A1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \text{const } \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \text{const } \hat{bty}*,$ $\text{public}, 1))$ such that (B1) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (F), (Y), and (B1), by Lemma 4.49 we have (C1) $\mathrm{DecodePtr}(\text{const } \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (D1) $l_1 = \hat{l}_1$.

Given (G), (T), and (D1), by Lemma 4.15 we have (E1) $\hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, bty, \text{public}, \hat{\alpha}))$ such that (F1) $\omega_1 \cong_{\psi_1} \hat{\omega}_1$ and (G1) $\alpha = \hat{\alpha}$.

Given (U), by Definition 4.20 we have (H1) $\alpha_e = \hat{\alpha}_e$. Given (H), (H1), and (G1), we have (I1) $\hat{\alpha}_e = \hat{\alpha}$.

Given (I) and (G1), we have (J1) $i = \hat{i} \in \{0 \ldots \alpha-1\}$. Given (I), (T), (D1), (U), (Z), (I1), (G1), and (J1), by Lemma 4.53 we have (K1) $\forall \hat{i} \in \{0 \ldots \hat{\alpha} - 1\}$ $\mathrm{UpdateArr}(\hat{\sigma}_{1+\hat{i}},\ (\hat{l}_1, \hat{i}),\ \hat{n}_{\hat{i}},\ \hat{bty}) = \sigma_{2+\hat{i}}$ such that (L1) $(\gamma, \sigma_{2+i}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_{2+\hat{i}})$.

Given (J), (Q), (W), (A1), (C1), (E1), (H1), and (K1), by Vanilla C rule Write Entire Array we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$ $\hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{p},[\hat{wea}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_{2+\hat{\alpha}-1}, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$.

Given (L1) and (V), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc}, \mathrm{skip})\ \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_{2+\hat{\alpha}-1}, \square, \square,$ $\mathrm{skip})\ \parallel \hat{C}_1)$.
By Definition 4.23 we have $wea1 \cong \hat{wea}$. Given (S), $\mathcal{D}_1 :: (\mathrm{p}, [wea1])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{wea}])$, by Lemma 4.10 we

have $\mathcal{D}_1 :: (\text{p}, [\mathit{wea1}]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{\mathit{wea}}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{wea2}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_1, 0), \ldots, (l_1, \alpha - 1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{wea1}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_1, 0), \ldots, (l_1, \alpha - 1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,$ acc, skip$) \parallel C_1)$. Given $n = \hat{n}$, we use Definition 4.18 to prove that $\text{encrypt}(n) \cong \hat{n}$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{wea}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_1, 0), \ldots, (l_1, \alpha - 1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{wea1}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_1, 0), \ldots, (l_1, \alpha - 1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,$ acc, skip$) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{rao}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_2, \mu)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [\mathit{rao}])}^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_2, \mu)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$ by SMC^2 rule Public Array Read Out of Bounds Public Index, we have (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ i) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{public const } \mathit{bty}*)$, (D) $(e) \nvdash \gamma$, (E) $\sigma_1(l) = (\omega,\ \text{public const } \mathit{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const } \mathit{bty}*, \text{public}, 1))$, (F) $\text{DecodePtr}(\text{public const } \mathit{bty}*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (G) $\sigma_1(l_1) = (\omega_1, \text{public } \mathit{bty}, \alpha, \text{PermL}(\text{Freeable}, \text{public } \mathit{bty}, \text{public}, \alpha))$, (H) $(i < 0) \lor (i \geq \alpha)$, and (I) $\text{ReadOOB}(i, \alpha, l_1, \text{public } \mathit{bty}, \sigma_1) = (n, 1, (l_2, \mu))$.

Given (J) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C})$ and ψ such that $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C})$, by Definition 4.22 we have (K) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (L) $C \cong_{\psi} \hat{C}$, and (M) $x[e] \cong_{\psi} \hat{x}[\hat{e}]$. Given (M), by Definition 4.20 we have (N) $e \cong_{\psi} \hat{e}$ and $x \cong_{\psi} \hat{x}$ such that (O) $x = \hat{x}$.

Given ψ, (K), (N), and (L), by Lemma 4.2 we have (P) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given

(B) and (P), by the inductive hypothesis we have (Q) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$ and ψ_1 such that (R) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ i) \parallel C_1) \cong_{\psi_1} (\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$ and (S) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (R), by Definition 4.22 we have (T) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (U) $i \cong_{\psi_1} \hat{i}$, and (V) $C_1 \cong_{\psi_1} \hat{C}_1$. Given (D) and (U) by Lemmas 4.4 and 4.3, we have (W) $i = \hat{i}$.

Given (C), (T), and (O), by Lemma 4.14 we have (X) $\hat{\gamma}(\hat{x}) = (\hat{l}, \text{const}\ \hat{bty}*)$ such that (Y) $l = \hat{l}$ and (Z) public const $bty* \cong_{\psi_1}$ const $\hat{bty}*$. Given (Z), by Definition 4.8 we have (A1) public $bty \cong_{\psi_1} \hat{bty}$.

Given (E), (T), and (Y), by Lemma 4.15 we have (B1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \text{const}\ \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{const}\ \hat{bty}*, \text{public}, 1))$ such that (C1) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (F), (Z), and (C1), by Lemma 4.49 we have (D1) $\text{DecodePtr}(\text{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (E1) $l_1 = \hat{l}_1$.

Given (G), (T), and (E1), by Lemma 4.15 we have (F1) $\hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \text{PermL}(\text{Freeable}, \hat{bty}, \text{public}, \hat{\alpha}))$ such that (G1) $\omega_1 \cong_{\psi_1} \hat{bty}_1$, and (H1) $\alpha = \hat{\alpha}$.

Given (H), (W), and (H1), we have (I1) $(\hat{i} < 0) \vee (\hat{i} \geq \hat{\alpha})$.

Given (I), (W), (H1), (E1), (A1), and (T), by Lemma 4.55 we have (J1) $\text{ReadOOB}(\hat{i}, \hat{\alpha}, \hat{l}_1, \hat{bty}, \hat{\sigma}_1) = (\hat{n}, 1)$ such that (K1) $n \cong_{\psi_1} \hat{n}$.

Given (J), (Q), (X), (B1), (D1), (F1), (I1), and (J1), by Vanilla C rule Array Read Out of Bounds we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{rao}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$.

Given (T), (K1), and (V), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$. By Definition 4.23 we have $rao \cong \hat{rao}$. Given (S), $\mathcal{D}_1 :: (\text{p}, [rao])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{rao}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [rao]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{rao}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_2, \mu)])}_{\mathcal{D}_1 :: (\text{p}, [rao1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_2, \mu)])}_{\mathcal{D}_1 :: (\text{p}, [rao])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_2, \mu)])}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [wao2])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \text{acc},\ \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_2, \mu)])}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [wao2])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \text{acc},\ \text{skip}) \parallel C_2)$ by SMC^2 rule Private Array Write Out of Bounds Public Index Private Value, we have (B) $(e_1) \nvdash \gamma$, (C) $(e_2) \vdash \gamma$, (D) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc}, e_1)\ \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, i) \parallel C_1)$, (E) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, e_2) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, n) \parallel C_2)$, (F) $\gamma(x) = (l, \text{private const}\ bty*)$, (G) $\sigma_2(l) = (\omega, \text{private const}\ bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const}\ bty*, \text{private}, 1))$, (H) $\text{DecodePtr}(\text{private const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (I) $\sigma_2(l_1) = (\omega_1,\ \text{private}\ bty,\ \alpha,\ \text{PermL}(\text{Freeable}, \text{private}\ bty, \text{private}, \alpha))$, (J) $(i\ <\ 0) \vee (i\ \geq\ \alpha)$, and (K) $\text{WriteOOB}(n, i, \alpha, l_1, \text{private}\ bty,\ \sigma_2,\ \Delta_2, \text{acc}) = (\sigma_3,\ \Delta_3, 1, (l_2, \mu))$.

Given (L) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C})$ and ψ such that (M) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e_1]\ = e_2) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

$\hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C})$, by Definition 4.22 we have (N) $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, (O) $x[e_1] = e_2 \cong_{\psi} \hat{x}[\hat{e}_1] = \hat{e}_2$, and (P) $C \cong_{\psi} \hat{C}$. By Definition 4.20 we have (Q) $e_1 \cong_{\psi} \hat{e}_1$, (R) $e_2 \cong_{\psi} \hat{e}_2$, and $x \cong_{\psi} \hat{x}$ such that (S) $x = \hat{x}$.

Given ψ, (N), (Q), and (P), by Lemma 4.2 we have (T) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e_1) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C})$ Given (D) and (T), by the inductive hypothesis we have (U) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$ and ψ_1 such that (V) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, i) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel \hat{C}_1)$. Given (V), by Definition 4.22 we have (W) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (X) $i \cong_{\psi_1} \hat{i}$, (Y) $C_1 \cong_{\psi_1} \hat{C}_1$, and (Z) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (B) and (X), by Lemma 4.3 we have (A1) $i = \hat{i}$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (R), by Lemma 4.7 we have (B1) $e_2 \cong_{\psi_1} \hat{e}_2$.

Given ψ_1, (W), (B1), and (Y), by Lemma 4.2 we have (C1) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, e_2) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1)$. Given (E) and (C1), by the inductive hypothesis we have (D1) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \hat{C}_2)$ and ψ_2 such that (E1) $((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, n) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel \hat{C}_2)$. Given (E1), by Definition 4.22 we have (F1) $(\gamma, \sigma_2) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)$, (G1) $n \cong_{\psi_2} \hat{n}$, and (H1) $C_2 \cong_{\psi_2} \hat{C}_2$, and (I1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (F), (F1), and (S), by Lemma 4.14 we have (J1) $\hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*)$ (K1) $l = \hat{l}$ and (L1) private const $bty* \cong_{\psi_2}$ const $\hat{bty}*$. Given (L1), by Definition 4.8 we have (M1) private $bty \cong_{\psi_2} \hat{bty}$.

Given (G), (F1), and (K1), by Lemma 4.15 we have (N1) $\hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*, \mathrm{public}, 1))$ such that (O1) $\omega \cong_{\psi_2} \hat{\omega}$.

Given (H), (L1), and (O1), by Lemma 4.49 we have (P1) $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ (Q1) $l_1 = \hat{l}_1$.

Given (I), (Q1), and (F1), by Lemma 4.15 we have (R1) $\hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha}))$ such that (S1) $\omega_1 \cong_{\psi_2} \hat{\omega}_1$, (T1) $\alpha = \hat{\alpha}$.

Given (J), (A1), and (T1), we have (U1) $(\hat{i} < 0) \lor (\hat{i} \geq \hat{\alpha})$.

Given (K), (G1), (A1), (T1), (Q1), (M1), and (F1), by Lemma 4.56 we have (V1) $\mathrm{WriteOOB}(\hat{n}, \hat{i}, \hat{\alpha}, \hat{l}_1, \hat{bty}, \hat{\sigma}_2) = (\hat{\sigma}_3, 1)$ such that (W1) $(\gamma, \sigma_3) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_3)$.

Given (L), (U), (D1), (J1), (N1), (P1), (R1), (U1), and (V1), by Vanilla C rule Array Write Out of Bounds we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wao}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$.

Given (W1) and (H1), by Definition 4.22 we have $((\mathrm{p}, \gamma, \sigma_3, \Delta_3, \mathrm{acc}, \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. By Definition 4.23 we have $wao2 \cong \hat{wao}$. Given (Z), (I1), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wao2])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{wao}])$, by

Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [\mathit{wao2}]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{\mathit{wao}}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [\mathit{wao}])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [\mathit{wao2}])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [\mathit{wao1}])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p}, [(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p}, [\mathit{wao2}])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p}, [\mathit{iep}])}$ $((1, \gamma^1, \sigma_6^1, \Delta_3^1, \mathrm{acc}, \mathrm{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2))$ $\Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p}, [\mathit{iep}])} ((1, \gamma^1, \sigma_6^1, \Delta_3^1, \mathrm{acc}, \mathrm{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Private If Else (Variable Tracking), we have (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1)$ $\parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$, (C) $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{\mathrm{Extract}(s_1,\ s_2, \gamma^{\mathrm{p}}) = (x_{list}, 0)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\{\mathrm{InitializeVariables}(x_{list}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, n^{\mathrm{p}}, \mathrm{acc}{+}1) = (\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, L_2^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $((1, \gamma_1^1, \sigma_2^1, \Delta_1^1, \mathrm{acc}{+}1, s_1)\ \parallel ... \parallel\ (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}{+}1, s_1)) \Downarrow^{\mathcal{L}_3}_{\mathcal{D}_2} ((1, \gamma_2^1, \sigma_3^1, \Delta_2^1, \mathrm{acc}{+}1, \mathrm{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma_2^{\mathrm{q}}, \sigma_3^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}{+}1, \mathrm{skip}))$, (G) $\{\mathrm{RestoreVariables}(x_{list}, \gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}, \mathrm{acc} + 1) = (\sigma_4^{\mathrm{p}}, L_4^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $((1, \gamma_1^1, \sigma_4^1, \Delta_2^1, \mathrm{acc} + 1, s_2)\ \parallel ... \parallel\ (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_4^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc} + 1, s_2)) \Downarrow^{\mathcal{L}_5}_{\mathcal{D}_3} ((1, \gamma_3^1, \sigma_5^1, \Delta_3^1, \mathrm{acc} + 1, \mathrm{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma_3^{\mathrm{q}}, \sigma_5^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc} + 1, \mathrm{skip}))$ (I) $\{\mathrm{ResolveVariables\_Retrieve}(x_{list},\ \mathrm{acc} + 1, \gamma_1^{\mathrm{p}}, \sigma_5^{\mathrm{p}}) = ([(v_{t1}^{\mathrm{p}}, v_{e1}^{\mathrm{p}}), ..., (v_{tm}^{\mathrm{p}}, v_{em}^{\mathrm{p}})], n^{\mathrm{p}}, L_6^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\mathrm{MPC}_{resolve}([n^1, ..., n^{\mathrm{q}}], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^{\mathrm{q}}, v_{e1}^{\mathrm{q}}), ..., (v_{tm}^{\mathrm{q}}, v_{em}^{\mathrm{q}})]]) = [[v_1^1, ..., v_m^1], ..., [v_1^{\mathrm{q}}, ..., v_m^{\mathrm{q}}]]$, (K) $\{\mathrm{ResolveVariables\_Store}(x_{list},\ \mathrm{acc} + 1, \gamma_1^{\mathrm{p}}, \sigma_5^{\mathrm{p}}, [v_1^{\mathrm{p}}, ..., v_m^{\mathrm{p}}]) = (\sigma_6^{\mathrm{p}}, L_7^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\mathcal{L}_2 = (1, L_2^1) \parallel ... \parallel (\mathrm{q}, L_2^{\mathrm{q}})$, $\mathcal{L}_4 = (1, L_4^1) \parallel ... \parallel (\mathrm{q}, L_4^{\mathrm{q}})$, $\mathcal{L}_6 = (1, L_6^1) \parallel ... \parallel (\mathrm{q}, L_6^{\mathrm{q}})$, and $\mathcal{L}_7 = (1, L_7^1) \parallel ... \parallel (\mathrm{q}, L_7^{\mathrm{q}})$.

Given Axiom 4.15, by Theorem 4.1 we have (L) $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (L), (M) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$ and ψ such that (N) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc},\ \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \cong_{\psi} ((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2)\ \parallel ...\ \parallel\ (\mathrm{q}, \hat{\gamma}^{\mathrm{q}}, \hat{\sigma}^{\mathrm{q}}, \square, \square,\ \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$, by Lemma 4.86, we have (O) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (P) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$.

Given (O), by Definition 4.22 we have (Q) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (R) $\mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2 \cong_{\psi} \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2$. Given (R), by Definition 4.20 we have (S) $e \cong_{\psi} \hat{e}$ such that (T) $s_1 \cong_{\psi} \hat{s}_1$ and (U) $s_2 \cong_{\psi} \hat{s}_2$.

Given ψ, (Q), and (S), by Lemma 4.2 we have (V) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e)\ \ \parallel ... \parallel\ \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \cong_{\psi} ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})\ \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}))$. Given (B) and (V), by the inductive hypothesis we have (W) $((1, \hat{\gamma}, \hat{\sigma}, \square,$

$\square, \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}))$ and ψ_1 such that (X) $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}})) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}))$ and (Y) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (X), by Definition 4.22 we have (Z) $\{(\gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (A1) $\{n^{\mathrm{p}} \cong_{\psi_1} \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given Axiom 4.15, we have $(l, \mu) \notin s_1$. Given (T), by Lemma 4.7 we have (B1) $s_1 \cong_{\psi_1} \hat{s}_1$.

Given (D), by Lemma 4.69 we have (C1) that all updates to memory in either branch will be caught by variables $x \in x_{list}$.

Given (E) and (C1), by Lemma 4.70 we have (D1) $\forall x_i \in x_{list}, \mathrm{p} \in \{1...\mathrm{q}\}, (\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \models (x_i\_else\_\mathrm{acc} \equiv v\_orig_i^{\mathrm{p}})$.

Given (E) and (Z), by Lemma 4.65 we have (E1) $\{(\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ such that (F1) $\{\sigma_2^{\mathrm{p}} = \sigma_1^{\mathrm{p}} :: \sigma_{temp1}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (E1) and (B1), by Lemma 4.2 we have (G1) $((1, \gamma_1^1, \sigma_2^1, \Delta_1^1, \mathrm{acc}+1, s_1) \parallel ... \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}+1, s_1)) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1))$. Given (F) and (G1), by the inductive hypothesis we have (H1) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$ and ψ_2 such that (I1) $((1, \gamma_2^1, \sigma_3^1, \Delta_2^1, \mathrm{acc}+1, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma_2^{\mathrm{q}}, \sigma_3^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, \mathrm{skip})) \cong_{\psi_2} ((1, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$ and (J1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (I1), by Definition 4.22 we have (K1) $\{(\gamma_2^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma}_1, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K1), (E1), (F), and (H1), by Lemma 4.9 we have (L1) $\{(\gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F) and (F1), by Lemma 4.71 we have (M1) $\{\sigma_3^{\mathrm{p}} = \sigma_3'^{\mathrm{p}} :: \sigma_{temp1}'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$ such that (N1) $\{\sigma_{temp1}'^{\mathrm{p}} = \sigma_{temp1}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (F1), (M1), (N1), and (D1), we have (O1) $\forall x_i \in x_{list}, \mathrm{p} \in \{1...\mathrm{q}\}, (\gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \models (x_i\_else\_\mathrm{acc} \equiv v\_orig_i^{\mathrm{p}})$.

Given (G), (C1), (O1), (E1), (L1), and (F1), by Lemma 4.72 we have (P1) $\{\forall x_i \in x_{list}, (\gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \models (x_i \equiv v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Q1) $\{\forall x_i \in x_{list}\ (\gamma_1^{\mathrm{p}}, \sigma_4^{\mathrm{p}}) \models (x_i\_then\_\mathrm{acc} \equiv v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (R1) $\{\sigma_4^{\mathrm{p}} = \sigma_1^{\mathrm{p}} :: \sigma_{temp2}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (S1) $\{(\gamma_1^{\mathrm{p}}, \sigma_4^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given Axiom 4.15, we have $(l, \mu) \notin s_2$. Given (U), by Lemma 4.7 we have (T1) $s_2 \cong_{\psi_2} \hat{s}_2$.

Given (S1) and (T1), by Lemma 4.2 we have (U1) $((1, \gamma_1^1, \sigma_4^1, \Delta_2^1, \mathrm{acc}+1, s_2) \parallel ... \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_4^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, s_2)) \cong_{\psi_2} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2))$. Given (H) and (U1), by the inductive hypothesis we have (V1) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_3} ((1, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$ and ψ_3 such that (W1) $((1, \gamma_3^1, \sigma_5^1, \Delta_3^1, \text{acc} + 1, \text{skip}) \parallel \ldots \parallel (\text{q}, \gamma_3^{\text{q}}, \sigma_5^{\text{q}}, \Delta_3^{\text{q}}, \text{acc} + 1, \text{skip})) \cong_{\psi_3} ((1, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \text{skip}) \parallel \ldots \parallel (\text{q}, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \text{skip}))$ and (X1) $\mathcal{D}_3 \cong \hat{\mathcal{D}}_3$.

Given (W1), by Definition 4.22 we have (Y1) $\{(\gamma_3^{\text{p}}, \sigma_5^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}_2, \hat{\sigma}_3)\}_{\text{p}=1}^{\text{q}}$.

Given (Y1), (S1), (H), and (V1), by Lemma 4.9 we have (Z1) $\{(\gamma_1^{\text{p}}, \sigma_5^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\text{p}=1}^{\text{q}}$.

Given (A1), (B), and (I), by Definition 4.19 and Lemma 4.73 we have (A2) $\{n^{\text{p}} \cong \hat{n}\}_{\text{p}=1}^{\text{q}}$.

Given (H) and (R1), by Lemma 4.71 we have (B2) $\{\sigma_5^{\text{p}} = \sigma_5'^{\text{p}} :: \sigma_{temp2}'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ such that (C2) $\{\sigma_{temp2}'^{\text{p}} = \sigma_{temp2}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.
Given (R1), (B2), (C2), and (Q1), we have (D2) $\forall x_i \in x_{list}, \text{p} \in \{1 \ldots \text{q}\}, (\gamma_1^{\text{p}}, \sigma_5^{\text{p}}) \models (x_i\_then\_\text{acc} \equiv v_{ti}^{\text{p}})$.

Given (I), (H), (A2), (Z1), (C1), and (D2), by Lemma 4.74 (E2) $\{\forall x_i \in x_{list}, (\gamma_1^{\text{p}}, \sigma_5^{\text{p}}) \models (x_i \equiv v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, and (F2) $\{\forall x_i \in x_{list}, (\gamma_1^{\text{p}}, \sigma_5^{\text{p}}) \models (x_i\_then\_\text{acc} \equiv v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$.

Subcase (G2) $\hat{n} = 0$

Given (J), (A2), (E2), (F2), and (G2), by Axiom 4.5 we have (H2) $\{\forall i \in \{1 \ldots m\}, v_i^{\text{p}} = v_{ei}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (K), (H2), (C1), (E2), and (Z1), by Lemma 4.84 we have (I2) $\{\forall x \in x_{list}, (\gamma^{\text{p}}, \sigma_f^{\text{p}}) \models (x \equiv v_{ei}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ and (J2) $\{(\gamma_1^{\text{p}}, \sigma_6^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\text{p}=1}^{\text{q}}$.

Given (J2) and (Q), by Lemma 4.9 we have (K2) $\{(\gamma^{\text{p}}, \sigma_6^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\text{p}=1}^{\text{q}}$.

Given (P), (W), (H1), (V1), and (G2), by Vanilla C rule Multiparty If Else False we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{if}(\hat{e})\ \hat{s}_1\ \text{else}\ \hat{s}_2) \parallel \ldots \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{if}(\hat{e})\ \hat{s}_1\ \text{else}\ \hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\text{p}, [\hat{mpief}])} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \text{skip}) \parallel \ldots \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \text{skip}))$.

Given (K2), by Definition 4.22 we have $((1, \gamma^1, \sigma_6^1, \Delta_3^1, \text{acc}, \text{skip}) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_6^{\text{q}}, \Delta_3^{\text{q}}, \text{acc}, \text{skip})) \cong_{\psi_3} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \text{skip}) \parallel \ldots \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \text{skip}))$.
By Definition 4.23 we have $iep \cong \hat{mpief}$.
Given (Y), (J1), (X1), $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [iep])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\text{p}, [\hat{mpief}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [iep]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\text{p}, [\hat{mpief}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_3} \Sigma$.

Subcase (G3) $\hat{n} \neq 0$

Given (J), (A2), (E2), (F2), and (G3), by Axiom 4.6 we have (H3) $\{\forall i \in \{1 \ldots m\}, v_i^{\text{p}} = v_{ti}^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (K), (C1), (H3), (L1), and (P1), by Lemma 4.85 we have (I3) $\{\forall x \in x_{list}, (\gamma^{\text{p}}, \sigma_f^{\text{p}}) \models (x \equiv v_{ti}^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ and (J3) $\{(\gamma_1^{\text{p}}, \sigma_6^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$.

Given (J3) and (Q), by Lemma 4.9 we have (K3) $\{(\gamma^{\text{p}}, \sigma_6^{\text{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_2)\}_{\text{p}=1}^{\text{q}}$.

Given (P), (W), (H1), (V1), and (G3), by Vanilla C rule Multiparty If Else True we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{if}(\hat{e})\ \hat{s}_1\ \text{else}\ \hat{s}_2) \parallel \ldots \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{if}(\hat{e})\ \hat{s}_1\ \text{else}\ \hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\text{p}, [\hat{mpiet}])} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \ldots \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}))$.

Given (K3), by Definition 4.22 we have $((1, \gamma^1, \sigma_6^1, \Delta_3^1, \text{acc}, \text{skip}) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_6^{\text{q}}, \Delta_3^{\text{q}}, \text{acc}, \text{skip})) \cong_{\psi_3} ((1, \hat{\gamma}, \hat{\sigma}_2, \square,$

$\square$, skip) $\|$ ... $\|$ (q, $\hat{\gamma}$, $\hat{\sigma}_2$, $\square$, $\square$, skip)).
By Definition 4.23 we have $iep \cong \hat{mpiet}$.
Given (Y), (J1), (X1), $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iep])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpiet}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iep]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpiet}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_3} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p}, [iepd])}$ $((1, \gamma^1, \sigma^1_6, \Delta^1_6, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}_6, \Delta^\mathrm{q}_6, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p}, [iepd])}$ $((1, \gamma^1, \sigma^1_6, \Delta^1_6, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}_6, \Delta^\mathrm{q}_6, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Private If Else (Location Tracking), we have (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}, e)) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}_1, \Delta^\mathrm{q}_1, \mathrm{acc}, n^\mathrm{q}))$, (C) $\{(e) \vdash \gamma^\mathrm{p}\}^\mathrm{q}_{\mathrm{p}=1}$, (D) $\{\mathrm{Extract}(s_1, s_2, \gamma^\mathrm{p}) = (x_{list}, 1)\}^\mathrm{q}_{\mathrm{p}=1}$, (E) $\{\mathrm{Initialize}(\Delta^\mathrm{p}_1, x_{list}, \gamma^\mathrm{p}, \sigma^\mathrm{p}_1, n^\mathrm{p}, \mathrm{acc} + 1) = (\gamma^\mathrm{p}_1, \sigma^\mathrm{p}_2, \Delta^\mathrm{p}_2, L^\mathrm{p}_2)\}^\mathrm{q}_{\mathrm{p}=1}$, (F) $((1, \gamma^1_1, \sigma^1_2, \Delta^1_2, \mathrm{acc} + 1, s_1) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}_1, \sigma^\mathrm{q}_2, \Delta^\mathrm{q}_2, \mathrm{acc} + 1, s_1)) \Downarrow^{\mathcal{L}_3}_{\mathcal{D}_2} ((1, \gamma^1_2, \sigma^1_3, \Delta^1_3, \mathrm{acc} + 1, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}_2, \sigma^\mathrm{q}_3, \Delta^\mathrm{q}_3, \mathrm{acc} + 1, \mathrm{skip}))$, (G) $\{\mathrm{Restore}(\sigma^\mathrm{p}_3, \Delta^\mathrm{p}_3, \mathrm{acc} + 1) = (\sigma^\mathrm{p}_4, \Delta^\mathrm{p}_4, L^\mathrm{p}_4)\}^\mathrm{q}_{\mathrm{p}=1}$, (H) $((1, \gamma^1_1, \sigma^1_4, \Delta^1_4, \mathrm{acc}+1, s_2) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}_1, \sigma^\mathrm{q}_4, \Delta^\mathrm{q}_4, \mathrm{acc}+1, s_2)) \Downarrow^{\mathcal{L}_5}_{\mathcal{D}_3} ((1, \gamma^1_3, \sigma^1_5, \Delta^1_5, \mathrm{acc}+1, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}_3, \sigma^\mathrm{q}_5, \Delta^\mathrm{q}_5, \mathrm{acc} + 1, \mathrm{skip}))$, (I) $\{\mathrm{Resolve\_Retrieve}(\gamma^\mathrm{p}_1, \sigma^\mathrm{p}_5, \Delta^\mathrm{p}_5, \mathrm{acc} + 1) = ([(v^\mathrm{p}_{t1}, v^\mathrm{p}_{e1}), ..., (v^\mathrm{p}_{tm}, v^\mathrm{p}_{em})], n^\mathrm{p}, L^\mathrm{p}_6)\}^\mathrm{q}_{\mathrm{p}=1}$, (J) $\mathrm{MPC}_{resolve}([n^1, ..., n^\mathrm{q}], [[(v^1_{t1}, v^1_{e1}), ..., (v^1_{tm}, v^1_{em})], ..., [(v^\mathrm{q}_{t1}, v^\mathrm{q}_{e1}), ..., (v^\mathrm{q}_{tm}, v^\mathrm{q}_{em})]]) = [[v^1_1, ..., v^1_m], ...[v^\mathrm{q}_1, ..., v^\mathrm{q}_m]]$, (K) $\{\mathrm{Resolve\_Store}(\Delta^\mathrm{p}_5, \sigma^\mathrm{p}_5, \mathrm{acc} + 1, [v^\mathrm{p}_1, ..., v^\mathrm{p}_m]) = (\sigma^\mathrm{p}_6, \Delta^\mathrm{p}_6, L^\mathrm{p}_7)\}^\mathrm{q}_{\mathrm{p}=1}$, $\mathcal{L}_2 = (1, L^1_2) \parallel ... \parallel (\mathrm{q}, L^\mathrm{q}_2)$, $\mathcal{L}_4 = (1, L^1_4) \parallel ... \parallel (\mathrm{q}, L^\mathrm{q}_4)$, $\mathcal{L}_6 = (1, L^1_6) \parallel ... \parallel (\mathrm{q}, L^\mathrm{q}_6)$, and $\mathcal{L}_7 = (1, L^1_7) \parallel ... \parallel (\mathrm{q}, L^\mathrm{q}_7)$.

Given Axiom 4.15, by Theorem 4.1 we have (L) $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \sim (\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)\}^\mathrm{q}_{\mathrm{p}=1}$.

Given (L), (M) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$ and ψ such that (N) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \cong_\psi ((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}^\mathrm{q}, \hat{\sigma}^\mathrm{q}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$, by Lemma 4.86, we have (O) $\{(\mathrm{p}, \gamma^\mathrm{p}, \sigma^\mathrm{p}, \Delta^\mathrm{p}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \cong_\psi (\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2)\}^\mathrm{q}_{\mathrm{p}=1}$. and therefore (P) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2))$.

Given (O), by Definition 4.22 we have (Q) $\{(\gamma^\mathrm{p}, \sigma^\mathrm{p}) \cong_\psi (\hat{\gamma}, \hat{\sigma})\}^\mathrm{q}_{\mathrm{p}=1}$, and (R) $\mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2 \cong_\psi \mathrm{if}(\hat{e})\ \hat{s}_1\ \mathrm{else}\ \hat{s}_2$.
Given (R), by Definition 4.20 we have (S) $e \cong_\psi \hat{e}$ such that (T) $s_1 \cong_\psi \hat{s}_1$ and (U) $s_2 \cong_\psi \hat{s}_2$.

Given ψ, (Q), and (S), by Lemma 4.2 we have (V) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}, \Delta^\mathrm{q}, \mathrm{acc}, e)) \cong_\psi ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}))$. Given (B) and (V), by the inductive hypothesis we have (W) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}))$ and ψ_1 such that (X) $((1, \gamma^1, \sigma^1_1, \Delta^1_1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^\mathrm{q}, \sigma^\mathrm{q}_1, \Delta^\mathrm{q}_1, \mathrm{acc}, n^\mathrm{q})) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}))$ and (Y) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$.

Given (X), by Definition 4.22 we have (Z) $\{(\gamma^\mathrm{p}, \sigma^\mathrm{p}_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}^\mathrm{q}_{\mathrm{p}=1}$ and (A1) $\{n^\mathrm{p} \cong_{\psi_1} \hat{n}\}^\mathrm{q}_{\mathrm{p}=1}$.

Given Axiom 4.15, we have $(l, \mu) \notin s_1$. Given (T), by Lemma 4.7 we have (B1) $s_1 \cong_{\psi_1} \hat{s}_1$.

Given (E) and (Z), by Lemma 4.78 we have (C1) $\{(\gamma^\mathrm{p}_1, \sigma^\mathrm{p}_2) \cong_\psi (\hat{\gamma}, \hat{\sigma}_1)\}^\mathrm{q}_{\mathrm{p}=1}$.

Given (D), (E), by Lemma 4.77 we have (D1) all updates to a *constant location* dictated by variable x will have their original value stored within $\{\Delta_2^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (E1) $\{(\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \vDash (res\_\mathrm{acc} \equiv n^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (F1) $\{\sigma_2^{\mathrm{p}} = \sigma_1^{\mathrm{p}} :: \sigma_{temp1}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C1) and (B1), by Lemma 4.2 we have (G1) $((1, \gamma_1^1, \sigma_2^1, \Delta_1^1, \mathrm{acc}+1, s_1) \parallel \ldots \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}+1, s_1)) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1))$. Given (F) and (G1), by the inductive hypothesis we have (H1) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_1)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$ and ψ_2 such that (I1) $((1, \gamma_2^1, \sigma_3^1, \Delta_2^1, \mathrm{acc}+1, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma_2^{\mathrm{q}}, \sigma_3^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, \mathrm{skip})) \cong_{\psi_2} ((1, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}_1, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$ and (J1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (I1), by Definition 4.22 we have (K1) $\{(\gamma_2^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma}_1, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K1), (C1), (F), and (H1), by Lemma 4.9 we have (L1) $\{(\gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F), by Lemma 4.79 we have (M1) $\{\Delta_3^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *complete*.

Given (G), (M1), (L1), and (C1), by Lemma 4.80 we have (N1) $\{\Delta_4^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *then-complete*, and (O1) $\{(\gamma_1^{\mathrm{p}}, \sigma_4^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given Axiom 4.15, we have (P1) $(l, \mu) \notin s_2$. Given (U) and (P1), by Lemma 4.7 we have (Q1) $s_2 \cong_{\psi_2} \hat{s}_2$.

Given (O1) and (Q1), by Lemma 4.2 we have (R1) $((1, \gamma_1^1, \sigma_4^1, \Delta_2^1, \mathrm{acc}+1, s_2) \parallel \ldots \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_4^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, s_2)) \cong_{\psi_2} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2))$. Given (H) and (R1), by the inductive hypothesis we have (S1) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_3} ((1, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$ and ψ_3 such that (T1) $((1, \gamma_3^1, \sigma_5^1, \Delta_3^1, \mathrm{acc}+1, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma_3^{\mathrm{q}}, \sigma_5^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}+1, \mathrm{skip})) \cong_{\psi_3} ((1, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}_2, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$ and (U1) $\mathcal{D}_3 \cong \hat{\mathcal{D}}_3$.

Given (T1), by Definition 4.22 we have (V1) $\{(\gamma_3^{\mathrm{p}}, \sigma_5^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}_2, \hat{\sigma}_3)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (V1), (O1), (H), and (S1), by Lemma 4.9 we have (W1) $\{(\gamma_1^{\mathrm{p}}, \sigma_5^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (A1), (B), (D), (E), (F), (G), (H), and (I), by Definition 4.19 and Lemma 4.81 we have (X1) $\{n^{\mathrm{p}} \cong \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (H), by Lemma 4.79 we have (Y1) $\{\Delta_5^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *complete*. Given (N1), (H), and (Y1), by Lemma 4.82 we have that (Z1) $\{\Delta_5^{\mathrm{p}}[\mathrm{acc}+1]\}_{\mathrm{p}=1}^{\mathrm{q}}$ is *else-complete*.

Given (Z1), (F), (H), and (I), by Lemma 4.83 we have (A2) $\{\forall (l_i, \mu_i) = (v_{oi}^{\mathrm{p}}, v_{ti}^{\mathrm{p}}, 1, ty_i) \in \Delta_5^{\mathrm{p}}[\mathrm{acc}], (\sigma_3^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty} v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (B2) $\{\forall (l_i, \mu_i) = (v_{ti}^{\mathrm{p}}, \mathrm{NULL}, 0, ty_i) \in \Delta_5^{\mathrm{p}}[\mathrm{acc}], (\sigma_3^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (C2) $\{\forall (l_i, \mu_i) = (v_{oi}^{\mathrm{p}}, v_{ti}^{\mathrm{p}}, j, ty_i) \in \Delta_5^{\mathrm{p}}[\mathrm{acc}], (\sigma_5^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Subcase (D2) $\hat{n} = 0$

Given (J), (X1), (A2), (B2), (D2), and (C2), by Axiom 4.5 we have (E2) $\{\forall i \in \{1...m\}, v_i^{\mathrm{p}} = v_{ei}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K), (W1), (Z1), (C2), and (E2), by Lemma 4.84 we have (F2) $\{(\gamma_1^{\mathrm{p}}, \sigma_6^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\mathrm{p}=1}^{\mathrm{q}}$ (G2) $\{\forall (l_i, \mu_i) = (v_{oi}^{\mathrm{p}}, v_{ti}^{\mathrm{p}}, j, ty_i) \in \Delta_1^{\mathrm{p}}[\mathrm{acc}], (\sigma_f^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ei}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F2) and (Q), by Lemma 4.9 we have (H2) $\{(\gamma^{\mathrm{p}}, \sigma_6^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_3)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (P), (W), (H1), (S1), and (D2), by Vanilla C rule Multiparty If Else False we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1$

else $\hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1$ else $\hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::\hat{\mathcal{D}}_3::(\mathrm{p},[\hat{mpief}])} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square,$ skip)).

Given (H2), by Definition 4.22 we have $((1, \gamma^1, \sigma_6^1, \Delta_6^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_6^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip})) \cong_{\psi_3} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$.
By Definition 4.23 we have $iepd \cong \hat{mpief}$.
Given (Y), (J1), (U1), $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iepd])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpief}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iepd]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpief}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_3} \Sigma$.

Subcase (D3) $\hat{n} \neq 0$

Given (J), (X1), (A2), (B2), (D3), and (C2), by Axiom 4.6 we have (E3) $\{\forall i \in \{1...m\}, v_i^{\mathrm{p}} = v_{ti}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K), (L1), (Z1), (A2), (B2), and (E3), by Lemma 4.85 we have (F3) $\{(\gamma_1^{\mathrm{p}}, \sigma_6^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$ (G3) $\{\forall (l_i, \mu_i) = (v_{oi}^{\mathrm{p}}, v_{ti}^{\mathrm{p}}, 1, ty_i) \in \Delta_5^{\mathrm{p}}[\mathrm{acc}], (\sigma_6^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (H3) $\{\forall (l_i, \mu_i) = (v_{ti}^{\mathrm{p}}, \mathrm{NULL}, 0, ty_i) \in \Delta_5^{\mathrm{p}}[\mathrm{acc}], (\sigma_6^{\mathrm{p}}) \vDash_l ((l_i, \mu_i) \equiv_{ty_i} v_{ti}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F3) and (Q), by Lemma 4.9 we have (I3) $\{(\gamma^{\mathrm{p}}, \sigma_6^{\mathrm{p}}) \cong_{\psi_3} (\hat{\gamma}, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (P), (W), (H1), (S1), and (D3), by Vanilla C rule Multiparty If Else True we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1$ else $\hat{s}_2) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{if}(\hat{e})\ \hat{s}_1$ else $\hat{s}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::\hat{\mathcal{D}}_3::(\mathrm{p},[\hat{mpiet}])} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square,$ skip)).

Given (I3), by Definition 4.22 we have $((1, \gamma^1, \sigma_6^1, \Delta_6^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_6^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip})) \cong_{\psi_3} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$.
By Definition 4.23 we have $iepd \cong \hat{mpiet}$.
Given (Y), (J1), (U1), $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iepd])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpiet}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iepd]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: \hat{\mathcal{D}}_3 :: (\mathrm{p}, [\hat{mpiet}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_3} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, ++\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin3])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, n_2) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, ++\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin3])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, n_2) \parallel C)$ by SMC^2 rule Pre-Increment Private Int Variable, we have (B) $\gamma(x) = (l, \mathrm{private\ int})$, (C) $\sigma(l) = (\omega, \mathrm{private\ int}, 1, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private\ int}, \mathrm{private}, 1))$, (D) $\mathrm{DecodeVal}(\mathrm{private\ int}, \omega) = n_1$, (E) $n_2 = n_1 + \mathrm{encrypt}(1)$, and (F) $\mathrm{UpdateVal}(\sigma, l, n_2, \mathrm{private\ int}) = \sigma_1$.

Given (G) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x}) \parallel \hat{C})$ and ψ such that (H) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, ++\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x})$

$\| \hat{C})$ by Definition 4.22 we have (I) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (J) $C \cong_\psi \hat{C}$, and (K) $++\ x \cong_\psi ++\ \hat{x}$. Given (K), by Definition 4.20 we have (L) $x = \hat{x}$.

Given (B), (I), and (L), by Lemma 4.14 we have (M) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (N) $l = \hat{l}$ and (O) private int $\cong_\psi \hat{bty}$.

Given (C), (I), and (N), by Lemma 4.15 we have (P) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}, 1, \text{PermL}(\text{Freeable}, \hat{bty}, \text{public}, 1))$ such that (Q) $\omega \cong_\psi \hat{\omega}$.

Given (D), (O), and (Q), by Lemma 4.45 we have (R) $\text{DecodeVal}(\hat{bty}, \hat{\omega}) = \hat{n}_1$ such that (S) $n_1 \cong_\psi \hat{n}_1$.

Given (E), by Definition 4.19 we have (T) $\text{encrypt}(1) \cong_\psi 1$. Given (E) and (T), we have (U) $\hat{n}_2 = \hat{n}_1 + 1$ such that (V) $n_2 \cong_\psi \hat{n}_2$.

Given (F), (I), (N), (V), and (O), by Lemma 4.51 we have (W) $\text{UpdateVal}(\hat{\sigma}, \hat{l}, \hat{n}_2, \hat{bty}) = \hat{\sigma}_1$ such that (X) $(\gamma,\ \sigma_1) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)$.

Given (G), (M), (P), (R), (U), and (W), by Vanilla C rule Pre-Increment Variable we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x}) \| \hat{C}) \Downarrow'_{(\text{p}, [\hat{pin}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2) \| \hat{C})$.

Given (X), (V), and (J), by Definition 4.22 we have $((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n_2) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2) \| \hat{C})$. By Definition 4.23 we have $pin3 \cong \hat{pin}$, and by Definition 4.25 we have $(\text{p}, [pin3]) \cong (\text{p}, [\hat{pin}])$. Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ n_1) \| C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ++\ x) \| C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin3])} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n_2) \| C)$. The main difference is the value of x is equal instead of congruent, and we add 1 without encryption.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l,0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$ by SMC^2 rule Pre-Increment Public Pointer Single Location, we have (B) $\gamma(x) = (l,\ \text{public}\ bty*)$, (C) $\sigma(l) = (\omega,\ \text{public}\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{public}\ bty*, \text{public}, 1))$, (D) $\text{DecodePtr}(\text{public}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1]$, (E) $((l_2, \mu_2), 1) = \text{GetLocation}((l_1, \mu_1), \tau(\text{public}\ bty), \sigma)$, and (F) $\text{UpdatePtr}(\sigma,\ (l, 0),\ [1,\ [(l_2, \mu_2)],\ [1],\ 1], \text{public}\ bty*) = (\sigma_1,\ 1)$.

Given (G) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x}) \| \hat{C})$ and ψ such that (H) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x})$

$\| \hat{C})$, by Definition 4.22 we have (I) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (J) $C \cong_\psi \hat{C}$, and (K) $++\ x \cong_\psi ++\ \hat{x}$. Given (K), by Definition 4.20 we have (L) $x = \hat{x}$.

Given (B), (I), and (L), by Lemma 4.14 we have (M) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty*})$ such that (N) $l = \hat{l}$ and (O) public $bty* \cong_\psi \hat{bty*}$.

Given (C), (I), and (N), by Lemma 4.15 we have (P) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty*}, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty*}, \text{public}, 1))$ such that (Q) $\omega \cong_\psi \hat{\omega}$.

Given (D), (O), and (Q) by Lemma 4.48 we have (R) $\text{DecodePtr}(\hat{bty*}, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$ (S) $[1, [(l_1, \mu_1)], [1],\ 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$. Given (S), by Definition 4.15 we have (T) $(l_1, \mu_1) \cong_\psi (\hat{l}_1, \hat{\mu}_1)$.

Given (O), by Definition 4.8 we have (U) public $bty \cong_\psi \hat{bty}$.

Given (E), (T), (U), and (I), by Lemma 4.57 we have (V) $((\hat{l}_2, \hat{\mu}_2), 1) = \text{GetLocation}((\hat{l}_1, \hat{\mu}_1), \tau(\hat{bty}), \hat{\sigma})$ such that (W) $(l_2, \mu_2) \cong_\psi (\hat{l}_2, \hat{\mu}_2)$.

Given (F), (I), (N), (W), and (O), by Lemma 4.54 we have (X) $\text{UpdatePtr}(\hat{\sigma}, (\hat{l}, 0), [1, [(\hat{l}_2, \hat{\mu}_2)], [1], 1], \hat{bty*}) = (\hat{\sigma}_1, 1)$ such that (Y) $(\gamma,\ \sigma_1) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)$.

Given (G), (M), (P), (R), (V), and (X), by Vanilla C rule Pre-Increment Pointer we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, ++\ \hat{x}) \| \hat{C}) \Downarrow'_{(\text{p}, [\hat{pin1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \| \hat{C})$.

Given (Y), (W), and (J), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \| \hat{C})$.
By Definition 4.23 we have $pin1 \cong \hat{pin1}$, by Definition 4.25 we have $(\text{p}, [pin1]) \cong (\text{p}, [\hat{pin1}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin2])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin6])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin7])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin5])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ [\alpha,\ L_1,\ J,\ i]) \| C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \| C) \Downarrow^{(\text{p}, [(l, 0)])}_{(\text{p}, [pin1])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \| C)$. We use Lemma 4.58 in place of Lemma 4.57 to reason about the use of IncrementList to increment every location, whereas GetLocation increments the single location. As for the resulting location that is returned, we reason about the true location of the pointer being ψ-congruent to the Vanilla C location that is returned.

Case $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {++}\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin4])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ [n,\ L_1,\ J,\ 1]) \parallel C)$

This case is similar to Case $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {++}\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin5])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ [\alpha,\ L_1,\ J,\ i]) \parallel C)$.

Case $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{malloc}(e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[mal])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta,\ \mathrm{acc},\ (l,0)) \parallel C_1)$

Given (A) $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{malloc}(e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[mal])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta,\ \mathrm{acc},\ (l,0)) \parallel C_1)$ by SMC^2 rule Public Malloc, we have (B) $\mathrm{acc} = 0$, (C) $(e) \nvDash \gamma$, (D) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ n) \parallel C_1)$, (E) $l = \phi()$, and (F) $\sigma_2 = \sigma_1\big[l \to \big(\mathrm{NULL}, \mathrm{void}*, n, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{public}, n)\big)\big]$.

Given (G) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{malloc}(\hat{e})) \parallel \hat{C})$ and ψ such that (H) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{malloc}(e)) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{malloc}(\hat{e})) \parallel \hat{C})$, by Definition 4.22 we have (I) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (J) $C \cong_\psi \hat{C}$, and (K) $\mathrm{malloc}(e) \cong_\psi \mathrm{malloc}(\hat{e})$. Given (K), by Definition 4.20 we have (L) $e \cong_\psi \hat{e}$.

Given (D), (I), (L), and (J), by Lemma 4.2 we have (M) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (M), by the inductive hypothesis we have (N) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and ψ_1 such that (O) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (P) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (O), by Definition 4.22 we have (Q) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (R) $n \cong_{\psi_1} \hat{n}$, and (S) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (E), by Axiom 4.1 we have (T) $\hat{l} = \phi()$ and (U) $l = \hat{l}$.

Given (D), (C), and (R), by Lemmas 4.4 and 4.3 we have (V) $n = \hat{n}$.

Given (F), (Q), (U), and (V), by Lemma 4.13 we have (W) $\hat{\sigma}_2 = \hat{\sigma}_1\big[\hat{l} \to \big(\mathrm{NULL}, \mathrm{void}*, \hat{n}, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{public}, \hat{n})\big)\big]$ such that (X) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (G), (N), (T), and (W), by Vanilla C rule Malloc we have $\Sigma\triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{malloc}(\hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\hat{mal}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.

Given (X), (U), and (S), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta,\ \mathrm{acc},\ (l,0)) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.
By Definition 4.23 we have $mal \cong \hat{mal}$. Given (P), $\mathcal{D}_1 :: (\mathrm{p}, [mal])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{mal}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [mal]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{mal}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[malp])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta,\ \mathrm{acc},\ (l,0)) \parallel C_1)$

Given (A) $\Pi\triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[malp])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta,\ \mathrm{acc},\ (l,0)) \parallel C_1)$ by SMC^2 rule Private Malloc, we have (B) $(e) \nvDash \gamma$, (C) $(ty = \mathrm{private}\ bty*) \vee (ty = \mathrm{private}\ bty)$, (D) $\mathrm{acc} = 0$, (E) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ n) \parallel C_1)$, (F) $l = \phi()$, and (G) $\sigma_2 = \sigma_1\big[l \to \big(\mathrm{NULL},\ \mathrm{void}*,\ n \cdot \tau(ty), \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{private}, n \cdot \tau(ty))\big)\big]$.

Given (H) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{malloc}(\hat{e} \cdot \mathrm{sizeof}(\hat{ty}))) \parallel \hat{C})$ and ψ such that (I) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{pmalloc}(e,\ ty)) \parallel C)$

$\cong_\psi$ ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square$, malloc($\hat{e} \cdot$ sizeof($\hat{ty}$))) $\parallel \hat{C}$), by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (K) $C \cong_\psi \hat{C}$, and (L) pmalloc($e,\ ty$) $\cong_\psi$ malloc($\hat{e} \cdot$ sizeof($\hat{ty}$)). Given (L), by Definition 4.20 we have (M) $e \cong_\psi \hat{e}$ and (N) $ty \cong_\psi \hat{ty}$.

Given (H), we have (O) ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e} \cdot$ sizeof($\hat{ty}$)) $\parallel \hat{C}$).

Given (J), (K), and (M), by Lemma 4.2 we have (P) ((p, $\gamma,\ \sigma,\ \Delta,$ acc, e) $\parallel C$) $\cong_\psi$ ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}$) $\parallel \hat{C}$). Given (E) and (P), by the inductive hypothesis we have (Q) ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}$) $\parallel \hat{C}$) $\Downarrow'_{\hat{\mathcal{D}}_1}$ ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}$) $\parallel \hat{C}_1$) and ψ_1 such that (N) ((p, $\gamma, \sigma_1, \Delta_1$, acc, n) $\parallel C_1$) $\cong_{\psi_1}$ ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}$) $\parallel \hat{C}_1$) and (O) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (N), by Definition 4.22 we have (P) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (Q) $n \cong_{\psi_1} \hat{n}$ and (R) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (O) and (Q), we have (S) ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square$, sizeof($\hat{ty}$)) $\parallel \hat{C}_1$).

Given $\hat{ty}$, by Algorithm τ we have (T) $\hat{n}_1 = \tau(\hat{ty})$.

Given (S), (T), by Vanilla C rule Size of Type we have (U) ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square$, sizeof($\hat{ty}$)) $\parallel \hat{C}_1$) $\Downarrow'_{(\mathrm{p}, [\hat{ty}])}$ ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1$) $\parallel \hat{C}_1$).

Given (Q) and (U), we have (V) $\hat{n} * \hat{n}_1 = \hat{n}_2$.

Given (O), (Q), (U), and (V), by Vanilla C rule Multiplication we have (W) ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e} \cdot$ sizeof($\hat{ty}$)) $\parallel \hat{C}$) $\Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\hat{ty}])::(\mathrm{p},[\hat{bm}])}$ ((p, $\hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2$) $\parallel \hat{C}_1$).

Given (F), by Axiom 4.1 we have (X) $\hat{l} = \phi()$ and (Y) $l = \hat{l}$.

Given (B) and (Q), by (Z) $n = \hat{n}$.

Given (G), (Y), (P), (V), (T), and (Z) by Lemma 4.21 we have (A1) $\hat{\sigma}_2 = \hat{\sigma}_1[\hat{l} \to$ (NULL, void$*$, $\hat{n}_2$, PermL(Freeable, void$*$, public, $\hat{n}_2$))] such that (B1) $(\gamma,\ \sigma_2) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_2)$.

Given (H), (W), (X), and (A1), by Vanilla C rule Malloc we have $\Sigma \triangleright$ ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square$, malloc($\hat{e} \cdot$ sizeof($\hat{ty}$))) $\parallel \hat{C}$) $\Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p},[\hat{ty},\hat{bm}])::(\mathrm{p},[\hat{mal}])}$ ((p, $\hat{\gamma}, \hat{\sigma}_2, \square, \square, (\hat{l}, 0)$) $\parallel \hat{C}_1$).

Given (B1), (Y), and (R), by Definition 4.22 we have ((p, $\gamma,\ \sigma_2,\ \Delta,$ acc, $(l, 0)$) $\parallel C_1$) $\cong_{\psi_1}$ ((p, $\hat{\gamma}, \hat{\sigma}_2, \square, \square, (\hat{l}, 0)$) $\parallel \hat{C}_1$).
By Definition 4.23 we have $malp \cong [\hat{ty}, \hat{bm}, \hat{mal}]$.
Given (O), $\mathcal{D}_1$ :: (p, [$malp$]) and $\hat{\mathcal{D}}_1$ :: (p, [$\hat{ty}, \hat{bm}$]) :: (p, [$\hat{mal}$]), by Lemma 4.10 we have $\mathcal{D}_1$ :: (p, [$malp$]) $\cong$ $\hat{\mathcal{D}}_1$ :: (p, [$\hat{ty}, \hat{bm}, \hat{mal}$]).
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright$ ((p, $\gamma,\ \sigma,\ \Delta,$ acc, free(x)) $\parallel C$) $\Downarrow^{(\mathrm{p},[(l,0),(l_1,0)])}_{(\mathrm{p},[fre])}$ ((p, $\gamma,\ \sigma_1,\ \Delta,$ acc, skip) $\parallel C$)

Given (A) $\Pi \triangleright$ ((p, $\gamma,\ \sigma,\ \Delta,$ acc, free(x)) $\parallel C$) $\Downarrow^{(\mathrm{p},[(l,0),(l_1,0)])}_{(\mathrm{p},[fre])}$ ((p, $\gamma,\ \sigma_1,\ \Delta,$ acc, skip) $\parallel C$) by SMC^2 rule Public Free, we have (B) $\gamma(x) = (l,$ public $bty*)$, (C) $\sigma(l) = (\omega,$ public $bty*$, 1, PermL(Freeable, public $bty*$, public, 1)), (D) acc $= 0$, (E) DecodePtr(public $bty*$, 1, ω) $= [1,\ [(l_1, 0)],\ [1],\ 1]$, (F) CheckFreeable($\gamma$, $[(l_1, 0)]$, $[1]$, σ) $= 1$, and (G) Free($\sigma,\ l_1$) $= (\sigma_1, (l_1, 0))$.

Given (H) ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square$, free($\hat{x}$)) $\parallel \hat{C}$) and ψ such that (I) ((p, $\gamma,\ \sigma,\ \Delta,$ acc, free(x)) $\parallel C$) $\cong_\psi$ ((p, $\hat{\gamma}, \hat{\sigma}, \square, \square$,

$\text{free}(\hat{x})) \parallel \hat{C})$, by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (K) $C \cong_\psi \hat{C}$, and (L) $\text{free}(x) \cong_\psi \text{free}(\hat{x})$. Given (L), by Definition 4.20 we have (M) $x = \hat{x}$.

Given (B), (J), and (M), by Lemma 4.14 we have (N) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty*})$ such that (O) $l = \hat{l}$ and (P) public $bty* \cong_\psi \hat{bty*}$.

Given (C), (J), and (O), by Lemma 4.15 we have (Q) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty*}, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty*}, \text{public}, 1))$ such that (R) $\omega \cong_\psi \hat{\omega}$.

Given (E), (P), and (R), by Lemma 4.48 we have (S) $\text{DecodePtr}(\hat{bty*}, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (T) $[1, [(l_1, 0)], [1], 1] \cong_\psi [1, [(\hat{l}_1, 0)], [1], 1]$. Given (T), by Definition 4.15 we have (U) $l_1 \cong_\psi \hat{l}_1$.

Given (F), (J), and (U), by Axiom 4.2 we have (V) $\text{CheckFreeable}(\hat{\gamma}, [(\hat{l}_1, 0)], [1], \hat{\sigma}) = 1$.

Given (G), (J), and (U), by Lemma 4.59 we have (W) $\text{Free}(\hat{\sigma}, \hat{l}_1) = \hat{\sigma}_1$ such that (X) $(\gamma,\ \sigma_1) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)$.

Given (H), (N), (Q), (S), (V), and (W), by Vanilla C rule Free we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \text{free}(\hat{x})) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\hat{fre}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C})$.

Given (X) and (K), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \text{skip}) \parallel \hat{C})$. By Definition 4.23 we have $fre \cong \hat{fre}$, and by Definition 4.25 we have $(\text{p}, [fre]) \cong (\text{p}, [\hat{fre}])$. Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{pfree}(x)) \parallel C) \Downarrow^{(\text{p}, [(l,0), (l_1, 0)])}_{(\text{p}, [pfre])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{free}(x)) \parallel C) \Downarrow^{(\text{p}, [(l,0), (l_1, 0)])}_{(\text{p}, [fre])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)])}_{\mathcal{D}_1::(\text{p}, [cl1])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)])}_{\mathcal{D}_1::(\text{p}, [cl1])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$ by SMC^2 rule Cast Private Location, we have (B) $(ty = \text{private } bty*)$, (C) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$, (D) $\sigma_1 = \sigma_2\big[l \to \big(\omega,\ \text{void*},\ n, \text{PermL\_Ptr}(\text{Freeable}, \text{void*}, \text{private}, n)\big)\big]$, and (E) $\sigma_3 = \sigma_2\big[l \to \big(\omega,\ ty,\ \frac{n}{\tau(ty)}, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, \frac{n}{\tau(ty)})\big)\big]$.

Given (F) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$ and ψ such that (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (I) $C \cong_\psi \hat{C}$, and (J) $(ty)\ e \cong_\psi (\hat{ty})\ \hat{e}$. Given (J), by Definition 4.20 we have (K) $ty \cong_\psi \hat{ty}$ and (L) $e \cong_\psi \hat{e}$.

Given (B) and (K), by Definition 4.8 we have (M) $(\hat{ty} = \hat{bty*})$.

Given (H), (L), and (I), by Lemma 4.2 we have (N) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$ Given (C) and (N), by the inductive hypothesis we have (O) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$ and

ψ_1 such that (P) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$ (Q) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (P), by Definition 4.22 we have (R) $(\gamma,\ \sigma_1) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_1)$, (S) $C_1 \cong_{\psi} \hat{C}_1$, and (T) $(l, 0) \cong_{\psi_1} (\hat{l}, 0)$.

Given (D), (R), and (T), by Lemma 4.21 we have (U) $\hat{\sigma}_1 = \hat{\sigma}_2[\hat{l} \rightarrow (\hat{\omega}, \mathrm{void}, \hat{n}, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{void}, \mathrm{public}, \hat{n}))]$ (V) $(\gamma,\ \sigma_2) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_2)$, and (W) $\exists ty_1, \hat{ty}_1$ such that $ty_1 \cong \hat{ty}_1$ and $\frac{n}{\tau(ty_1)} = \frac{\hat{n}}{\tau(\hat{ty}_1)}$.

Given Axiom 4.15, (W), and (K), we have (X) $\frac{n}{\tau(ty)} = \frac{\hat{n}}{\tau(\hat{ty})}$.

Given (E), (V), (T), (K), and (X), by Lemma 4.13 we have (Y) $\hat{\sigma}_3 = \hat{\sigma}_2[\hat{l} \rightarrow (\hat{\omega}, \hat{ty}, \frac{\hat{n}}{\tau(\hat{ty})}, \mathrm{PermL}(\mathrm{Freeable}, \hat{ty}, \mathrm{public}, \frac{\hat{n}}{\tau(\hat{ty})}))]$ such that (Z) $(\gamma,\ \sigma_3) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_3)$.

Given (F), (M), (O), (U), and (Y), by Vanilla C rule Cast Location we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{cl}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.

Given (Z), (T), and (S), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.
By Definition 4.23 we have $cl1 \cong \hat{cl}$. Given (Q), $\mathcal{D}_1 :: (\mathrm{p}, [cl1])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{cl}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [cl1]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{cl}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [cl])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [cl])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1)$ by SMC^2 rule Cast Public Location, we have $\mathrm{acc} = 0$, (B) $(ty = \mathrm{public}\ bty*)$, (C) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1)$, (D) $\sigma_1 = \sigma_2\big[l \rightarrow \big(\omega,\ \mathrm{void}*,\ n,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{void}*, \mathrm{public}, n)\big)\big]$, and (E) $\sigma_3 = \sigma_2\big[l \rightarrow \big(\omega,\ ty,\ \frac{n}{\tau(ty)}, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, ty, \mathrm{public}, \frac{n}{\tau(ty)})\big)\big]$.

Given (F) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$ and ψ such that (G) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (I) $C \cong_{\psi} \hat{C}$, and (J) $(ty)\ e \cong_{\psi} (\hat{ty})\ \hat{e}$. Given (J), by Definition 4.20 we have (K) $ty \cong_{\psi} \hat{ty}$ and (L) $e \cong_{\psi} \hat{e}$.

Given (B) and (K), by Definition 4.8 we have (M) $(\hat{ty} = \hat{bty}*)$.

Given (H), (L), and (I), by Lemma 4.2 we have (N) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$ Given (C) and (N), by the inductive hypothesis we have (O) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$ and

ψ_1 such that (P) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$ (Q) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (P), by Definition 4.22 we have (R) $(\gamma,\ \sigma_1) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_1)$, (S) $C_1 \cong_{\psi} \hat{C}_1$, and (T) $(l, 0) \cong_{\psi_1} (\hat{l}, 0)$.

Given (T), by Lemma 4.17 we have (U) $l = \hat{l}$.

Given (D), (R), and (U), by Lemma 4.20 we have (V) $\hat{\sigma}_1 = \hat{\sigma}_2[\hat{l} \to (\hat{\omega}, \text{void}*, \hat{n}, \text{PermL}(\text{Freeable}, \text{void}*, \text{public}, \hat{n}))]$ such that (W) $(\gamma,\ \sigma_2) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_2)$ and (X) $n = \hat{n}$.

Given (B) and (K), by Lemma 4.22 we have (Y) $\tau(ty) = \tau(\hat{ty})$.

Given (E), (X), (Y), (U), and (W) by Lemma 4.13 we have (Z) $\hat{\sigma}_3 = \hat{\sigma}_2[\hat{l} \to (\hat{\omega}, \hat{ty}, \frac{\hat{n}}{\tau(\hat{ty})}, \text{PermL}(\text{Freeable}, \hat{ty}, \text{public}, \frac{\hat{n}}{\tau(\hat{ty})}))]$ such that (A1) $(\gamma,\ \sigma_3) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma}_3)$.

Given (F), (M), (O), (U), and (Z), by Vanilla C rule Cast Location we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{cl}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.

Given (A1), (T), and (S), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, (\hat{l}, 0)) \parallel \hat{C}_1)$.
By Definition 4.23 we have $cl \cong \hat{cl}$. Given (Q), $\mathcal{D}_1 :: (\text{p}, [cl])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{cl}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [cl]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{cl}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [cv])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1 :: (\text{p}, [cv])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)$ by SMC² rule Cast Public Value, we have (B) $(e) \nvdash \gamma$, (C) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$, (D) $(ty = \text{public}\ bty)$, and (E) $n_1 = \text{Cast}(\text{public},\ ty,\ n)$.

Given (F) $((\text{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$ and ψ such that (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (I) $C \cong_{\psi} \hat{C}$, and (J) $(ty)\ e \cong_{\psi} (\hat{ty})\ \hat{e}$. Given (J), by Definition 4.20 we have (K) $ty \cong_{\psi} \hat{ty}$ (L) $e \cong_{\psi} \hat{e}$.

Given (H), (L), and (I), by Lemma 4.2 we have (M) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (M), by the inductive hypothesis we have (N) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and ψ_1 such that (O) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (P) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (O), by Definition 4.22 we have (Q) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (R) $n \cong_{\psi_1} \hat{n}$ and (S) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (B), (C), and (R), by Lemmas 4.4 and 4.3 we have (T) $n = \hat{n}$.

Given (E), (K), and (T), by Lemma 4.60 we have (U) $\hat{n}_1 = \text{Cast}(\text{public}, \hat{ty}, \hat{n})$ such that (V) $n_1 \cong_{\psi_1} \hat{n}_1$.

Given (F), (M), and (U), by Vanilla C rule Cast Value we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, (\hat{ty})\ \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{cv}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$.

Given (Q), (V), and (S), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_1) \parallel \hat{C}_1)$.
By Definition 4.23 we have $cv \cong \hat{cv}$. Given (P), $\mathcal{D}_1 :: (\text{p}, [cv])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{cv}])$, by Lemma 4.10 we have

$\mathcal{D}_1 :: (\mathrm{p}, [cv]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{cv}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1::(\mathrm{p},[cv1])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, (ty)\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1::(\mathrm{p},[cv])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$. The main difference is using Lemma 4.61 in place of Lemma 4.60, as we are reasoning about private values that are congruent, whereas the previous case has public values that are equivalent.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \Downarrow^{\epsilon}_{(\mathrm{p},[loc])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l, 0)) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \Downarrow^{\epsilon}_{(\mathrm{p},[loc])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l, 0)) \parallel C)$ by SMC^2 rule Address Of, we have (B) $\gamma(x) = (l,\ ty)$.

Given (C) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \&\hat{x}) \parallel \hat{C})$ and ψ such that (D) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \&\hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (E) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (F) $C \cong_{\psi} \hat{C}$, and (G) $\&x \cong_{\psi} \&\hat{x}$. Given (G), by Definition 4.20 we have (H) $x = \hat{x}$.

Given (B), (E), and (H), by Lemma 4.14 we have (I) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{ty})$ such that (J) $l = \hat{l}$ and (K) $ty \cong_{\psi} \hat{ty}$.

Given (C) and (I), by Vanilla C rule Address Of we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \&\hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\hat{loc}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}, 0)) \parallel \hat{C})$.

Given (E), (J), and (F), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l, 0)) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}, 0)) \parallel \hat{C})$.
By Definition 4.23 we have $loc \cong \hat{loc}$, and by Definition 4.25 we have $(\mathrm{p}, [loc]) \cong (\mathrm{p}, [\hat{loc}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \Downarrow^{\epsilon}_{(\mathrm{p},[ty])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \Downarrow^{\epsilon}_{(\mathrm{p},[ty])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$ by SMC^2 rule Size of Type, we have (B) $(ty) \nvdash \gamma$ and (C) $n = \tau(ty)$.

Given (D) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{sizeof}(\hat{ty})) \parallel \hat{C})$ and ψ such that (E) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{sizeof}(\hat{ty})) \parallel \hat{C})$, by Definition 4.22 we have (F) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (G) $C \cong_{\psi} \hat{C}$, and (H) $\mathrm{sizeof}(ty) \cong_{\psi} \mathrm{sizeof}(\hat{ty})$. Given (H), by Definition 4.20 we have (I) $ty \cong_{\psi} \hat{ty}$.

Given (B), (C), and (I), by Lemma 4.22 we have (J) $\hat{n} = \tau(\hat{ty})$ and (K) $n = \hat{n}$.

Given (D) and (J), by Vanilla C rule Size of Type we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{sizeof}(\hat{ty})) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\hat{ty}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

Given (F), (K), and (G), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

By Definition 4.23 we have $ty \cong \hat{ty}$, and by Definition 4.25 we have $(\mathrm{p}, [ty]) \cong (\mathrm{p}, [\hat{ty}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [inp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [inp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$ by SMC^2 rule SMC Input Public Value, we have (B) $(e) \nvDash \gamma$, (C) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$, (D) $\gamma(x) = (l, \mathrm{public}\ bty)$, (E) $\mathrm{acc} = 0$, (F) $\mathrm{InputValue}(x, n) = n_1$, and (G) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, x = n_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)$.

Given (H) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcinput}(\hat{x}, \hat{e})) \parallel \hat{C})$ and ψ such that (I) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcinput}(\hat{x}, \hat{e})) \parallel \hat{C})$, by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, (K) $\mathrm{smcinput}(x,\ e) \cong_{\psi} \mathrm{mcinput}(\hat{x}, \hat{e})$, and (L) $C \cong_{\psi} \hat{C}$. Given (K), by Definition 4.20 we have (M) $e \cong_{\psi} \hat{e}$ and $x \cong_{\psi} \hat{x}$ such that (N) $x = \hat{x}$.

Given (J), (L), and (M), by Lemma 4.2 we have (O) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc}, e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, \hat{e}) \parallel \hat{C})$. Given (C) and (O), by the inductive hypothesis we have (P) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma},\ \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and ψ_1 such that (Q) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \quad \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$. (R) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (Q), by Definition 4.22 we have (S) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$ (T) $n \cong_{\psi_1} \hat{n}$, and (U) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (T) and (B), by Lemmas 4.4 and 4.3 we have (V) $n = \hat{n}$.

Given (D), (J), and (N), by Lemma 4.14 we have (W) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (X) $l = \hat{l}$ and (Y) $\mathrm{public}\ bty \cong_{\psi_1} \hat{bty}$.

Given (F), (N), and (V), by Lemma 4.23 we have (Z) $\mathrm{InputValue}(\hat{x}, \hat{n}) = \hat{n}_1$ such that (A1) $n_1 \cong_{\psi_1} \hat{n}_1$.

Given (A1) and (N), by Definition 4.20 we have (B1) $x = n_1 \cong_{\psi_1} \hat{x} = \hat{n}_1$.

Given (S), (B1), and (U), by Lemma 4.2 we have (C1) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, x = n_1) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{x} = \hat{n}_1) \parallel \hat{C}_1)$. Given (G) and (C1), by the inductive hypothesis we have (D1) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{x} = \hat{n}_1) \parallel \hat{C}_1) \Downarrow'_{\hat{\mathcal{D}}_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$ and ψ_2 such that (E1) $((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. Given (E1), by Definition 4.22 we have (F1) $(\gamma, \sigma_2) \cong_{\psi_2} (\hat{\gamma}, \hat{\sigma}_2)$ (G1) $C_2 \cong_{\psi_2} \hat{C}_2$, and (H1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$.

Given (H), (P), (W), (Z), and (D1), by Vanilla C rule Input Value we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcinput}(\hat{x}, \hat{e})) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{inp}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$.

Given (F1) and (G1), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2) \cong_{\psi_2} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_2)$. By Definition 4.23 we have $inp \cong \hat{inp}$. Given (R), (H1), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [inp])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{inp}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [inp]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{p}, [\hat{inp}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [inp2])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[inp])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e_1,\ e_2)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[inp1])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_3)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[inp])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$. The difference is an additional use of the inductive hypothesis to evaluate e_2, which contains the length of the array to be read in, and Lemma 4.24 to reason about the use of InputArray in place of Lemma 4.23 to reason about the use of InputValue.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e_1,\ e_2)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[inp3])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_3)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcinput}(x,\ e_1,\ e_2)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[inp1])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_3)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e)) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0)])}_{\mathcal{D}_1::(\mathrm{p},[out2])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e)) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0)])}_{\mathcal{D}_1::(\mathrm{p},[out2])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC^2 rule SMC Output Private Value, we have (B) $(e) \nvdash \gamma$, (C) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)$, (D) $\gamma(x) = (l,\ \mathrm{private}\ bty)$, (E) $\sigma_1(l) = (\omega,\ \mathrm{private}\ bty,\ 1,\ \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private}\ bty, \mathrm{private}, 1))$, (F) $\mathrm{DecodeVal}(\mathrm{private}\ bty,\ \omega) = n_1$, and (G) $\mathrm{OutputValue}(x,\ n, n_1)$.

Given (H) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcoutput}(\hat{x}, \hat{e})) \parallel \hat{C})$ and ψ such that (I) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{smcoutput}(x,\ e)) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcoutput}(\hat{x}, \hat{e})) \parallel \hat{C})$, by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, (K) $\mathrm{smcoutput}(x,\ e) \cong_\psi \mathrm{mcoutput}(\hat{x}, \hat{e})$, and (L) $C \cong_\psi \hat{C}$. Given (K), by Definition 4.20 we have (M) $e \cong_\psi \hat{e}$ and $x \cong_\psi \hat{x}$ such that (N) $x = \hat{x}$.

Given (J), (M), and (L), by Lemma 4.2 we have (O) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \cong_\psi ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (C) and (O), by the inductive hypothesis we have (P) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and

ψ_1 such that (Q) $((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ n) \ \ \| \ C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \ \| \ \hat{C}_1)$ and (R) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (Q), by Definition 4.22 we have (S) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$ (T) $n \cong_{\psi_1} \hat{n}$, and (U) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (D), (S), and (N), by Lemma 4.14 we have (V) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (W) $l = \hat{l}$ and (X) private $bty \cong_{\psi_1} \hat{bty}$.

Given (E), (S), and (W), by Lemma 4.15 we have (Y) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}, 1, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, 1))$ such that (Z) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (F), (X), and (Z), by Lemma 4.45 we have (A1) $\mathrm{DecodeVal}(\hat{bty}, \hat{\omega}) = \hat{n}_1$ such that (B1) $n_1 \cong_{\psi_1} \hat{n}_1$.

Given (B), (C), and (T), by Lemmas 4.4 and 4.3 we have (C1) $n = \hat{n}$.

Given (G), (N), (C1), and (B1), by Lemma 4.25 we have (D1) $\mathrm{OutputValue}(\hat{x}, \hat{n}, \hat{n}_1)$ such that we have congruent output files.

Given (H), (P), (V), (Y), (A1), and (D1), by Vanilla C rule Output Value we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{mcoutput}(\hat{x}, \hat{e})) \ \| \ \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{out}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \ \| \ \hat{C}_1)$.

Given (S) and (U), by Definition 4.22 we have $((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ \mathrm{skip}) \ \| \ C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}) \ \| \ \hat{C}_1)$. By Definition 4.23 we have $out2 \cong \hat{out}$. Given (R), $\mathcal{D}_1 :: (\mathrm{p}, [out2])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{out}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [out2]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{out}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ \mathrm{smcoutput}(x, \ e)) \ \| \ C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [out])} ((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ \mathrm{skip}) \ \| \ C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ \mathrm{smcoutput}(x, \ e)) \ \| \ C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [out2])} ((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ \mathrm{skip}) \ \| \ C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{smcoutput}(x, e_1, e_2)) \ \| \ C) \Downarrow^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l, 0), (l_1, 0), \ldots, (l_1, \alpha - 1)])}_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [out1])} ((\mathrm{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \mathrm{acc}, \ \mathrm{skip}) \ \| \ C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ \mathrm{smcoutput}(x, \ e)) \ \| \ C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [out2])} ((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ \mathrm{skip}) \ \| \ C_1)$. The difference is an additional use of the inductive hypothesis to evaluate e_2, which contains the length of the array to be output, additional handling of the constant pointer to the array data and reading the entire array, and Lemma 4.26 to reason about the use of OutputArray in place of Lemma 4.25 to reason about the use of OutputValue. The handling of reading the array is similar to what is shown in Case $\Pi \triangleright ((\mathrm{p}, \gamma, \ \sigma, \ \Delta, \ \mathrm{acc}, \ x[e]) \ \| \ C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0), (l_1, i)])}_{\mathcal{D}_1 :: (\mathrm{p}, [ra])} ((\mathrm{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \mathrm{acc}, \ n_i) \ \| \ C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{smcoutput}(x, e_1, e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[out3])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{smcoutput}(x, e_1, e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[out1])}^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}$ $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[rp])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l_1, \mu_1)) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[rp])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l_1, \mu_1)) \parallel C)$ by SMC^2 rule Pointer Read Single Location, we have (B) $\gamma(x) = (l,\ a\ bty*)$, (C) $\sigma(l) = (\omega,\ a\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a\ bty*, a, 1))$, and (D) $\mathrm{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i]$.

Given (E) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$ and ψ such that (F) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (G) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (H) $C \cong_{\psi} \hat{C}$, and (I) $x \cong_{\psi} \hat{x}$. Given (I), by Definition 4.20 we have (J) $x = \hat{x}$.

Given (B), (G), and (J), by Lemma 4.14 we have (K) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (L) $l = \hat{l}$ and (M) $a\ bty* \cong_{\psi} \hat{bty}*$.

Given (C), (G), and (L), by Lemma 4.16 we have (N) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1))$ such that (O) $\omega \cong_{\psi} \hat{\omega}$.

Given (D), (M), and (O), by Lemma 4.48 we have (P) $\mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (Q) $[1,\ [(l_1, \mu_1)],\ [1],\ i] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (Q), by Definition 4.15 we have (R) $(l_1, \mu_1) \cong_{\psi} (\hat{l}_1, \hat{\mu}_1)$.

Given (E), (K), (N), and (P), by Vanilla C rule Pointer Read Location we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C}) \Downarrow'_{(\mathrm{p},[\hat{rp}])}$ $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_1, \hat{\mu}_1)) \parallel \hat{C})$.

Given (G), (R), and (H), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l_1, \mu_1)) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_1, \hat{\mu}_1)) \parallel \hat{C})$.
By Definition 4.23 we have $rp \cong \hat{rp}$, and by Definition 4.25 we have $(\mathrm{p}, [rp]) \cong (\mathrm{p}, [\hat{rp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[rp1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [\alpha,\ L,\ J,\ i]) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[rp1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [\alpha,\ L,\ J,\ i]) \parallel C)$ by SMC^2 rule Private Pointer Read Multiple Locations, we have (B) $\gamma(x) = (l,\ \mathrm{private}\ bty*)$, (C) $\sigma(l) = (\omega,\ \mathrm{private}\ bty*,\ \alpha,$ $\mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, \alpha))$, (D) $(bty = \mathrm{int}) \vee (bty = \mathrm{float})$, and (E) $\mathrm{DecodePtr}(\mathrm{private}\ bty*,$ $\alpha,\ \omega) = [\alpha,\ L,\ J,\ i]$.

Given (F) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$ and ψ such that (G) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C})$, by

Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (I) $C \cong_\psi \hat{C}$, and (J) $x \cong_\psi \hat{x}$. Given (J), by Definition 4.20 we have (K) $x = \hat{x}$.

Given (B), (H), and (K), by Lemma 4.14 we have (L) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (M) $l = \hat{l}$ and (N) private $bty* \cong_\psi \hat{bty}*$.

Given (C), (H), and (M), by Lemma 4.16 we have (O) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (P) $\omega \cong_\psi \hat{\omega}$.

Given (D), (N), and (P), by Lemma 4.48 we have (Q) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (R) $[\alpha,\ L,\ J,\ i] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (R), by Lemma 4.27 we have (S) $[\alpha,\ L,\ J,\ i] \cong_\psi (\hat{l}_1, \hat{\mu}_1)$.

Given (F), (L), (O), and (Q), by Vanilla C rule Pointer Read Location we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p},[\hat{rp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_1, \hat{\mu}_1)) \parallel \hat{C})$.

Given (H), (S), and (I), by Lemma 4.27 we have $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ [\alpha,\ L,\ J,\ i]) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_1, \hat{\mu}_1)) \parallel \hat{C})$.
By Definition 4.23 we have $rp1 \cong \hat{rp}$, and by Definition 4.25 we have $(\text{p}, [rp1]) \cong (\text{p}, [\hat{rp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x) \parallel C) \Downarrow_{(\text{p},[rdp])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x) \parallel C) \Downarrow_{(\text{p},[rdp])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C)$ by SMC^2 rule Pointer Dereference Single Location, we have (B) $\gamma(x) = (l,\ a\ bty*)$, (C) $\sigma(l) = (\omega,\ a\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1))$, (D) $\text{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1]$, and (E) $\text{DerefPtr}(\sigma, a\ bty, (l_1, \mu_1)) = (n, 1)$.

Given (F) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$ and ψ such that (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (H) $(\gamma,\ \sigma) \cong_\psi (\hat{\gamma},\ \hat{\sigma})$, (I) $C \cong_\psi \hat{C}$, and (J) $*x \cong_\psi *\hat{x}$. Given (J), by Definition 4.20 we have (K) $x = \hat{x}$.

Given (B), (H), and (K), by Lemma 4.14 we have (L) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (M) $l = \hat{l}$ and (N) $a\ bty* \cong_\psi \hat{bty}*$.

Given (C), (H), and (M), by Lemma 4.15 we have (O) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (P) $\omega \cong_\psi \hat{\omega}$.

Given (D), (N), and (P), by Lemma 4.48 we have (Q) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$ (R) $[1,\ [(l_1, \mu_1)],\ [1],\ 1] \cong_\psi [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$. Given (R), by Definition 4.15 we have (S) $(l_1, \mu_1) \cong_\psi (\hat{l}_1, \hat{\mu}_1)$.

Given (E), (H), (N), and (S), by Lemma 4.62 we have (T) $\text{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)$ such that (U) $n \cong_\psi \hat{n}$.

Given (F), (L), (O), (Q), and (T), by Vanilla C rule Pointer Dereference we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p},[\hat{rdp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

Given (H), (U), and (I), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ n) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \hat{C})$.

By Definition 4.23 we have $rdp \cong \hat{rdp}$, and by Definition 4.25 we have $(\text{p}, [rdp]) \cong (\text{p}, [\hat{rdp}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow_{(\text{p},[rdp1])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_2, \mu_2)) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow_{(\text{p},[rdp1])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_2, \mu_2)) \parallel C)$ by SMC^2 rule Pointer Dereference Single Location Higher Level Indirection, we have (B) $\gamma(x) = (l, \ a\ bty*)$, (C) $\sigma(l) = (\omega_1, \ a\ bty*, \ 1, \ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1))$, (D) $\text{DecodePtr}(a\ bty*, \ 1, \ \omega) = [1, \ [(l_1, \mu_1)], \ [1], \ i]$, (E) $\text{DerefPtrHLI}(\sigma, a\ bty*, (l_1, \mu_1)) = ([1, \ [(l_2, \mu_2)], \ [1], \ i - 1], 1)$, and (F) $i > 1$.

Given (G) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$ and ψ such that (H) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$, by Definition 4.22 we have (I) $(\gamma, \sigma) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})$, (J) $C \cong_{\psi} \hat{C}$, and (K) $*x \cong_{\psi} *\hat{x}$. Given (K), by Definition 4.20 we have (L) $x = \hat{x}$.

Given (B), (I), and (L), by Lemma 4.14 we have (M) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (N) $l = \hat{l}$ and (O) $a\ bty* \cong_{\psi} \hat{bty}*$.

Given (C), (I), and (N), by Lemma 4.15 we have (P) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (Q) $\omega \cong_{\psi} \hat{\omega}$.

Given (D), (O), and (Q), by Lemma 4.48 we have (R) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (S) $[1, \ [(l_1, \mu_1)], \ [1], \ i] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (S), by Definition 4.15 we have (T) $(l_1, \mu_1) \cong_{\psi} (\hat{l}_1, \hat{\mu}_1)$ and (U) $i = \hat{i}$.

Given (F) and (U), we have (V) $\hat{i} > 1$.

Given (E), (I), (O), and (T), by Lemma 4.63 we have (W) $\text{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], 1)$ such that (X) $[1, \ [(l_2, \mu_2)], \ [1], \ i - 1] \cong_{\psi} [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i} - 1]$. Given (X), by Definition 4.15 we have (Y) $(l_2, \mu_2) \cong_{\psi} (\hat{l}_2, \hat{\mu}_2)$.

Given (G), (M), (P), (R), (V), and (W), by Vanilla C rule Pointer Dereference Higher Level Indirection we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p},[\hat{rdp1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})$.

Given (I), (Y), and (J), by Definition 4.22 we have $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_2, \mu_2)) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})$. By Definition 4.23 we have $rdp1 \cong \hat{rdp1}$, and by Definition 4.25 we have $(\text{p}, [rdp1]) \cong (\text{p}, [\hat{rdp1}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow_{(\text{p},[rdp2])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, [\alpha, \ L, \ J, \ i - 1]) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \Downarrow_{(\text{p},[rdp2])}^{(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, [\alpha, \ L, \ J, \ i - 1]) \parallel C)$ by SMC^2 rule Pointer Dereference Single Location Higher Level Indirection, we have (B) $\gamma(x) = (l, \ \text{private}\ bty*)$, (C) $\sigma(l) = (\omega_1, \ \text{private}\ bty*, \ 1, \ \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, 1))$, (D) $\text{DecodePtr}(\text{private}\ bty*, \ 1, \ \omega) = [1, \ [(l_1, \mu_1)], \ [1], \ i]$, (E) $\text{DerefPtrHLI}(\sigma, \text{private}\ bty*, (l_1, \mu_1)) = ([\alpha, \ L, \ J, \ i - 1], 1)$, and (F) $i > 1$.

Given (G) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$ and ψ such that (H) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C})$, by

Definition 4.22 we have (I) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (J) $C \cong_{\psi} \hat{C}$, and (K) $*x \cong_{\psi} *\hat{x}$. Given (K), by Definition 4.20 we have (L) $x = \hat{x}$.

Given (B), (I), and (L), by Lemma 4.14 we have (M) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (N) $l = \hat{l}$ and (O) private $bty* \cong_{\psi} \hat{bty}*$.

Given (C), (I), and (N), by Lemma 4.15 we have (P) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (Q) $\omega \cong_{\psi} \hat{\omega}$.

Given (D), (O), and (Q), by Lemma 4.48 we have (R) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (S) $[1,\ [(l_1, \mu_1)],\ [1],\ i] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (S), by Definition 4.15 we have (T) $(l_1, \mu_1) \cong_{\psi} (\hat{l}_1, \hat{\mu}_1)$ and (U) $i = \hat{i}$.

Given (F) and (U), we have (V) $\hat{i} > 1$.

Given (E), (I), (O), and (T), by Lemma 4.63 we have (W) $\text{DerefPtrHLI}(\hat{\sigma}, \hat{bty}*, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], 1)$ such that (X) $[\alpha,\ L,\ J,\ i-1] \cong_{\psi} [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1]$. Given (X), by Lemma 4.27 we have (Y) $[\alpha,\ L,\ J,\ i-1] \cong_{\psi} (\hat{l}_2, \hat{\mu}_2)$.

Given (G), (M), (P), (R), (V), and (W), by Vanilla C rule Pointer Dereference Higher Level Indirection we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \hat{C}) \Downarrow'_{(\text{p}, [\hat{rdp1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})$.

Given (I), (Y), and (J), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ [\alpha,\ L,\ J,\ i-1]) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel \hat{C})$.
By Definition 4.23 we have $rdp2 \cong \hat{rdp1}$, and by Definition 4.25 we have $(\text{p}, [rdp2]) \cong (\text{p}, [\hat{rdp1}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)])}_{\mathcal{D}_1::(\text{p}, [wp1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)])}_{\mathcal{D}_1::(\text{p}, [wp1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Write, we have (B) $(e) \nvdash \gamma$, (C) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l_e, \mu_e)) \parallel C_1)$, (D) $\gamma(x) = (l,\ \text{private}\ bty*)$, (E) $\sigma_1(l) = (\omega,\ \text{private}\ bty*,\ \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, \alpha))$, (F) $\text{DecodePtr}(\text{private}\ bty*,\ \alpha,\ \omega) = [\alpha,\ L,\ J,\ i]$, and (G) $\text{UpdatePtr}(\sigma_1,\ (l, 0),\ [1,\ [(l_e, \mu_e)],\ [1],\ i],\ \text{private}\ bty*) = (\sigma_2,\ 1)$.

Given (H) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that (I) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C})$, by Definition 4.22 we have (J) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (K) $C \cong_{\psi} \hat{C}$, and (L) $x = e \cong_{\psi} \hat{x} = \hat{e}$. Given (M), by Definition 4.20 we have (M) $e \cong_{\psi} \hat{e}$ and (N) $x = \hat{x}$.

Given (J), (M), and (K), by Lemma 4.2 we have (O) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (C) and (O), by the inductive hypothesis we have (P) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1)$

and ψ_1 such that (Q) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l_e, \mu_e)) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1)$ and (R) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (Q), by Definition 4.22 we have (S) $(\gamma,\ \sigma_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)$, (T) $(l_e, \mu_e) \cong_{\psi_1} (\hat{l}_e, \hat{\mu}_e)$, and (U) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (D), (S), and (N), by Lemma 4.14 we have (V) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (W) $l = \hat{l}$ and (X) private $bty* \cong_{\psi_1} \hat{bty}*$.

Given (E), (S), and (W), by Lemma 4.15 we have (Y) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (Z) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (F), (X), and (Z), by Lemma 4.48 we have (A1) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (B1) $[\alpha,\ L,\ J,\ i] \cong_{\psi_1} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (B1), by Definition 4.15 we have (C1) $i = \hat{i}$. Given (T) and (C1), by Definition 4.15 we have (D1) $[1,\ [(l_e, \mu_e)],\ [1],\ i] \cong_{\psi_1} 1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}]$.

Given (G), (S), (W), (D1), and (X), by Lemma 4.54 we have (E1) $\text{UpdatePtr}(\hat{\sigma}_1, (\hat{l}, 0), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}], \hat{bty}*) = (\hat{\sigma}_2, 1)$ such that (F1) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (H), (P), (V), (Y), (A1), and (E1), by Vanilla C rule Pointer Write Location we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\text{p},[\hat{wp}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)$.

Given (F1) and (U), by Definition 4.22 we have $((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $wp1 \cong \hat{wp}$. Given (R), $\mathcal{D}_1 :: (\text{p}, [wp1])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{wp}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [wp1]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{wp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[wp])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[wp1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[wp2])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[wp1])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p},[wdp3])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p},[wdp3])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Dereference Write Single Location Private Value, we have (B) $(e) \vdash \gamma$, (C) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$, (D) $\gamma(x) = (l,\ \text{private } bty*)$, (E) $\sigma_1(l) = (\omega,\ \text{private } bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, 1))$, (F) $(bty = \text{int}) \vee (bty = \text{float})$,
(G) $\text{DecodePtr}(\text{private } bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1]$, (H) $\text{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \text{acc}, \text{private } bty) = (\Delta_2, L_1)$, and (I) $\text{UpdateOffset}(\sigma_1,\ (l_1, \mu_1),\ n, \text{private } bty) = (\sigma_2, 1)$.

Given (J) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that (K) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ *x = e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})\ \parallel \hat{C})$ by Definition 4.22 we have (L) $(\gamma,\ \sigma) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})$, (M) $C \cong_{\psi} \hat{C}$, and (N) $*x = e \cong_{\psi} *\hat{x} = \hat{e}$. Given (N), by Definition 4.20 we have (O) $e \cong_{\psi} \hat{e}$ and (P) $x = \hat{x}$.

Given (L), (O), and (M), by Lemma 4.2 we have (Q) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \cong_{\psi} ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$. Given (C) and (Q), by the inductive hypothesis we have (R) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and

ψ_1 such that (S) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel \hat{C}_1)$ and (T) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (S), by Definition 4.22 we have (U) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (V) $n \cong_{\psi_1} \hat{n}$ and (W) $C_1 \cong_{\psi_1} \hat{C}_1$.

Given (D), (U), and (P), by Lemma 4.14 we have (X) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (Y) $l = \hat{l}$ and (Z) private $bty* \cong_{\psi_1} \hat{bty}*$.

Given (E), (U), and (Y), by Lemma 4.15 we have (A1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1))$ such that (B1) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (G), (Z), and (B1), by Lemma 4.48 we have (C1) $\mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$ such that (D1) $[1,\ [(l_1, \mu_1)],\ [1],\ 1] \cong_{\psi_1} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$. Given (D1), by Definition 4.15 we have (E1) $(l_1, \mu_1) \cong_{\psi_1} (\hat{l}_1, \hat{\mu}_1)$.

Given (Z), by Definition 4.8 we have (F1) private $bty \cong_{\psi_1} \hat{bty}$.

Given (I), (U), (E1), (V), and (F1), by Lemma 4.64 we have (G1) $\mathrm{UpdateOffset}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, 1)$ such that (H1) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (J), (R), (X), (A1), (C1), and (G1), by Vanilla C rule Pointer Dereference Write Value we have $\Sigma \triangleright ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{p}, [\hat{wdp}])} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$.

Given (H1) and (W), by Definition 4.22 we have $((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1) \cong_{\psi_1} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $wdp3 \cong \hat{wdp}$. Given (T), $\mathcal{D}_1 :: (\mathrm{p}, [wdp3])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{wdp}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{p}, [wdp3]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{p}, [\hat{wdp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0), (l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0)]::L_1::[(l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp3])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0)]::L_1::[(l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp4])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc}, \mathrm{skip}) \parallel C_1)$.

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0)]::L_1::[(l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp3])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$. Given $n = \hat{n}$, we use Definition 4.19 to prove that $\mathrm{encrypt}(n) \cong \hat{n}$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0)]::L_1::[(l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp2])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p}, [(l,0)]::L_1::[(l_1, \mu_1)])}_{\mathcal{D}_1::(\mathrm{p}, [wdp2])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Dereference Write Multiple Locations to Single Location Higher Level Indirection, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ [\alpha, L_e, J_e, i{-}1]) \parallel C_1)$, (C) $\gamma(x) = (l,\ \mathrm{private}\ bty*)$, (D) $\sigma_1(l) = (\omega,\ \mathrm{private}\ bty*,\ 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, 1))$, (E) $\mathrm{DecodePtr}(\mathrm{private}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i]$, (F) $i > 1$, (G) $\mathrm{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \mathrm{acc}, \mathrm{private}\ bty*) = (\Delta_2, L_1)$, and (H) $\mathrm{UpdatePtr}(\sigma_1,\ (l_1, \mu_1),\ [\alpha, L_e, J_e, i-1], \mathrm{private}\ bty*) = (\sigma_2, 1)$.

Given (I) $((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C})$ and ψ such that (J) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \cong_{\psi} ((\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square,$

$*\hat{x} = \hat{e}) \parallel \hat{C})$ by Definition 4.22 we have (K) $(\gamma, \sigma) \cong_\psi (\hat{\gamma}, \hat{\sigma})$, (L) $C \cong_\psi \hat{C}$, and (M) $*x = e \cong_\psi *\hat{x} = \hat{e}$. Given (M), by Definition 4.20 we have (N) $e \cong_\psi \hat{e}$ and (O) $x = \hat{x}$.

Given (K), (N), and (L), by Lemma 4.2 we have (P) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \cong_\psi ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C})$ Given (B) and (P), by the inductive hypothesis we have (Q) $((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1)$ and ψ_1 such that (R) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, [\alpha, L_e, J_e, i-1]) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)) \parallel \hat{C}_1)$ and (S) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (R), by Definition 4.22 we have (T) $(\gamma, \sigma_1) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)$, (U) $[\alpha, L_e, J_e, i-1] \cong_{\psi_1} (\hat{l}_e, \hat{\mu}_e)$ and (V) $C_1 \cong_{\psi_1} \hat{C}_1$. Given (U), by Definition 4.15 we have (W) $[\alpha, L_e, J_e, i-1] \cong_{\psi_1} [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}-1]$

Given (C), (T), and (O), by Lemma 4.14 we have (X) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (Y) $l = \hat{l}$ and (Z) private $bty* \cong_{\psi_1} \hat{bty}*$.

Given (D), (T), and (Y), by Lemma 4.15 we have (A1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (B1) $\omega \cong_{\psi_1} \hat{\omega}$.

Given (E), (Z), and (B1), by Lemma 4.48 we have (C1) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (D1) $[1, [(l_1, \mu_1)], [1], i] \cong_{\psi_1} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (D1), by Definition 4.15 we have (E1) $(l_1, \mu_1) \cong_{\psi_1} (\hat{l}_1, \hat{\mu}_1)$ and (F1) $i = \hat{i}$.

Given (F) and (F1), by (G1) $\hat{i} > 1$.

Given (H), (T), (E1), (W), and (Z), by Lemma 4.54 we have (H1) $\text{UpdatePtr}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}-1], \hat{bty}*) = (\hat{\sigma}_2, 1)$ such that (I1) $(\gamma, \sigma_2) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)$.

Given (I), (Q), (X), (A1), (C1), (G1), and (H1), by Vanilla C rule Pointer Dereference Write Higher Level Indirection we have $\Sigma \triangleright ((\text{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel \hat{C}) \Downarrow'_{\hat{\mathcal{D}}_1::(\text{p}, [\hat{wdp1}])} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)$.

Given (I1) and (V), by Definition 4.22 we have $((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_1) \cong_{\psi_1} ((\text{p}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}) \parallel \hat{C}_1)$. By Definition 4.23 we have $wdp2 \cong \hat{wdp1}$. Given (S), $\mathcal{D}_1 :: (\text{p}, [wdp2])$ and $\hat{\mathcal{D}}_1 :: (\text{p}, [\hat{wdp1}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{p}, [wdp2]) \cong \hat{\mathcal{D}}_1 :: (\text{p}, [\hat{wdp1}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p}, [wdp5])} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p}, [wdp2])} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0),(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p}, [wdp1])} ((\text{p}, \gamma, \sigma_2, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p}, [(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\text{p}, [wdp2])} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e]) \parallel \ldots \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, x[e])) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::(\text{ALL}, [mpra])} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, n^1) \parallel \ldots \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_1, \Delta^\text{q}_1, \text{acc}, n^\text{q}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e]) \parallel \ldots \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, x[e])) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2}_{\mathcal{D}_1::(\text{ALL}, [mpra])} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, n^1)$

$\| \ldots \|$ $(\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$ by SMC^2 rule Multiparty Array Read Private Index, we have (B) $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $\{(n^{\mathrm{p}}) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \;\|\; \ldots \;\|\; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, i^1) \;\|\; \ldots \;\|$ $(\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, i^{\mathrm{q}}))$, (E) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{const}\ a\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}},\ a\ \mathrm{const}\ bty*, 1,$ $\mathrm{PermL\_Ptr}(\mathrm{Freeable}, a\ \mathrm{const}\ bty*, a, 1))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $\{\mathrm{DecodePtr}(a\ \mathrm{const}\ bty*,\ 1,\ \omega^{\mathrm{p}}) = [1,\ [(l_1^{\mathrm{p}}, 0)],\ [1],\ 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $\{\sigma_1^{\mathrm{p}}(l_1^{\mathrm{p}}) = (\omega_1^{\mathrm{p}},\ a\ bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable}, a\ bty, a, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (I) $\{\forall i \in \{0 \ldots \alpha - 1\} \mathrm{DecodeArr}(a\ bty, i, \omega_1^{\mathrm{p}})$ $= n_i^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\mathrm{MPC}_{ar}((i^1, [n_0^1, \ldots, n_{\alpha-1}^1]), \ldots, (i^{\mathrm{q}}, [n_0^{\mathrm{q}}, \ldots, n_{\alpha-1}^{\mathrm{q}}])) = (n^1, \ldots, n^{\mathrm{q}})$, and $\mathcal{L}_2 = (1, [(l^1, 0), (l_1^1, 0), \ldots,$ $(l_1^1, \alpha - 1)]) \;\|\; \ldots \;\|\; (\mathrm{q}, [(l^{\mathrm{q}}, 0), (l_1^{\mathrm{q}}, 0), \ldots, (l_1^{\mathrm{q}}, \alpha - 1)])$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square, \square,\ \hat{x}[\hat{e}]) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma}^{\mathrm{q}},\ \hat{\sigma}^{\mathrm{q}},\ \square, \square,\ \hat{x}[\hat{e}]))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, x[e]) \cong_\psi$ $(\mathrm{p}, \hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}},\ \square, \square,\ \hat{x}[\hat{e}])\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_\psi (\hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (K) $x[e] \cong_\psi \hat{x}[\hat{e}]$. By Definition 4.20 we have $x \cong_\psi \hat{x}$ such that (L) $x = \hat{x}$ and (M) $e \cong_\psi \hat{e}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, x[e]) \sim (\mathrm{p},\ \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}},\ \mathrm{acc}, x[e])\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, x[e]) \cong_\psi (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{x}[\hat{e}])\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (N) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,$ $\hat{x}[\hat{e}]) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{x}[\hat{e}]))$. By Definition 4.22 we have (O) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_\psi (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (D), (M), (O), and ψ, by Lemma 4.28 we have (P) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}))$ such that (Q) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e) \cong_\psi (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (P) and (Q), by the inductive hypothesis, we have (R) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e})) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{i}) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{i}))$ and ψ_1 such that (S) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \mathrm{acc}, i^{\mathrm{p}}) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{i})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (T) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (S), by Definition 4.22 we have (U) $\{(\gamma^{\mathrm{p}},\ \sigma_1^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (V) $\{i^{\mathrm{p}} \cong_{\psi_1} \hat{i}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (E), (U), and (L), by Lemma 4.29 we have (W) $\hat{\gamma}(\hat{x}) = (\hat{l}, \mathrm{const}\ \hat{bty}*)$ such that (X) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (Y) a $\mathrm{const}\ bty* \cong_\psi \mathrm{const}\ \hat{bty}*$. By Definition 4.8 we have $bty = \hat{bty}$ and therefore (Z) $abty \cong_\psi \hat{bty}$.

Given (F), (U), and (X), by Lemma 4.30 we have (A1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \mathrm{const}\ \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{const}\ \hat{bty}*,$ $\mathrm{public}, 1))$ such that (B1) $\{\omega^{\mathrm{p}} \cong_\psi 1\hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (Y), and (B1), by Lemma 4.49 we have (C1) $\mathrm{DecodePtr}(\mathrm{const}\ \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (D1) $\{l_1^{\mathrm{p}} = \hat{l}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (H), (U), and (D1), by Lemma 4.30 we have (E1) $\hat{\sigma}_1(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha}))$ such that (F1) $\{\omega_1^{\mathrm{p}} \cong_\psi 1\hat{\omega}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (G1) $\alpha = \hat{\alpha}$.

Given (V), and (G1), by Axiom 4.15 we have (H1) $0 \leq \hat{i} \leq \hat{\alpha} - 1$.

Given (I), (Z), (V), and (F1), by Lemma 4.46 we have (I1) $\mathrm{DecodeArr}(\hat{bty},\ \hat{i},\ \hat{\omega}_1) = \hat{n}_{\hat{i}}$ such that (J1) $\{n_i^{\mathrm{p}} \cong_{\psi_1}$ $\hat{n}_{\hat{i}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (J), (J1), (H1), (G1), and (V), by Axiom 4.7 we have (K1) $\{n^{\mathrm{p}} \cong_{\psi_1} \hat{n}_{\hat{i}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (N), (R), (W), (A1), (C1), (E1), (H1), and (I1), by Vanilla C rule Multiparty Array Read we have $\Sigma \triangleright$ $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}]) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}])) \Downarrow'_{\hat{\mathcal{D}}_1::(\mathrm{ALL}, [\hat{mpra}])} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}))$.

Given (U) and (K1), by Definition 4.22 we have $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1) \;\|\; \ldots \;\|\; (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}})) \cong_{\psi_1}$ $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}) \;\|\; \ldots \;\|\; (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_{\hat{i}}))$.
By Definition 4.23 we have $mpra \cong \hat{mpra}$.

Given (T), $\mathcal{D}_1$ :: (ALL, $[mpra]$) and $\hat{\mathcal{D}}_1$ :: (ALL, $[\hat{mpra}]$), by Lemma 4.10 we have $\mathcal{D}_1$ :: (ALL, $[mpra]$) $\cong \hat{\mathcal{D}}_1$ :: (ALL, $[\hat{mpra}]$).
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e_1] = e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, x[e_1] = e_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::(\text{ALL}, [mpwa])} ((1, \gamma^1, \sigma^1_{3+\alpha-1}, \Delta^1_2, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_{3+\alpha-1}, \Delta^\text{q}_2, \text{acc}, \text{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e_1] = e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, x[e_1] = e_2)) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::(\text{ALL}, [mpwa])} ((1, \gamma^1, \sigma^1_{3+\alpha-1}, \Delta^1_2, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_{3+\alpha-1}, \Delta^\text{q}_2, \text{acc}, \text{skip}))$ by SMC^2 rule Multiparty Array Write Private Index, we have (B) $\{(e_1) \vdash \gamma^\text{p}\}^\text{q}_{\text{p}=1}$, (C) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e_1)) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, i^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_1, \Delta^\text{q}_1, \text{acc}, i^\text{q}))$, (D) $((1, \gamma^1, \sigma^1_1, \Delta^1_1, \text{acc}, e_2) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_1, \Delta^\text{q}_1 \text{acc}, e_2)) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((1, \gamma^1, \sigma^1_2, \Delta^1_2, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}_2, \Delta^\text{q}_2, \text{acc}, n^\text{q}))$, (E) $\{\gamma^\text{p}(x) = (l^\text{p}, \text{private const } bty*)\}^\text{q}_{\text{p}=1}$, (F) $\{\sigma^\text{p}_2(l^\text{p}) = (\omega^\text{p}, \text{ private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1))\}^\text{q}_{\text{p}=1}$, (G) $\{\text{DecodePtr}(\text{private const } bty*,\ 1,\ \omega^\text{p}) = [1,\ [(l^\text{p}_1,\ 0)],\ [1],\ 1]\}^\text{q}_{\text{p}=1}$, (H) $\{\sigma^\text{p}_2(l^\text{p}_1) = (\omega^\text{p}_1, \text{private } bty, \alpha, \text{ PermL}(\text{Freeable, private } bty, \text{private}, \alpha))\}^\text{q}_{\text{p}=1}$, (I) $\{\forall j \in \{0...\alpha-1\}\ \text{DecodeArr}(\text{private } bty, j, \omega^\text{p}_1) = n^\text{p}_j\}^\text{q}_{\text{p}=1}$, (J) $\text{MPC}_{aw}((i^1, n^1, [n^1_0, ..., n^1_{\alpha-1}]), ..., (i^\text{q}, n^\text{q}, [n^\text{q}_0, ..., n^\text{q}_{\alpha-1}])) = ([n'^1_0, ..., n'^1_{\alpha-1}], ..., [n'^\text{q}_0, ..., n'^\text{q}_{\alpha-1}])$, (K) $\{\forall j \in \{0...\alpha-1\}\ \text{UpdateArr}(\sigma^\text{p}_{2+j}, (l^\text{p}_1, j), n'^\text{p}_j, \text{ private } bty) = \sigma^\text{p}_{3+j}\}^\text{q}_{\text{p}=1}$, and $\mathcal{L}_3 = (1, [(l^\text{p}, 0), (l^\text{p}_1, 0), ..., (l^\text{p}_1, \alpha-1)]) \parallel ... \parallel (\text{q}, [(l^\text{p}, 0), (l^\text{p}_1, 0), ..., (l^\text{p}_1, \alpha-1)])$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}^\text{q},\ \hat{\sigma}^\text{q},\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2))$ and ψ such that $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, x[e_1] = e_2) \cong_\psi (\text{p}, \hat{\gamma}^\text{p},\ \hat{\sigma}^\text{p},\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2)\}^\text{q}_{\text{p}=1}$, by Definition 4.22 we have $\{(\gamma^\text{p},\ \sigma^\text{p}) \cong_\psi (\hat{\gamma}^\text{p},\ \hat{\sigma}^\text{p})\}^\text{q}_{\text{p}=1}$ and (L) $x[e_1] = e_2 \cong_\psi \hat{x}[\hat{e}_1] = \hat{e}_2$. By Definition 4.20 we have $x \cong_\psi \hat{x}$ such that (M) $x = \hat{x}$, (N) $e_1 \cong_\psi \hat{e}_1$, and (O) $e_2 \cong_\psi \hat{e}_2$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e_1] = e_2) \sim (\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, x[e_1] = e_2)\}^\text{q}_{\text{p}=1}$. By Lemma 4.86, we have $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, x[e_1] = e_2) \cong_\psi (\text{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2)\}^\text{q}_{\text{p}=1}$. and therefore (P) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{x}[\hat{e}_1] = \hat{e}_2))$. By Definition 4.22 we have (Q) $\{(\gamma^\text{p},\ \sigma^\text{p}) \cong_\psi (\hat{\gamma},\ \hat{\sigma})\}^\text{q}_{\text{p}=1}$.

Given (C), (Q), (N), and ψ, by Lemma 4.28 we have (R) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1) \parallel ... \parallel (\text{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1))$ such that (S) $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, e_1) \cong_\psi (\text{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_1)\}^\text{q}_{\text{p}=1}$. Given (R) and (S), by the inductive hypothesis, we have (T) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}_1)) \Downarrow'_{\hat{\mathcal{D}}_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{i}))$ and ψ_1 such that (U) $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}_1, \Delta^\text{p}_1, \text{acc}, i^\text{p}) \cong_{\psi_1} (\text{p}, \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ \hat{i})\}^\text{q}_{\text{p}=1}$ and (V) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (U), by Definition 4.22 we have (W) $\{(\gamma^\text{p},\ \sigma^\text{p}_1) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}^\text{q}_{\text{p}=1}$ and (X) $\{i^\text{p} \cong_{\psi_1} \hat{i}\}^\text{q}_{\text{p}=1}$.

Given Axiom 4.15, we have $(l, \mu) \notin e_2$. Given (O), by Lemma 4.7 we have (Y) $e_2 \cong_{\psi_1} \hat{e}_2$.

Given (D), (W), (Y), and ψ_1, by Lemma 4.28 we have (Z) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_2))$ such that (A1) $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, e_2) \cong_{\psi_1} (\text{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}_2)\}^\text{q}_{\text{p}=1}$. Given (Z) and (A1), by the inductive hypothesis, we have (B1) $((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_2} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}) \parallel ... \parallel (\text{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \hat{n}))$ and

ψ_2 such that (C1) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, \Delta_2^{\mathrm{p}}, \mathrm{acc}, n^{\mathrm{p}}) \cong_{\psi_2} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_2,\ \square, \square,\ \hat{n})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (D1) $\mathcal{D}_2 \cong \hat{\mathcal{D}}_2$. Given (C1), by Definition 4.22 we have (E1) $\{(\gamma^{\mathrm{p}},\ \sigma_2^{\mathrm{p}}) \cong_{\psi_2} (\hat{\gamma},\ \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (F1) $\{n^{\mathrm{p}} \cong_{\psi_2} \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (E), (E1), and (M), by Lemma 4.29 we have (G1) $\hat{\gamma}(\hat{x}) = (\hat{l}, \text{const } \hat{bty}*)$ such that (H1) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (I1) private const $bty* \cong_{\psi_2}$ const $\hat{bty}*$. By Definition 4.8 we have (J1) private $bty \cong_{\psi_2} \hat{bty}$.

Given (F), (E1), and (H1), by Lemma 4.30 we have (K1) $\hat{\sigma}_2(\hat{l}) = (\hat{\omega}, \text{const } \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \text{const } \hat{bty}*, \mathrm{public}, 1))$ such that (L1) $\{\omega^{\mathrm{p}} \cong_{\psi_2} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (I1), and (L1), by Lemma 4.49 we have (M1) $\mathrm{DecodePtr}(\text{const } \hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], 1]$ such that (N1) $\{l_1^{\mathrm{p}} = \hat{l}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (H), (E1), and (N1), by Lemma 4.30 we have (O1) $\hat{\sigma}_2(\hat{l}_1) = (\hat{\omega}_1, \hat{bty}, \hat{\alpha}, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, \hat{\alpha}))$ such that (P1) $\{\omega_1^{\mathrm{p}} \cong_{\psi_2} \hat{\omega}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (Q1) $\alpha = \hat{\alpha}$.

Given (C) and (H), by Axiom 4.15, we have (R1) $\{0 \leq i^{\mathrm{p}} \leq \alpha - 1\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (R1), (X), and (Q1), we have (S1) $0 \leq \hat{i} \leq \hat{\alpha} - 1$.

Given (J), (S1), (Q1), (X), and (F1), by Axiom 4.8 we have (T1) $\{n_{\hat{i}}^{\prime\mathrm{p}} \cong_{\psi_2} \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (U1) $\{\forall j \neq \hat{i} \in \{0...\alpha-1\} n_j^{\mathrm{p}} = n_j^{\prime\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K), (E1), (N1), (Q1), (P1), (I), (T1), (U1), and (J1), by Lemma 4.32 we have (V1) $\mathrm{UpdateArr}(\hat{\sigma}_2,\ (\hat{l}_1, \hat{i}),\ \hat{n},\ \hat{bty}) = \hat{\sigma}_3$ such that (W1) $\{\sigma_{3+\alpha-1}^{\mathrm{p}} \cong_{\psi_2} \hat{\sigma}_3\}_{\mathrm{p}=1}^{\mathrm{q}}$

Given (P), (T), (B1), (G1), (K1), (M1), (O1), (S1), and (V1), by Vanilla C rule Multiparty Array Write we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2) \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{x}[\hat{e}_1] = \hat{e}_2)) \Downarrow'_{\hat{\mathcal{D}}_1::\hat{\mathcal{D}}_2::(\mathrm{ALL}, [\hat{mpwa}])} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$.

Given (W1), by Definition 4.22 we have $((1, \gamma^1, \sigma_{3+\alpha-1}^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_{3+\alpha-1}^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip})) \cong_{\psi_2} ((1, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}) \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_3, \square, \square, \mathrm{skip}))$.
By Definition 4.23 we have $mpwa \cong \hat{mpwa}$.
Given (V), (D1), $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpwa])$ and $\hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{ALL}, [\hat{mpwa}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{ALL}, [mpwa]) \cong \hat{\mathcal{D}}_1 :: \hat{\mathcal{D}}_2 :: (\mathrm{ALL}, [\hat{mpwa}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_2} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, ++\ x)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, ++\ x)) \Downarrow_{(\mathrm{ALL}, [mppin])}^{(1, [(l^1, 0)])\ \parallel\ ...\ \parallel\ (\mathrm{q}, [(l^{\mathrm{q}}, 0)])} ((1, \gamma^1, \sigma_1^1, \Delta^1, \mathrm{acc}, n_2^1)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n_2^{\mathrm{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, ++\ x)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, ++\ x)) \Downarrow_{(\mathrm{ALL}, [mppin])}^{(1, [(l^1, 0)])\ \parallel\ ...\ \parallel\ (\mathrm{q}, [(l^{\mathrm{q}}, 0)])} ((1, \gamma^1, \sigma_1^1, \Delta^1, \mathrm{acc}, n_2^1)\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n_2^{\mathrm{q}}))$ by SMC^2 rule Multiparty Pre-Increment Private Float Variable, we have (B) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \text{private float})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $\{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private float}, 1, \mathrm{PermL}(\mathrm{Freeable}, \text{private float}, \mathrm{private}, 1))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{(x) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\{\mathrm{DecodeVal}(\text{private float}, \omega^{\mathrm{p}}) = n_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\mathrm{MPC}_u(++, n_1^1, ..., n_1^{\mathrm{q}}) = (n_2^1, ..., n_2^{\mathrm{q}})$, and (G) $\{\mathrm{UpdateVal}(\sigma^{\mathrm{p}}, l^{\mathrm{p}}, n_2^{\mathrm{p}}, \text{private float}) = \sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square, \square,\ ++\ \hat{x})\ \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}^{\mathrm{q}},\ \hat{\sigma}^{\mathrm{q}},\ \square, \square,\ ++\ \hat{x}))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, ++\ x)$

$\cong_\psi$ (p, $\hat{\gamma}^{\mathrm{p}}$, $\hat{\sigma}^{\mathrm{p}}$, $\square, \square$, $++\ \hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_\psi (\hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (H) $++\ x \cong_\psi ++\ \hat{x}$. By Definition 4.20 we have $x \cong_\psi \hat{x}$ such that (I) $x = \hat{x}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, ++\ x) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, ++\ x)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, ++\ x) \cong_\psi (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ ++\ \hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (J) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ ++\ \hat{x})\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ ++\ \hat{x}))$. By Definition 4.22 we have (K) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_\psi (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B), (K), and (I), by Lemma 4.29 we have (L) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty})$ such that (M) $\{l^{\mathrm{p}} \cong_\psi \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (N) private float $\cong_\psi \hat{bty}$.

Given (C), (K), and (M), by Lemma 4.30 we have (O) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}, 1, \mathrm{PermL}(\mathrm{Freeable}, \hat{bty}, \mathrm{public}, 1))$ such that (P) $\{\omega^{\mathrm{p}} \cong_\psi \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (E), (N), and (P), by Lemma 4.45 we have (Q) $\mathrm{DecodeVal}(\hat{bty}, \hat{\omega}) = \hat{n}_1$ such that (R) $\{n_1^{\mathrm{p}} \cong_\psi \hat{n}_1\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F) and (R), by Axiom 4.9 we have (S) $\hat{n}_2 = \hat{n}_1 + 1$ such that (T) $\{n_2^{\mathrm{p}} \cong \hat{n}_2\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (K), (M), (T), and (N), by Lemma 4.51 we have (U) $\mathrm{UpdateVal}(\hat{\sigma}, \hat{l}, \hat{n}_2, \hat{bty}) = \hat{\sigma}_1$ such that (V) $\{(\gamma^{\mathrm{p}},\ \sigma_1^{\mathrm{p}}) \cong_\psi (\hat{\gamma},\ \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (J), (L), (O), (Q), (S), and (U), by Vanilla C rule Multiparty Pre-Increment Variable we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma},\ \square, \square,\ ++\ \hat{x})\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma},\ \square, \square, ++\ \hat{x})) \Downarrow'_{(\mathrm{ALL}, [\hat{mppin}])} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2)\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2))$.

Given (V) and (T), by Definition 4.22 we have $((1, \gamma^1, \sigma_1^1, \Delta^1, \mathrm{acc}, n_2^1)\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n_2^{\mathrm{q}})) \cong_\psi ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2)\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}_2))$.
By Definition 4.23 we have $mppin \cong \hat{mppin}$. by Definition 4.25 we have $(\mathrm{ALL}, [mppin]) \cong (\mathrm{ALL}, [\hat{mppin}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_\psi \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x)\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp])}^{(1, (l^1, 0)::L^1)\ \|\ \ldots\ \|\ (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1)\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x)\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp])}^{(1, (l^1, 0)::L^1)\ \|\ \ldots\ \|\ (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1)\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$ by SMC^2 rule Multiparty Private Pointer Dereference Single Level Indirection, we have (B) $\{(x) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private}\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}},\ \mathrm{private}\ bty*,\ \alpha,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\alpha > 1$, (F) $\{\mathrm{DecodePtr}(\mathrm{private}\ bty*,\ \alpha,\ \omega^{\mathrm{p}}) = [\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $\{\mathrm{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \mathrm{private}\ bty, \sigma^{\mathrm{p}}) = ([n_0^{\mathrm{p}}, \ldots n_{\alpha-1}^{\mathrm{p}}], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (H) $\mathrm{MPC}_{dv}([[n_0^1, \ldots, n_{\alpha-1}^1], \ldots, [n_0^{\mathrm{q}}, \ldots, n_{\alpha-1}^{\mathrm{q}}]], [J^1, \ldots, J^{\mathrm{q}}]) = (n^1, \ldots, n^{\mathrm{q}})$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square, \square,\ *\hat{x})\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}^{\mathrm{q}},\ \hat{\sigma}^{\mathrm{q}},\ \square, \square,\ *\hat{x}))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x) \cong_\psi (\mathrm{p}, \hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}},\ \square, \square,\ *\hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_\psi (\hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (I) $*x \cong_\psi *\hat{x}$. By Definition 4.20 we have $x \cong_\psi \hat{x}$ such that (J) $x = \hat{x}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x) \cong_{\psi} (\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (K) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}))$. By Definition 4.22 we have (L) $\{(\gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C), (L), and (J), by Lemma 4.29 we have (M) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (N) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (O) private $bty* \cong_{\psi} \hat{bty}*$.

Given (D), (L), and (N), by Lemma 4.30 we have (P) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1))$ such that (Q) $\{\omega^{\mathrm{p}} \cong_{\psi} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F), (O), and (Q), by Lemma 4.48 we have (R) $\mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$ such that (S) $\{[\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, 1] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (O), by Definition 4.8 we have (T) private $bty \cong_{\psi} \hat{bty}$.

Given (G), (H), (S), (T), and (L), by Lemma 4.33 we have (U) $\mathrm{DerefPtr}(\hat{\sigma}, \hat{bty}, (\hat{l}_1, \hat{\mu}_1)) = (\hat{n}, 1)$ such that (V) $\{n^{\mathrm{p}} \cong \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (K), (M), (P), (R), and (U), by Vanilla C rule Multiparty Pointer Dereference we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x})) \Downarrow'_{(\mathrm{ALL}, [\hat{mprdp}])} ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}))$.

Given (L) and (V), by Definition 4.22 we have $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}})) \cong_{\psi} ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{n}))$.
By Definition 4.23 we have $mprdp \cong \hat{mprdp}$. by Definition 4.25 we have $(\mathrm{ALL}, [mprdp]) \cong (\mathrm{ALL}, [\hat{mprdp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp1])}^{(1, (l^1, 0)::L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, [\alpha_\alpha, L_\alpha^1, J_\alpha^1, i-1]) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, [\alpha_\alpha, L_\alpha^{\mathrm{q}}, J_\alpha^{\mathrm{q}}, i-1]))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp1])}^{(1, (l^1, 0)::L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, [\alpha_\alpha, L_\alpha^1, J_\alpha^1, i-1]) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, [\alpha_\alpha, L_\alpha^{\mathrm{q}}, J_\alpha^{\mathrm{q}}, i-1]))$ by SMC^2 rule Multiparty Private Pointer Dereference Higher Level Indirection, we have (B) $\{(x) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private}\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \mathrm{private}\ bty*, \alpha, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\alpha > 1$,
(F) $\{\mathrm{DecodePtr}(\mathrm{private}\ bty*, \alpha, \omega^{\mathrm{p}}) = [\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $i > 1$, (H) $\{\mathrm{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \mathrm{private}\ bty*, \sigma^{\mathrm{p}}) = ([[\alpha_0, L_0^{\mathrm{p}}, J_0^{\mathrm{p}}, i-1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{p}}, J_{\alpha-1}^{\mathrm{p}}, i-1]], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (I) $\mathrm{MPC}_{dp}([[[\alpha_0, L_0^1, J_0^1], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^1, J_{\alpha-1}^1]], \ldots, [[\alpha_0, L_0^{\mathrm{q}}, J_0^{\mathrm{q}}], \ldots, [\alpha_{\alpha-1}, L_{\alpha-1}^{\mathrm{q}}, J_{\alpha-1}^{\mathrm{q}}]]], [J^1, \ldots, J^{\mathrm{q}}]) = ([[\alpha_\alpha, L_\alpha^1, J_\alpha^1], \ldots, [\alpha_\alpha, L_\alpha^{\mathrm{q}}, J_\alpha^{\mathrm{q}}]])$.

Given (A), $((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, *\hat{x}) \parallel \ldots \parallel (\mathrm{q}, \hat{\gamma}^{\mathrm{q}}, \hat{\sigma}^{\mathrm{q}}, \square, \square, *\hat{x}))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x) \cong_{\psi} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}}, \hat{\sigma}^{\mathrm{p}}, \square, \square, *\hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}^{\mathrm{p}}, \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (J) $*x \cong_{\psi} *\hat{x}$. By Definition 4.20 we have $x \cong_{\psi} \hat{x}$ such that (K) $x = \hat{x}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x) \cong_{\psi} (\mathrm{p}, \hat{\gamma}, \ \hat{\sigma}, \ \square, \square, \ *\hat{x})\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (L) $((1, \hat{\gamma}, \ \hat{\sigma}, \ \square, \square, \ *\hat{x}) \ \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \ \hat{\sigma}, \ \square, \square, \ *\hat{x}))$. By Definition 4.22 we have (M) $\{(\gamma^{\mathrm{p}}, \ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C), (M), and (K), by Lemma 4.29 we have (N) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty*})$ such that (O) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (P) private $bty* \cong_{\psi} \hat{bty*}$.

Given (D), (M), and (O), by Lemma 4.30 we have (Q) $\hat{\sigma}(\hat{l}) = (\hat{\omega}, \hat{bty*}, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty*}, \mathrm{public}, 1))$ such that (R) $\{\omega^{\mathrm{p}} \cong_{\psi} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F), (P), and (R), by Lemma 4.48 we have (S) $\mathrm{DecodePtr}(\hat{bty*}, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (T) $\{[\alpha, \ L^{\mathrm{p}}, \ J^{\mathrm{p}}, i] \cong_{\psi} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (T), by Definition 4.15 we have (U) $i = \hat{i}$.

Given (G) and (U), we have (V) $\hat{i} > 1$.

Given (H), (I), (T), (P), and (M), by Lemma 4.34 we have (W) $\mathrm{DerefPtrHLI}(\hat{\sigma}, \hat{bty*}, (\hat{l}_1, \hat{\mu}_1)) = ([1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1], 1)$ such that (X) $\{[\alpha_\alpha, \ L_\alpha^{\mathrm{q}}, \ J_\alpha^{\mathrm{q}}, \hat{i}-1] \cong_{\psi} [1, [(\hat{l}_2, \hat{\mu}_2)], [1], \hat{i}-1]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (X), by Lemma 4.27 we have (Y) $\{[\alpha_\alpha, \ L_\alpha^{\mathrm{q}}, \ J_\alpha^{\mathrm{q}}, \hat{i}-1] \cong_{\psi} (\hat{l}_2, \hat{\mu}_2)$.

Given (L), (N), (Q), (S), (V), and (W), by Vanilla C rule Multiparty Pointer Dereference Higher Level Indirection we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x}) \parallel ... \parallel \ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x})) \Downarrow'_{(\mathrm{ALL}, [\hat{mprdp1}])} ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)))$.

Given (M) and (Y), by Definition 4.22 we have $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, [\alpha_\alpha, L_\alpha^1, J_\alpha^1, i-1]) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, [\alpha_\alpha, L_\alpha^{\mathrm{q}}, J_\alpha^{\mathrm{q}}, i-1])) \cong_{\psi} ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)) \parallel ... \parallel \ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, (\hat{l}_2, \hat{\mu}_2)))$.
By Definition 4.23 we have $mprdp1 \cong \hat{mprdp1}$. by Definition 4.25 we have $(\mathrm{ALL}, [mprdp1]) \cong (\mathrm{ALL}, [\hat{mprdp1}])$. Therefore, by Definition 4.26 we have $\Pi \cong_{\psi} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \ \sigma^1, \ \Delta^1, \ \mathrm{acc}, \ *x = e) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \ \sigma^{\mathrm{q}}, \ \Delta^{\mathrm{q}}, \ \mathrm{acc}, \ *x = e))$
$\Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp3])}^{\mathcal{L}_1::(1, (l^1, 0)::L_1^1::L^1) \ \parallel \ ... \ \parallel \ (\mathrm{q}, (l^{\mathrm{q}}, 0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \ \sigma_2^1, \ \Delta_2^1, \ \mathrm{acc}, \ \mathrm{skip}) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \ \sigma_2^{\mathrm{q}}, \ \Delta_2^{\mathrm{q}}, \ \mathrm{acc}, \ \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \ \sigma^1, \ \Delta^1, \ \mathrm{acc}, \ *x = e) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \ \sigma^{\mathrm{q}}, \ \Delta^{\mathrm{q}}, \ \mathrm{acc}, \ *x = e))$
$\Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp3])}^{\mathcal{L}_1::(1, (l^1, 0)::L_1^1::L^1) \ \parallel \ ... \ \parallel \ (\mathrm{q}, (l^{\mathrm{q}}, 0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \ \sigma_2^1, \ \Delta_2^1, \ \mathrm{acc}, \ \mathrm{skip}) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \ \sigma_2^{\mathrm{q}}, \ \Delta_2^{\mathrm{q}}, \ \mathrm{acc}, \ \mathrm{skip}))$ by SMC^2 rule Multiparty Private Pointer Dereference Write Private Value, we have (B) $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $((1, \gamma^1, \ \sigma^1, \ \Delta^1, \ \mathrm{acc}, \ e) \ \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1) \parallel ... \parallel \ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$, (D) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private} \ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \mathrm{private} \ bty*, \ \alpha, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private} \ bty*, \mathrm{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\alpha > 1$, (G) $\{\mathrm{DecodePtr}(\mathrm{private} \ bty*, \ \alpha, \ \omega^{\mathrm{p}}) = [\alpha, \ L^{\mathrm{p}}, \ J^{\mathrm{p}}, 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $\{\mathrm{DynamicUpdate}(\Delta_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, L^{\mathrm{p}}, \ \mathrm{acc}, \mathrm{private} \ bty) = (\Delta_2^{\mathrm{p}}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (I) $\{\mathrm{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \mathrm{private} \ bty, \sigma_1^{\mathrm{p}}) = ([n_0^{\mathrm{p}}, ...n_{\alpha-1}^{\mathrm{p}}], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\mathrm{MPC}_{wdv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^{\mathrm{q}}, ..., n_{\alpha-1}^{\mathrm{q}}]], [n^1, ..., n^{\mathrm{q}}], [J^1, ..., J^{\mathrm{q}}]) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^{\mathrm{q}}, ..., n_{\alpha-1}'^{\mathrm{q}}])$, and
(K) $\{\mathrm{UpdateDerefVals}(\alpha, L^{\mathrm{p}}, [n_0'^{\mathrm{p}}, ..., n_{\alpha-1}'^{\mathrm{p}}], \mathrm{private} \ bty, \sigma_1^{\mathrm{p}}) = \sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (A), $((1, \hat{\gamma}^1, \ \hat{\sigma}^1, \ \square, \square, \ *\hat{x} = \hat{e}) \ \parallel ... \parallel \ (\mathrm{q}, \hat{\gamma}^{\mathrm{q}}, \ \hat{\sigma}^{\mathrm{q}}, \ \square, \square, \ *\hat{x} = \hat{e}))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x = e) \cong_{\psi} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}}, \ \hat{\sigma}^{\mathrm{p}}, \ \square, \square, \ *\hat{x} = \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}}, \ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}^{\mathrm{p}}, \ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (L) $*x = e \cong_{\psi} *\hat{x} = \hat{e}$. By Definition 4.20 we have $x \cong_{\psi} \hat{x}$ such that (M) $x = \hat{x}$ and (N) $e \cong_{\psi} \hat{e}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \sim (\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x = e)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, *x = e) \cong_{\psi} (\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (O) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}))$. By Definition 4.22 we have (P) $\{(\gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}, \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C), (P), (N), and ψ, by Lemma 4.28 we have (Q) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}))$ such that (R) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, e) \cong_{\psi} (\mathrm{p}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (Q) and (R), by the inductive hypothesis, we have (S) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n}))$ and ψ_1 such that (T) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \mathrm{acc}, n^{\mathrm{p}}) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \hat{n})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (U) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (T), by Definition 4.22 we have (V) $\{(\gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (W) $\{n^{\mathrm{p}} \cong_{\psi_1} \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (D), (V), and (M), by Lemma 4.29 we have (X) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (Y) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (Z) private $bty* \cong_{\psi_1} \hat{bty}*$. By Definition 4.8 we have (A1) private $bty \cong_{\psi_1} \hat{bty}$.

Given (E), (V), and (Y), by Lemma 4.30 we have (B1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty}*, \mathrm{public}, 1))$ such that (C1) $\{\omega^{\mathrm{p}} \cong_{\psi_1} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (Z), and (C1), by Lemma 4.48 we have (D1) $\mathrm{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]$ such that (E1) $\{[\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, 1] \cong_{\psi_1} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (I), (J), (K), (E1), (W), (A1), and (V), by Lemma 4.35 we have (F1) $\mathrm{UpdateOffset}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), \hat{n}, \hat{bty}) = (\hat{\sigma}_2, 1)$ such that (G1) $\{(\gamma^{\mathrm{p}}, \sigma_2^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma}, \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (O), (S), (X), (B1), (D1), and (F1), by Vanilla C rule Multiparty Pointer Dereference Write Value we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})) \Downarrow'_{\hat{\mathcal{D}}::(\mathrm{ALL}, [\hat{mpwdp}])} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$.

Given (G1), by Definition 4.22 we have $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip})) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \mathrm{skip}))$.
By Definition 4.23 we have $mpwdp3 \cong \hat{mpwdp}$. Given (U), $\mathcal{D}_1 :: (\mathrm{ALL}, [mpwdp3])$ and $\hat{\mathcal{D}}_1 :: (\mathrm{ALL}, [\hat{mpwdp}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\mathrm{ALL}, [mpwdp3]) \cong \hat{\mathcal{D}}_1 :: (\mathrm{ALL}, [\hat{mpwdp}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp3])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$. Given $\{n^{\mathrm{p}} = \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$, we use Definition 4.19 to prove that $\{\mathrm{encrypt}(n^{\mathrm{p}}) \cong \hat{n}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp2])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp2])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Multiparty Private Pointer Dereference Write Value Higher Level Indirection, we have (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, (l_e^1, \mu_e^1)) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, (l_e^{\mathrm{q}}, \mu_e^{\mathrm{q}})))$,

(C) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \text{private } bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}},\ \text{private } bty*,\ \alpha,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\alpha > 1$, (F) $\{\text{DecodePtr}(\text{private } bty*,\ \alpha,\ \omega^{\mathrm{p}}) = [\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $i > 1$, (H) $\{\text{DynamicUpdate}(\Delta_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, L^{\mathrm{p}},\ \text{acc}, \text{private } bty*) = (\Delta_2^{\mathrm{p}}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (I) $\{\text{Retrieve\_vals}(\alpha,\ L^{\mathrm{p}},\ \text{private } bty*,\ \sigma_1^{\mathrm{p}}) = ([[\alpha_0,\ L_0^{\mathrm{p}},\ J_0^{\mathrm{p}}, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{p}}, J_{\alpha-1}^{\mathrm{p}}, i-1]], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\text{MPC}_{wdp}([[[1, [(l_e^1, \mu_e^1)], [1], i-1], [\alpha_0,\ L_0^1,\ J_0^1, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1, i-1]], ..., [[1, [(l_e^{\mathrm{q}}, \mu_e^{\mathrm{q}})], [1], i-1], [\alpha_0,\ L_0^{\mathrm{q}},\ J_0^{\mathrm{q}}, i-1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{q}},\ J_{\alpha-1}^{\mathrm{q}}, i-1]]], [J^1, ..., J^{\mathrm{q}}]) = [[[\alpha_0',\ L_0'^1,\ J_0'^1, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^1,\ J_{\alpha-1}'^1, i-1]], ..., [[\alpha_0',\ L_0'^{\mathrm{q}},\ J_0'^{\mathrm{q}}, i-1], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^{\mathrm{q}},\ J_{\alpha-1}'^{\mathrm{q}}, i-1]]]$, (K) $\{\text{UpdateDerefVals}(\alpha, L^{\mathrm{p}}, [[\alpha_0', L_0'^{\mathrm{p}}, J_0'^{\mathrm{p}}, i-1], ..., [\alpha_{\alpha-1}', L_{\alpha-1}'^{\mathrm{p}}, J_{\alpha-1}'^{\mathrm{p}}, i-1]], \text{private } bty*, \sigma_1^{\mathrm{p}}) = \sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (A), $((1, \hat{\gamma}^1,\ \hat{\sigma}^1,\ \square, \square,\ *\hat{x} = \hat{e})\ \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}^{\mathrm{q}},\ \hat{\sigma}^{\mathrm{q}},\ \square, \square,\ *\hat{x} = \hat{e}))$ and ψ such that $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \text{acc}, *x = e) \cong_{\psi} (\mathrm{p}, \hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}},\ \square, \square,\ *\hat{x} = \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$, by Definition 4.22 we have $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma}^{\mathrm{p}},\ \hat{\sigma}^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (L) $*x = e \cong_{\psi} *\hat{x} = \hat{e}$. By Definition 4.20 we have $x \cong_{\psi} \hat{x}$ such that (M) $x = \hat{x}$ and (N) $e \cong_{\psi} \hat{e}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \sim (\mathrm{p},\ \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}},\ \text{acc}, *x = e)\}_{\mathrm{p}=1}^{\mathrm{q}}$. By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \text{acc}, *x = e) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ *\hat{x} = \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (O) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square, *\hat{x} = \hat{e})\ \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ *\hat{x} = \hat{e}))$. By Definition 4.22 we have (P) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B), (P), (N), and ψ, by Lemma 4.28 we have (Q) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e})\parallel ... \parallel\ (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e}))$ such that (R) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \text{acc}, e) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \hat{e})\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (Q) and (R), by the inductive hypothesis, we have (S) $((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})\ \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \hat{e})) \Downarrow'_{\hat{\mathcal{D}}} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e))\ \parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, (\hat{l}_e, \hat{\mu}_e)))$ and ψ_1 such that (T) $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \Delta_1^{\mathrm{p}},\ \text{acc}, (l_e^{\mathrm{p}}, \mu_e^{\mathrm{p}})) \cong_{\psi_1} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma}_1,\ \square, \square,\ (\hat{l}_e, \hat{\mu}_e))\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (U) $\mathcal{D}_1 \cong \hat{\mathcal{D}}_1$. Given (T), by Definition 4.22 we have (V) $\{(\gamma^{\mathrm{p}},\ \sigma_1^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (W) $\{(l_e^{\mathrm{p}}, \mu_e^{\mathrm{p}}) \cong_{\psi_1} (\hat{l}_e, \hat{\mu}_e)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C), (V), and (M), by Lemma 4.29 we have (X) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty}*)$ such that (Y) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (Z) private $bty* \cong_{\psi_1} \hat{bty}*$.

Given (D), (V), and (Y), by Lemma 4.30 we have (A1) $\hat{\sigma}_1(\hat{l}) = (\hat{\omega}, \hat{bty}*, 1, \text{PermL\_Ptr}(\text{Freeable}, \hat{bty}*, \text{public}, 1))$ such that (B1) $\{\omega^{\mathrm{p}} \cong_{\psi_1} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F), (Z), and (B1), by Lemma 4.48 we have (C1) $\text{DecodePtr}(\hat{bty}*, 1, \hat{\omega}) = [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$ such that (D1) $[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}}, i] \cong_{\psi_1} [1, [(\hat{l}_1, \hat{\mu}_1)], [1], \hat{i}]$. Given (D1) by Definition 4.15 we have (E1) $i = \hat{i}$.

Given (G) and (E1), we have (F1) $\hat{i} > 1$.

Given (I), (J), (K), (D1), (W), (Z), and (V), by Lemma 4.36 we have (G1) $\text{UpdatePtr}(\hat{\sigma}_1, (\hat{l}_1, \hat{\mu}_1), [1, [(\hat{l}_e, \hat{\mu}_e)], [1], \hat{i}-1], \hat{bty}*) = (\hat{\sigma}_2, 1)$ such that (H1) $\{(\gamma^{\mathrm{p}},\ \sigma_2^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_2)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (O), (S), (X), (A1), (C1), (F1), and (G1), by Vanilla C rule Multiparty Pointer Dereference Write Value Higher Level Indirection we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})\parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, *\hat{x} = \hat{e})) \Downarrow'_{\hat{\mathcal{D}}::(\text{ALL}, [\hat{mpwdp1}])} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip})\parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}))$.

Given (H1), by Definition 4.22 we have $((1, \gamma^1,\ \sigma_2^1,\ \Delta_2^1,\ \text{acc},\ \text{skip})\ \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \text{acc}, \text{skip})) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip})\parallel ... \parallel\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_2, \square, \square, \text{skip}))$.
By Definition 4.23 we have $mpwdp2 \cong \hat{mpwdp1}$. Given (U), $\mathcal{D}_1 :: (\text{ALL}, [mpwdp2])$ and $\hat{\mathcal{D}}_1 :: (\text{ALL}, [\hat{mpwdp1}])$, by Lemma 4.10 we have $\mathcal{D}_1 :: (\text{ALL}, [mpwdp2]) \cong \hat{\mathcal{D}}_1 :: (\text{ALL}, [\hat{mpwdp1}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpwdp1])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e))$ $\Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpwdp2])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$. The main difference between the two is that *mpwdp1* uses reasoning about evaluating an expression to multiple locations, similar to that in Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, *x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p}, [wdp2])}^{\mathcal{L}_1::(\text{p}, [(l,0)]::L_1::[(l_1,\mu_1)])} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, \text{pfree}(x)))$ $\Downarrow_{(\text{ALL}, [mpfre])}^{(1,[(l^1,0)]::L^1::L_1^1) \parallel ... \parallel (\text{q},[(l^\text{q},0)]::L^\text{q}::L_1^\text{q})} ((1, \gamma^1, \sigma_2^1, \Delta^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta^\text{q}, \text{acc}, \text{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, \text{pfree}(x)))$ $\Downarrow_{(\text{ALL}, [mpfre])}^{(1,[(l^1,0)]::L^1::L_1^1) \parallel ... \parallel (\text{q},[(l^\text{q},0)]::L^\text{q}::L_1^\text{q})} ((1, \gamma^1, \sigma_2^1, \Delta^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta^\text{q}, \text{acc}, \text{skip}))$ by SMC^2 rule Private Free Multiple Locations, we have (B) $\{\gamma^\text{p}(x) = (l^\text{p}, \text{private } bty*)\}_{\text{p}=1}^{\text{q}}$, (C) $\text{acc} = 0$, (D) $(bty = \text{int}) \vee (bty = \text{float})$, (E) $\{\sigma^\text{p}(l^\text{p}) = (\omega^\text{p}, \text{private } bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, \alpha))\}_{\text{p}=1}^{\text{q}}$, (F) $\{\alpha > 1\}_{\text{p}=1}^{\text{q}}$, (G) $\{[\alpha, L^\text{p}, J^\text{p}, i] = \text{DecodePtr}(\text{private } bty*, \alpha, \omega^\text{p})\}_{\text{p}=1}^{\text{q}}$, (H) $\text{if}(i > 1)\{ty = \text{private } bty*\} \text{ else } \{ty = \text{private } bty\}$, (I) $\{\text{CheckFreeable}(\gamma^\text{p}, L^\text{p}, J^\text{p}, \sigma^\text{p}) = 1\}_{\text{p}=1}^{\text{q}}$, (J) $\{\forall (l_m^\text{p}, 0) \in L^\text{p}.\ \sigma^\text{p}(l_m^\text{p}) = (\omega_m^\text{p}, ty, \alpha_m, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha_m))\}_{\text{p}=1}^{\text{q}}$, (K) $\text{MPC}_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^\text{q}, ..., \omega_{\alpha-1}^\text{q}]], [J^1, ... J^\text{q}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^\text{q}, ..., \omega_{\alpha-1}'^\text{q}]], [J'^1, ..., J'^\text{q}])$, (L) $\{\text{UpdateBytesFree}(\sigma^\text{p}, L^\text{p}, [\omega_0'^\text{p}, ..., \omega_{\alpha-1}'^\text{p}]) = \sigma_1^\text{p}\}_{\text{p}=1}^{\text{q}}$, and (M) $\{\sigma_2^\text{p} = \text{UpdatePointerLocations}(\sigma_1^\text{p}, L^\text{p}[1 : \alpha - 1], J^\text{p}[1 : \alpha - 1], L^\text{p}[0], J^\text{p}[0])\}_{\text{p}=1}^{\text{q}}$.

Given (A), $((1, \hat{\gamma}^1, \hat{\sigma}^1, \square, \square, \text{free}(\hat{x})) \parallel ... \parallel (\text{q}, \hat{\gamma}^\text{q}, \hat{\sigma}^\text{q}, \square, \square, \text{free}(\hat{x})))$ and ψ such that $\{(\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, \text{pfree}(x)) \cong_\psi (\text{p}, \hat{\gamma}^\text{p}, \hat{\sigma}^\text{p}, \square, \square, \text{free}(\hat{x}))\}_{\text{p}=1}^{\text{q}}$, by Definition 4.22 we have $\{(\gamma^\text{p}, \sigma^\text{p}) \cong_\psi (\hat{\gamma}^\text{p}, \hat{\sigma}^\text{p})\}_{\text{p}=1}^{\text{q}}$ and (N) $\text{pfree}(x) \cong_\psi \text{free}(\hat{x})$. By Definition 4.20 we have $x \cong_\psi \hat{x}$ such that (O) $x = \hat{x}$.

Given Axiom 4.15, by Theorem 4.1 we have $\{(1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \sim (\text{p}, \gamma^\text{p}, \sigma^\text{p}, \Delta^\text{p}, \text{acc}, \text{pfree}(x))\}_{\text{p}=1}^{\text{q}}$.

By Lemma 4.86, we have $\{(\mathrm{p}, \gamma^{\mathrm{p}}, \sigma^{\mathrm{p}}, \Delta^{\mathrm{p}}, \mathrm{acc}, \mathrm{pfree}(x)) \cong_{\psi} (\mathrm{p}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \mathrm{free}(\hat{x}))\}_{\mathrm{p}=1}^{\mathrm{q}}$. and therefore (P) $((1, \hat{\gamma},\ \hat{\sigma},\ \square, \square, \mathrm{free}(\hat{x}))\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma},\ \hat{\sigma},\ \square, \square,\ \mathrm{free}(\hat{x})))$. By Definition 4.22 we have (Q) $\{(\gamma^{\mathrm{p}},\ \sigma^{\mathrm{p}}) \cong_{\psi} (\hat{\gamma},\ \hat{\sigma})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B), (Q), and (O), by Lemma 4.29 we have (R) $\hat{\gamma}(\hat{x}) = (\hat{l}, \hat{bty*})$ such that (S) $\{l^{\mathrm{p}} = \hat{l}\}_{\mathrm{p}=1}^{\mathrm{q}}$ and (T) private $bty* \cong_{\psi} \hat{bty*}$.

Given (E), (Q), and (S), by Lemma 4.30 we have (U) $\sigma(\hat{l}) = (\hat{\omega}, \hat{bty*}, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \hat{bty*}, \mathrm{public}, 1))$ such that (V) $\{\omega^{\mathrm{p}} \cong_{\psi} \hat{\omega}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (T), and (V), by Lemma 4.48 we have (W) $\mathrm{DecodePtr}(\hat{bty*}, 1, \hat{\omega}) = [1, [(\hat{l}_1, 0)], [1], \hat{i}]$ such that (X) $\{[\alpha,\ L^{\mathrm{p}}\ J^{\mathrm{p}},\ i] \cong_{\psi} [1, [(\hat{l}_1, 0)], [1], \hat{i}]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (I), (Q), and (X), by Axiom 4.2 we have (Y) $\mathrm{CheckFreeable}(\hat{\gamma}, [(\hat{l}_1, 0)], [1], \hat{\sigma}) = 1$.

Given (J), (K), (L), (M), (X), and (Q), by Lemma 4.37 we have (Z) $\mathrm{Free}(\hat{\sigma}, \hat{l}_1) = \hat{\sigma}_1$ and ψ_1 such that (A1) $\{(\gamma^{\mathrm{p}},\ \sigma_2^{\mathrm{p}}) \cong_{\psi_1} (\hat{\gamma},\ \hat{\sigma}_1)\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (P), (R), (U), (W), (Y), and (Z), by Vanilla C rule Multiparty Free we have $\Sigma \triangleright ((1, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{free}(\hat{x}))\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}, \square, \square, \mathrm{free}(\hat{x}))) \Downarrow'_{(\mathrm{ALL}, [\hat{mpfre}])} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip})\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}))$.

Given (A1), by Definition 4.22 we have $((1, \gamma^1, \sigma_2^1, \Delta^1, \mathrm{acc}, \mathrm{skip})\ \|\ \ldots\ \|\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip})) \cong_{\psi_1} ((1, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip})\ \|\ \ldots\ \|\ (\mathrm{q}, \hat{\gamma}, \hat{\sigma}_1, \square, \square, \mathrm{skip}))$.
By Definition 4.23 we have $mpfre \cong \hat{mpfre}$. by Definition 4.25 we have $(\mathrm{ALL}, [mpfre]) \cong (\mathrm{ALL}, [\hat{mpfre}])$.
Therefore, by Definition 4.26 we have $\Pi \cong_{\psi_1} \Sigma$.

$\square$

5 NONINTERFERENCE

5.1 Noninterference: Definitions

Definition 5.1 (ϕ). We define the function ϕ to return a single unused memory block identifier in a monotonically increasing fashion.

Axiom 5.1 (encrypt). *Given the use of an encryption scheme that ensures encrypted numbers are indistinguishable, we assume that given any two numbers n_1, n_2, their respective encrypted values* $\text{encrypt}(n_1), \text{encrypt}(n_2)$ *can be viewed as equivalent.*

Axiom 5.2 (InputValue). *Given two input files $input1, input2$ and variable x corresponding to a program of statement s, if and only if $input1 = input2$ by Definition 5.7 then* $\text{InputValue}(x, input1) = n$ *and* $\text{InputValue}(x, input2) = n'$ *such that $n = n'$.*

Axiom 5.3 (InputArray). *Given two input files $input1, input2$ and array x of length m corresponding to a program of statement s, if and only if $input1 = input2$ by Definition 5.7 then* $\text{InputArray}(x, input1, m) = [n_0, ..., n_{m-1}]$ *and* $\text{InputArray}(x, input2, m) = [n'_0, ..., n'_{m-1}]$ *such that for every index i in $0...m$, $n_i = n'_i$.*

Axiom 5.4 (ϕ). *Given a program of statement s, during any two executions Π, Σ over s such that $\Pi \simeq_L \Sigma$ by Definition 5.2, if ϕ returns memory block identifier l at step d in Π, then by definition 5.1 ϕ will also return l at step d in Σ.*

Definition 5.2 ($\Pi \simeq_L \Sigma$). Two SMC^2 evaluation trees Π and Σ are *low-equivalent*, in symbols $\Pi \simeq_L \Sigma$, if and only if Π and Σ have the same structure as trees, and for each node in
Π proving $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}^1, s) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}^{\text{q}}, s))$
$\Downarrow_{\mathcal{D}}^{\mathcal{L}} ((1, \gamma_1^1, \sigma_1^1, \Delta_1^1, \text{acc}_1^1, v^1) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}_1^{\text{q}}, v^{\text{q}}))$, the corresponding node in
Σ proves $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}^1, s) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}^{\text{q}}, s))$
$\Downarrow_{\mathcal{D}'}^{\mathcal{L}'} ((1, \gamma_1^1, \sigma_1^1, \Delta_1^1, \text{acc}_1^1, v'^1) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}_1^{\text{q}}, v^{\text{q}}))$ and both $\mathcal{D} = \mathcal{D}'$ and $\mathcal{L} = \mathcal{L}'$.

Definition 5.3 ($\gamma = \gamma'$). Two environments are equivalent, in symbols $\gamma = \gamma'$, if and only if $(x \to (l, ty)) \in \gamma \iff (x \to (l, ty)) \in \gamma'$.

Definition 5.4 ($\sigma = \sigma'$). Two memories are equivalent, in symbols $\sigma = \sigma'$, if and only if $(l \to (\omega, ty, \alpha, \text{PermL}(p, ty, a, \alpha))) \in \sigma \iff (l \to (\omega, ty, \alpha, \text{PermL}(p, ty, a, \alpha))) \in \sigma'$.

Definition 5.5 ($\Delta = \Delta'$). Two location maps are equivalent, in symbols $\Delta = \Delta'$, if and only if $\delta \in \Delta \iff \delta' \in \Delta'$ such that $\delta = \delta'$.

Definition 5.6 ($\delta = \delta'$). Two nested location maps are equivalent, in symbols $\delta = \delta'$, if and only if $((l, \mu) \to (v_1, v_2, j, ty)) \in \delta \iff ((l, \mu) \to (v_1, v_2, j, ty)) \in \delta'$.

Definition 5.7 (Input Equality). Given input files $input1, input2$, $input1 = input2$ if and only if

- for every public variable x, if $\{x = n\} \in input1$ then $\{x = n\} \in input2$,
- for every public array x, if $\{x = n_0, ..., n_m\} \in input1$ then $\{x = n_0, ..., n_m\} \in input2$,
- for every private variable x, if $\{x = n\} \in input1$ then $\{x = n'\} \in input2$ such that $n = n'$ by Axiom 5.1, and
- for every private array x, if $\{x = n_0, ..., n_m\} \in input1$ then $\{x = n'_0, ..., n'_m\} \in input2$ such that for every index i in $0...m$, $n_i = n'_i$ by Axiom 5.1.

5.2 Noninterference: Lemmas

Lemma 5.1 (OutputValue). *Given variable x, x', values n, n_1, n', n'_1 such that* $\text{OutputValue}(x, n, n_1)$ *and* $\text{OutputValue}(x', n', n'_1)$, *if $x = x'$, $n = n'$, and $n_1 = n'_1$, then* OutputValue *will give identical output to the same parties.*

Proof. By definition of Algorithm OutputValue, the content of the output and the parties it is given to by OutputValue is deterministic based on the given input. □

Lemma 5.2 (OutputArray). *Given variable x, x', values n, n', α, α', $[m_0, \ldots, m_{\alpha-1}]$, $[m'_0, \ldots, m'_{\alpha'-1}]$) such that* OutputArray(x, n, $[m_0, \ldots, m_{\alpha-1}]$) *and* OutputArray($x'$, n', $[m'_0, \ldots, m'_{\alpha'-1}]$)*, if $x = x'$, $n = n'$, $\alpha = \alpha'$, and $[m_0, \ldots, m_{\alpha-1}] = [m'_0, \ldots, m'_{\alpha'-1}]$), then* OutputArray *will give identical output to the same parties.*

Proof. By definition of Algorithm OutputArray, the content of the output and the parties it is given to by OutputArray is deterministic based on the given input. □

Lemma 5.3 (GetFunTypeList). *Given parameter list P, P', such that* GetFunTypeList(P) $= tyL$ *and* GetFunTypeList(P') $= tyL'$*, if $P = P'$ then $tyL = tyL'$.*

Proof. By definition of Algorithm GetFunTypeList, the type list returned by GetFunTypeList is deterministic based on the given input. □

Lemma 5.4 (GetFunParamAssign). *Given parameter list $P = P'$ and expression list $E = E'$ such that* GetFunParamAssign(P, E) $= s$ *and* GetFunParamAssign(P', E') $= s'$ *if $P = P'$, and $E = E'$, then $s = s'$.*

Proof. By definition of Algorithm GetFunParamAssign, the statement returned by GetFunParamAssign is deterministic based on the given input. □

Lemma 5.5 (CheckPublicEffects). *Given statement s, s', variable x, x', environment γ, γ', and memory σ, σ' such that* CheckPublicEffects(s, x, γ, σ) $= n$ *and* CheckPublicEffects(s', x', γ', σ') $= n'$ *if $s = s'$, $x = x'$, $\gamma = \gamma'$, and $\sigma = \sigma'$, then $n = n'$.*

Proof. By definition of Algorithm CheckPublicEffects, the value returned by CheckPublicEffects is deterministic based on the given input. □

Lemma 5.6 (τ). *Given type ty, ty' such that $\tau(ty) = n$ and $\tau(ty') = n'$ if $ty = ty'$ then $n = n'$.*

Proof. By definition of Algorithm τ, the value returned by τ is deterministic based on the given input. □

Lemma 5.7 (Cast). *Given type ty, ty', privacy label a, a', and value n_1, n'_1 such that $n_2 =$* Cast(a, ty, n_1) *and $n'_2 =$* Cast(a', ty', n'_1) *if $ty = ty'$, $a = a'$, and $n_1 = n'_1$, then $n_2 = n'_2$.*

Proof. By definition of Algorithm Cast, the value returned by Cast is deterministic based on the given input. □

Lemma 5.8 (Free). *Given memory σ, σ' and memory block identifier l, l' such that* Free(σ, l) $= (\sigma_1, (l, 0))$ *and* Free(σ', l') $= (\sigma'_1, (l', 0))$ *if $\sigma = \sigma'$ and $l = l'$, then $\sigma_1 = \sigma'_1$.*

Proof. By definition of Algorithm Free, the memory returned by Free is deterministic based on the given input. □

Lemma 5.9 (IncrementList). *Given location list L_1, L'_1, type* private $bty*$, private $bty'*$*, and memory σ, σ' such that* IncrementList(L_1, τ(private $bty*$), σ) $= (L_2, j)$ *and* IncrementList(L'_1, τ(private $bty'*$), σ) $= (L'_2, j')$ *if $L_1 = L'_1$, $bty = bty'$, and $\sigma = \sigma'$, then $L_2 = L'_2$ and $j = j'$.*

Proof. By definition of Algorithm IncrementList, the location list and tag returned by IncrementList is deterministic based on the given input. □

Lemma 5.10 (GetLocation). *Given locations $(l_1, \mu_1), (l'_1, \mu'_1)$, type $a\ bty*$, $a\ bty'*$, and memory σ, σ' such that $((l_2, \mu_2), j) =$* GetLocation((l_1, μ_1), $\tau(a\ bty'*)$, σ) *and $((l'_2, \mu'_2), j') =$* GetLocation((l'_1, μ'_1), $\tau(a'\ bty'*)$, σ') *if $l_1 = l'_1$, $\mu_1 = \mu'_1$, $a\ bty = a'\ bty'$, and $\sigma = \sigma'$, then $l_2 = l'_2$, $\mu_2 = \mu'_2$, and $j = j'$.*

Proof. By definition of Algorithm GetLocation, the location and tag returned by GetLocation is deterministic based on the given input. □

Lemma 5.11 (ReadOOB). *Given index i, i', number α, α', location l_1, l'_1, type $ty, ty' \in \{a\ bty\}$, and memory σ, σ' such that* ReadOOB($i, \alpha, l_1, ty, \sigma$) $= (n, j, (l_2, \mu))$ *and* ReadOOB($i', \alpha', l'_1, ty', \sigma'$) $= (n', j', (l'_2, \mu'))$*, if $i = i'$, $\alpha = \alpha'$, $l_1 = l'_1$, $ty = ty'$, and $\sigma = \sigma'$, then $n = n'$, $j = j'$, and $(l_2, \mu) = (l'_2, \mu')$.*

Proof. By definition of Algorithm ReadOOB, the value, tag, and location returned by ReadOOB is deterministic based on the given input. □

Lemma 5.12 (WriteOOB). *Given index* i, i'*, number* α, α', n, n'*, location* l_1, l_1'*, type* $ty, ty' \in \{a\ bty\}$*, and memory* σ_1, σ_1'*, location map* Δ_1, Δ_1'*, and accumulator* $\mathrm{acc}, \mathrm{acc}'$ *such that* $\mathrm{WriteOOB}(n,\ i,\ \alpha,\ l_1,\ ty,\ \sigma_1,\ \Delta_1,\ \mathrm{acc}) = (\sigma_2,\ \Delta_2, j, (l_2, \mu))$ *and* $\mathrm{WriteOOB}(n',\ i',\ \alpha',\ l_1',\ ty',\ \sigma_1',\ \Delta_1', \mathrm{acc}') = (\sigma_2',\ \Delta_2', j', (l_2', \mu'))$*, if* $i = i'$*,* $n = n'$*,* $\alpha = \alpha'$*,* $l_1 = l_1'$*,* $ty = ty'$*,* $\sigma_1 = \sigma_1'$*,* $\Delta_1 = \Delta_1'$*, and* $\mathrm{acc} = \mathrm{acc}'$*, then* $\sigma_2 = \sigma_2'$*,* $\Delta_2 = \Delta_2'$*,* $j = j'$*, and* $(l_2, \mu) = (l_2', \mu')$*.*

Proof. By definition of Algorithm WriteOOB, the memory, location map, tag, and location returned by WriteOOB is deterministic based on the given input. □

Lemma 5.13 (GetIndirection). *Given* $*, *'$ *such that* $\mathrm{GetIndirection}(*) = i$ *and* $\mathrm{GetIndirection}(*') = i'$*, if* $|*| = |*'|$ *then* $i = i'$*.*

Proof. By definition of Algorithm GetIndirection, the level of indirection returned by GetIndirection is deterministic based on the given input, as GetIndirection counts and returns the number of * to allow for any level of indirection for pointers within our semantics. □

Lemma 5.14 (DerefPtr). *Given memory* σ, σ'*, type* ty, ty'*, and location* $(l_1, \mu_1), (l_1', \mu_1')$ *such that* $\mathrm{DerefPtr}(\sigma, ty, (l_1, \mu_1)) = (n, j)$ *and* $\mathrm{DerefPtr}(\sigma', ty', (l_1', \mu_1')) = (n', j')$*, if* $\sigma = \sigma'$*,* $ty = ty'$*, and* $(l_1, \mu_1) = (l_1', \mu_1')$*, then* $n = n'$ *and* $j = j'$*.*

Proof. By definition of Algorithm DerefPtr, the value and tag (which indicates whether the access is aligned) that are returned by DerefPtr is deterministic based on the given input, and if all elements of the input are equivalent, then the output will also be equivalent. □

Lemma 5.15 (DerefPtrHLI). *Given memory* σ, σ'*, type* ty, ty'*, and location* $(l_1, \mu_1), (l_1', \mu_1')$ *such that* $\mathrm{DerefPtrHLI}(\sigma, ty, (l_1, \mu_1)) = ([\alpha,\ L,\ J,\ i], j)$ *and* $\mathrm{DerefPtrHLI}(\sigma', ty', (l_1', \mu_1')) = ([\alpha',\ L',\ J',\ i'], j')$*, if* $\sigma = \sigma'$*,* $ty = ty'$*, and* $(l_1, \mu_1) = (l_1', \mu_1')$*, then* $[\alpha,\ L,\ J,\ i] = [\alpha',\ L',\ J',\ i']$ *and* $j = j'$*.*

Proof. By definition of Algorithm DerefPtrHLI, the value and tag (which indicates whether the access is aligned) that are returned by DerefPtrHLI is deterministic based on the given input, and if all elements of the input are equivalent, then the output will also be equivalent. □

Lemma 5.16 (Extract). *Given statement* s_1, s_2, s_1', s_2' *such that* $\mathrm{Extract}(s_1, s_2) = (x_{list}, j)$ *and* $\mathrm{Extract}(s_1', s_2') = (x_{list}', j')$ *if* $s_1 = s_1'$ *and* $s_2 = s_2'$*, then* $x_{list} = x_{list}'$ *and* $j = j'$*.*

Proof. By definition of Algorithm Extract, the variable list and tag returned by Extract is deterministic based on the given input. □

Lemma 5.17 (InitializeVariables). *Given variable list* x_{list}, x_{list}'*, environment* γ_1, γ_1'*, memory* σ_1, σ_1'*, value* n, n' *and accumulator* $\mathrm{acc}, \mathrm{acc}'$ *such that* $\mathrm{InitializeVariables}(x_{list}, \gamma_1, \sigma_1, n, \mathrm{acc}) = (\gamma_2, \sigma_2, L)$ *and* $\mathrm{InitializeVariables}(x_{list}', \gamma_1', \sigma_1', n', \mathrm{acc}') = (\gamma_2', \sigma_2', L')$ *if* $x_{list} = x_{list}'$*,* $\gamma_1 = \gamma_1'$*,* $\sigma_1 = \sigma_1'$*,* $n = n'$*, and* $\mathrm{acc} = \mathrm{acc}'$*, then* $\gamma_2 = \gamma_2'$*,* $\sigma_2 = \sigma_2'$*, and* $L = L'$*.*

Proof. By definition of Algorithm InitializeVariables, the environment, memory, and location list returned by InitializeVariables are deterministic based on the given input. □

Lemma 5.18 (RestoreVariables). *Given variable list* x_{list}, x_{list}'*, environment* γ, γ'*, memory* σ_1, σ_1'*, and accumulator* $\mathrm{acc}, \mathrm{acc}'$ *such that*
$\mathrm{RestoreVariables}(x_{list}, \gamma, \sigma_1, \mathrm{acc}) = (\sigma_2, L)$ *and* $\mathrm{RestoreVariables}(x_{list}', \gamma', \sigma_1', \mathrm{acc}') = (\sigma_2', L')$ *if* $x_{list} = x_{list}'$*,* $\gamma = \gamma'$*,* $\sigma_1 = \sigma_1'$*, and* $\mathrm{acc} = \mathrm{acc}'$*, then* $\sigma_2 = \sigma_2'$ *and* $L = L'$*.*

Proof. By definition of Algorithm RestoreVariables, the memory and location list returned by RestoreVariables are deterministic based on the given input. □

Lemma 5.19 (ResolveVariables_Retrieve). *Given variable list* x_{list}, x'_{list}, *accumulator* acc, acc′, *environment* γ, γ', *and memory* σ, σ', *such that* ResolveVariables_Retrieve(x_{list}, acc, γ, σ) = $([(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})], n, L)$ *and* ResolveVariables_Retrieve(x'_{list}, acc′, γ', σ') = $([(v'_{t1}, v'_{e1}), ..., (v'_{tm}, v'_{em})], n', L')$ *if* $x_{list} = x'_{list}$ *and* acc = acc′, *then* $[(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})] = [(v'_{t1}, v'_{e1}), ..., (v'_{tm}, v'_{em})]$ $n = n'$, *and* $L = L'$.

Proof. By definition of Algorithm ResolveVariables_Retrieve, the value list, number, and location list returned by ResolveVariables_Retrieve are deterministic based on the given input. □

Lemma 5.20 (ResolveVariables_Store). *Given variable list* x_{list}, x'_{list}, *accumulator* acc, acc′, *environment* γ, γ', *memory* σ_1, σ'_1, *and value list* $[v_1, ..., v_m], [v'_1, ..., v'_m]$, *such that* ResolveVariables_Store(x_{list}, acc, $\gamma, \sigma_1, [v_1, ..., v_m]$) = (σ_2, L) *and* ResolveVariables_Store(x'_{list}, acc′, $\gamma', \sigma'_1, [v'_1, ..., v'_m]$) = (σ'_2, L') *if* $x_{list} = x'_{list}$, acc = acc′, $\gamma = \gamma'$, $\sigma_1 = \sigma'_1$, *and* $[v_1, ..., v_m] = [v'_1, ..., v'_m]$ *then* $\sigma_2 = \sigma'_2$ *and* $L = L'$.

Proof. By definition of Algorithm ResolveVariables_Store, the memory and location list returned by ResolveVariables_Store are deterministic based on the given input. □

Lemma 5.21 (Initialize). *Given location map* Δ_1, Δ'_1, *variable list* x_{list}, x'_{list}, *environment* γ_1, γ'_1, *memory* σ_1, σ'_1, *value* n, n', *and accumulator* acc, acc′, *such that* Initialize(Δ_1, x_{list}, γ_1, σ_1, n, acc) = $(\gamma_2, \sigma_2, \Delta_2, L)$ *and* Initialize(Δ'_1, x'_{list}, γ'_1, σ'_1, n', acc) = $(\gamma'_2, \sigma'_2, \Delta'_2, L')$ *if* $\Delta_1 = \Delta'_1$, $x_{list} = x'_{list}$, $\gamma_1 = \gamma'_1$, $\sigma_1 = \sigma'_1$, $n = n'$ *and* acc = acc′ *then* $\gamma_2 = \gamma'_2$, $\sigma_2 = \sigma'_2$, $\Delta_2 = \Delta'_2$ *and* $L = L'$.

Proof. By definition of Algorithm Initialize, the environment, memory, location map, and location list returned by Initialize is deterministic based on the given input. □

Lemma 5.22 (Restore). *Given memory* σ_1, σ'_1, *location map* Δ_1, Δ'_1, *and accumulator* acc, acc′, *such that* Restore(σ_1, Δ_1, acc) = (σ_2, Δ_2, L) *and* Restore(σ'_1, Δ'_1, acc′) = $(\sigma'_2, \Delta'_2, L')$ *if* $\sigma_1 = \sigma'_1$, $\Delta_1 = \Delta'_1$, *and* acc = acc′ *then* $\sigma_2 = \sigma'_2$, $\Delta_2 = \Delta'_2$, *and* $L = L'$.

Proof. By definition of Algorithm Restore, the memory, location map, and location list returned by Restore is deterministic based on the given input. □

Lemma 5.23 (Resolve_Retrieve). *Given environment* γ, γ', *memory* σ, σ', *location map* Δ, Δ', *and accumulator* acc, acc′, *such that* Resolve_Retrieve(γ, σ, Δ, acc) = $([(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})], n, L)$ *and* Resolve_Retrieve($\gamma', \sigma', \Delta'$, acc′) = $([(v'_{t1}, v'_{e1}), ..., (v'_{tm}, v'_{em})], n', L')$ *if* $\gamma = \gamma'$, $\sigma = \sigma'$, $\Delta = \Delta'$, *and* acc = acc′, *then* $[(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})] = [(v'_{t1}, v'_{e1}), ..., (v'_{tm}, v'_{em})]$, $n = n'$, *and* $L = L'$.

Proof. By definition of Algorithm Resolve_Retrieve, the value list, value, and location list returned by Resolve_Retrieve is deterministic based on the given input. □

Lemma 5.24 (Resolve_Store). *Given memory* σ_1, σ'_1, *location map* Δ_1, Δ'_1, *accumulator* acc, acc′, *and values* $[v_1, ..., v_m]$, $[v'_1, ..., v'_m]$, *such that* Resolve_Store(Δ_1, σ_1, acc, $[v_1, ..., v_m]$) = (σ_2, Δ_2, L) *and* Resolve_Store(Δ'_1, σ'_1, acc′, $[v'_1, ..., v'_m]$) = $(\sigma'_2, \Delta'_2, L')$ *if* $\sigma_1 = \sigma'_1$, $\Delta_1 = \Delta'_1$, acc = acc′, *and* $[v_1, ..., v_m] = [v'_1, ..., v'_m]$ *then* $\sigma_2 = \sigma'_2$, $\Delta_2 = \Delta'_2$, *and* $L = L'$

Proof. By definition of Algorithm Resolve_Store, the memory, location map, and location list returned by Resolve_Store is deterministic based on the given input. □

Lemma 5.25 (DynamicUpdate). *Given memory* σ, σ', *location map* Δ_1, Δ'_1, *location list* L_1, L'_1, *and type* $ty, ty' \in \{$private *a bty*, private *a bty*∗$\}$, *such that* DynamicUpdate(Δ_1, σ, L_1, acc, ty) = (Δ_2, L_2) *and* DynamicUpdate(Δ'_1, σ', L'_1, acc, ty') = (Δ'_2, L'_2) *if* $\sigma = \sigma'$, $\Delta_1 = \Delta'_1$, $L_1 = L'_1$, acc = acc′, *and* $ty = ty'$, *then* $\Delta_2 = \Delta'_2$, *and* $L_2 = L'_2$.

Proof. By definition of Algorithm DynamicUpdate, the location map and location list returned by DynamicUpdate is deterministic based on the given input. □

Lemma 5.26 (DecodePtr). *Given type ty, ty', value α, α', and bytes ω, ω' such that* $\mathrm{DecodePtr}(ty,\ \alpha,\ \omega) = [\alpha,\ L,\ J,\ i]$ *and* $\mathrm{DecodePtr}(ty',\ \alpha',\ \omega') = [\alpha',\ L',\ J',\ i']$, *if $ty = ty'$, $\alpha = \alpha'$, and $\omega = \omega'$, then $L = L'$, $J = J'$, and $i = i'$.*

Proof. By definition of Algorithm DecodePtr, the pointer data structure returned by DecodePtr is deterministic based on the given input. □

Lemma 5.27 (DecodeArr). *Given type $a\ bty, a'\ bty'$, index i, i', and bytes ω, ω' such that* $\mathrm{DecodeArr}(a\ bty, i, \omega) = n$ *and* $\mathrm{DecodeArr}(a'\ bty', i', \omega') = n'$ *if $a = a'$, $bty = bty'$, $i = i'$, and $\omega = \omega'$, then $n = n'$.*

Proof. By definition of Algorithm DecodeArr, the value returned by DecodeArr is deterministic based on the given input. □

Lemma 5.28 (DecodeFun). *Given bytes ω, ω' such that* $\mathrm{DecodeFun}(\omega) = (s,\ n,\ P)$ *and* $\mathrm{DecodeFun}(\omega') = (s',\ n',\ P')$ *if $\omega = \omega'$, then $s = s'$, $n = n'$, and $P = P'$.*

Proof. By definition of Algorithm DecodeFun, the statement, tag, and parameter list returned by DecodeFun is deterministic based on the given input. □

Lemma 5.29 (DecodeVal). *Given type $a\ bty, a'\ bty'$ and bytes ω, ω' such that* $\mathrm{DecodeVal}(a\ bty, \omega) = n$ *and* $\mathrm{DecodeVal}(a'\ bty', \omega') = n'$ *if $a = a'$, $bty = bty'$, and $\omega = \omega'$, then $n = n'$.*

Proof. By definition of Algorithm DecodeVal, the value returned by DecodeVal is deterministic based on the given input. □

Lemma 5.30 (EncodeVal). *Given type $ty, ty' \in \{a\ bty\}$ and value $v, v' \in \{n, \mathrm{NULL}\}$ such that* $\omega = \mathrm{EncodeVal}(ty, v)$ *and* $\omega' = \mathrm{EncodeVal}(ty', v')$ *if $ty = ty'$ and $v = v'$ then $\omega = \omega'$.*

Proof. By definition of Algorithm EncodeVal, the byte representation returned by EncodeVal is deterministic based on the given input. □

Lemma 5.31 (EncodeArr). *Given type $ty, ty' \in \{a\ bty\}$, index i, i', number α, α', and value $v, v' \in \{n, \mathrm{NULL}\}$ such that* $\omega = \mathrm{EncodeArr}(ty, i, \alpha, v)$ *and* $\omega' = \mathrm{EncodeArr}(ty', i', \alpha', v')$ *if $ty = ty'$, $i = i'$, $\alpha = \alpha'$, and $v = v'$, then $\omega = \omega'$.*

Proof. By definition of Algorithm EncodeArr, the byte representation returned by EncodeArr is deterministic based on the given input. □

Lemma 5.32 (EncodePtr). *Given type $ty, ty' \in \{a\ bty{*}, a\ \mathrm{const}\ bty{*}\}$, number of locations α, α', location list L, L', tag list J, J', and level of indirection i, i' such that* $\omega = \mathrm{EncodePtr}(ty,\ [\alpha,\ L,\ J,\ i])$ *and* $\omega' = \mathrm{EncodePtr}(ty',\ [\alpha',\ L',\ J',\ i'])$ *if $ty = ty'$, $\alpha = \alpha'$, $L = L'$, $J = J'$, and $i = i'$, then $\omega = \omega'$.*

Proof. By definition of Algorithm EncodePtr, the byte representation returned by EncodePtr is deterministic based on the given input. □

Lemma 5.33 (EncodeFun). *Given statement s, s', value n, n', and parameter list P, P' such that* $\mathrm{EncodeFun}(s,\ n,\ P) = \omega$ *and* $\mathrm{EncodeFun}(s',\ n',\ P') = \omega'$, *if $s = s'$, $n = n'$, and $P = P'$, then $\omega = \omega'$.*

Proof. By definition of Algorithm EncodeFun, the byte representation returned by EncodeFun is deterministic based on the given input. □

Lemma 5.34 (UpdateVal). *Given memory σ_1, σ_1', memory block identifier l, l', value n, n', and type $a\ bty$, $a'\ bty'$ such that* $\mathrm{UpdateVal}(\sigma_1,\ l,\ n,\ a\ bty) = \sigma_2$ *and* $\mathrm{UpdateVal}(\sigma_1',\ l',\ n',\ a'\ bty') = \sigma_2'$ *if $\sigma_1 = \sigma_1'$, $l = l'$, $n = n'$, $a = a'$, and $bty = bty'$, then $\sigma_2 = \sigma_2'$.*

Proof. By definition of Algorithm UpdateVal, the memory returned by UpdateVal is deterministic based on the given input. □

Lemma 5.35 (UpdateArr). *Given memory σ_1, σ_1', memory block identifier l, l', index i, i', value n, n', and type $a\ bty, a'\ bty'$ such that* $\mathrm{UpdateArr}(\sigma_1,\ (l, i),\ n,\ a\ bty) = \sigma_2$ *and* $\mathrm{UpdateArr}(\sigma_1',\ (l', i'),\ n',\ a'\ bty') = \sigma_2'$ *if $\sigma_1 = \sigma_1'$, $l = l'$, $i = i'$, $n = n'$, $a = a'$, and $bty = bty'$, then $\sigma_2 = \sigma_2'$.*

Proof. By definition of Algorithm UpdateArr, the memory returned by UpdateArr is deterministic based on the given input. ☐

Lemma 5.36 (UpdatePtr). *Given memory* σ_1, σ'_1, *location* $(l, \mu), (l', \mu')$, *pointer data structure* $[\alpha,\ L,\ J,\ i]$, $[\alpha',\ L',\ J',\ i']$, *and type* $a\ bty*, a'\ bty'*$ *such that* $\text{UpdatePtr}(\sigma_1,\ (l, \mu),\ [\alpha,\ L,\ J,\ i],\ a\ bty*) = (\sigma_2,\ j)$ *and* $\text{UpdatePtr}(\sigma'_1, (l', \mu'),\ [\alpha',\ L',\ J',\ i'],\ a'\ bty'*) = (\sigma'_2,\ j')$ *if* $\sigma_1 = \sigma'_1$, $l = l'$, $\mu = \mu'$ $\alpha = \alpha'$, $a = a'$, $bty = bty'$, $L = L'$, $J = J'$, *and* $i = i'$, *then* $\sigma_2 = \sigma'_2$ *and* $j = j'$.

Proof. By definition of Algorithm UpdatePtr, the memory and tag returned by UpdatePtr is deterministic based on the given input. ☐

Lemma 5.37 (UpdateOffset). *Given memory* σ_1, σ'_1, *location* $(l, \mu), (l', \mu')$, *number* n, n' *and type* $a\ bty, a'\ bty'$ *such that* $\text{UpdateOffset}(\sigma_1,\ (l, \mu),\ n,\ a\ bty) = (\sigma_2, j)$ *and* $\text{UpdateOffset}(\sigma'_1,\ (l', \mu'),\ n',\ a\ bty') = (\sigma'_2, j')$ *if* $\sigma_1 = \sigma'_1$, $l = l'$, $\mu = \mu'$ $n = n'$, $a = a'$, *and* $bty = bty'$, *then* $\sigma_2 = \sigma'_2$ *and* $j = j'$.

Proof. By definition of Algorithm UpdateOffset, the memory and tag returned by UpdateOffset is deterministic based on the given input. ☐

Lemma 5.38 ($\mathcal{D}_1 :: \mathcal{D}_2 = \mathcal{D}'_1 :: \mathcal{D}'_2$). *Given* $\mathcal{D}_1 :: \mathcal{D}_2$, $\mathcal{D}'_1 :: \mathcal{D}'_2$, *if* $\mathcal{D}_1 = \mathcal{D}'_1$ *and* $\mathcal{D}'_2 = \mathcal{D}'_2$ *then* $\mathcal{D}_1 :: \mathcal{D}_2 = \mathcal{D}'_1 :: \mathcal{D}'_2$.

Proof. By definition of Algorithm 31, the result of adding party-wise evaluation code lists is deterministic based on the content and ordering of the party-wise evaluation code lists. ☐

Lemma 5.39. *Given number* α, α', *location list* $\{L^{\text{p}}, L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *type* ty, ty', *and memory* $\{\sigma^{\text{p}}, \sigma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\text{Retrieve\_vals}(\alpha, L^{\text{p}}, ty, \sigma^{\text{p}}) = ([v_0^{\text{p}}, ...v_{\alpha-1}^{\text{p}}], j^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\text{Retrieve\_vals}(\alpha', L'^{\text{p}}, ty', \sigma'^{\text{p}}) = ([v_0'^{\text{p}}, ...v_{\alpha'-1}'^{\text{p}}], j'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *if* $\alpha = \alpha'$, $\{L^{\text{p}} = L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $ty = ty'$, *and* $\{\sigma^{\text{p}} = \sigma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *then* $\{\forall i \in \{0...\alpha - 1\}, v_i^{\text{p}} = v_i'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{j^{\text{p}} = j'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm Retrieve_vals, the values returned by Retrieve_vals are deterministic based on the given input. ☐

Lemma 5.40. *Given environment* $\{\gamma^{\text{p}}, \gamma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *location list* $\{L^{\text{p}}, L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *tag list* $\{J^{\text{p}}, J'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and memory* $\{\sigma^{\text{p}}, \sigma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\text{CheckFreeable}(\gamma^{\text{p}}, L^{\text{p}}, J^{\text{p}}, \sigma^{\text{p}}) = j\}_{\text{p}=1}^{\text{q}}$ *and* $\{\text{CheckFreeable}(\gamma'^{\text{p}}, L'^{\text{p}}, J'^{\text{p}}, \sigma'^{\text{p}}) = j'\}_{\text{p}=1}^{\text{q}}$ *if* $\{\gamma^{\text{p}} = \gamma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{L^{\text{p}} = L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{J^{\text{p}} = J'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and* $\{\sigma^{\text{p}} = \sigma'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *then* $j = j'$.

Proof. By definition of Algorithm CheckFreeable, the tag returned by CheckFreeable is deterministic based on the input. ☐

Lemma 5.41. *Given memory* $\{\sigma_1^{\text{p}}, \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *number* α, α', *location list* $\{L^{\text{p}}, L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and byte representations* $\{[\omega_0^{\text{p}}, ..., \omega_{\alpha-1}^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$, $\{[\omega_0'^{\text{p}}, ..., \omega_{\alpha'-1}'^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\text{UpdateBytesFree}(\sigma_1^{\text{p}}, L^{\text{p}}, [\omega_0^{\text{p}}, ..., \omega_{\alpha-1}^{\text{p}}]) = \sigma_2^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{\text{UpdateBytesFree}(\sigma_1^{\text{p}}, L'^{\text{p}}, [\omega_0'^{\text{p}}, ..., \omega_{\alpha'-1}'^{\text{p}}]) = \sigma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *if* $\{\sigma_1^{\text{p}} = \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{L^{\text{p}} = L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\alpha = \alpha'$, *and* $\{[\omega_0^{\text{p}}, ..., \omega_{\alpha-1}^{\text{p}}] = [\omega_0'^{\text{p}}, ..., \omega_{\alpha'-1}'^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$, *then* $\{\sigma_2^{\text{p}} = \sigma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm UpdateBytesFree, the memory returned by UpdateBytesFree is deterministic based on the input. ☐

Lemma 5.42. *Given memory* $\{\sigma_1^{\text{p}}, \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *location list* $\{L^{\text{p}}, L'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and tag list* $\{J^{\text{p}}, J'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *such that* $\{\text{UpdatePointerLocations}(\sigma_1^{\text{p}}, L_1^{\text{p}}[1 : \alpha - 1], J^{\text{p}}[1 : \alpha - 1], L_1^{\text{p}}[0], J^{\text{p}}[0]) = (\sigma_2^{\text{p}}, L_2^{\text{p}})\}_{\text{p}=1}^{\text{q}}$ *and* $\{\text{UpdatePointerLocations}(\sigma_1'^{\text{p}}, L_1'^{\text{p}}[1 : \alpha' - 1], J'^{\text{p}}[1 : \alpha' - 1], L_1'^{\text{p}}[0], J'^{\text{p}}[0]) = (\sigma_2'^{\text{p}}, L_2'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, *if* $\{\sigma_1^{\text{p}} = \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{L_1^{\text{p}} = L_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *and* $\{J^{\text{p}} = J'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, *then* $\{\sigma_2^{\text{p}} = \sigma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$ *and* $\{L_2^{\text{p}} = L_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Proof. By definition of Algorithm UpdatePointerLocations, the memory and location list returned by UpdatePointerLocations is deterministic based on the input. ☐

Lemma 5.43. *Given number α, α', location list $\{L^p, L'^p\}_{p=1}^q$, type ty, ty', values $\{[v_0^p, ..., v_{\alpha-1}^p], [v_0'^p, ..., v_{\alpha'-1}'^p]\}_{p=1}^q$, and memory $\{\sigma_1^p, \sigma_1'^p\}_{p=1}^q$ such that $\{\text{UpdateDerefVals}(\alpha, L^p, [v_0^p, ..., v_{\alpha-1}^p], ty, \sigma_1^p) = \sigma_2^p\}_{p=1}^q$ and $\{\text{UpdateDerefVals}(\alpha', L'^p, [v_0'^p, ..., v_{\alpha'-1}'^p], ty', \sigma_1'^p) = \sigma_2'^p\}_{p=1}^q$, if $\alpha = \alpha'$, $\{L^p = L'^p\}_{p=1}^q$, $ty = ty'$, $\{[v_0^p, ..., v_{\alpha-1}^p] = [v_0'^p, ..., v_{\alpha'-1}'^p]\}_{p=1}^q$, and $\{\sigma_1^p = \sigma_1'^p\}_{p=1}^q$, then $\{\sigma_2^p = \sigma_2'^p\}_{p=1}^q$.*

Proof. By definition of Algorithm UpdateDerefVals, the memory returned by UpdateDerefVals is deterministic based on the input. □

Axiom 5.5 (MPC$_{ar}$). *Given indices $\{i^p, i'^p\}_{p=1}^q$, arrays $\{[n_0^p, ..., n_{\alpha-1}^p], [n_0'^p, ..., n_{\alpha'-1}'^p]\}_{p=1}^q$,*
if $\text{MPC}_{ar}((i^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^q, [n_0^q, ..., n_{\alpha-1}^q])) = (n^1, ..., n^q)$,
$\text{MPC}_{ar}((i'^1, [n_0'^1, ..., n_{\alpha'-1}'^1]), ..., (i'^q, [n_0'^q, ..., n_{\alpha'-1}'^q])) = (n'^1, ..., n'^q)$,
$\{i^p = i'^p\}_{p=1}^q$, *and* $\{[n_0^p, ..., n_{\alpha-1}^p] = [n_0'^p, ..., n_{\alpha'-1}'^p]\}_{p=1}^q$
then $\{n^p = n'^p\}_{p=1}^q$.

Axiom 5.6 (MPC$_{aw}$). *Given indices $\{i^p, i'^p\}_{p=1}^q$, arrays $\{[n_0^p, ..., n_{\alpha-1}^p], [n_0''^p, ..., n_{\alpha'-1}''^p]\}_{p=1}^q$, and values $\{n^p, n'^p\}_{p=1}^q$,*
if $\text{MPC}_{aw}((i^1, n^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^q, n^q, [n_0^q, ..., n_{\alpha-1}^q])) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^q, ..., n_{\alpha-1}'^q])$,
$\text{MPC}_{aw}((i'^1, n'^1, [n_0''^1, ..., n_{\alpha'-1}''^1]), ..., (i'^q, n'^q, [n_0''^q, ..., n_{\alpha'-1}''^q])) = ([n_0'''^1, ..., n_{\alpha'-1}'''^1], ..., [n_0'''^q, ..., n_{\alpha'-1}'''^q])$,
$\{i^p = i'^p\}_{p=1}^q$, $\{n^p = n'^p\}_{p=1}^q$ *and* $\{[n_0^p, ..., n_{\alpha-1}^p] = [n_0''^p, ..., n_{\alpha'-1}''^p]\}_{p=1}^q$
then $\{[n_0'^p, ..., n_{\alpha-1}'^p] = [n_0'''^p, ..., n_{\alpha'-1}'''^p]\}_{p=1}^q$.

Axiom 5.7 (MPC$_b$). *Given values $\{v_1^p, v_2^p, v_3^p, v_1'^p, v_2'^p, v_3'^p\}_{p=1}^q$ and binary operation $bop \in \{\cdot, +, -, \div\}$,*
if $\text{MPC}_b(bop, v_1^1, v_2^1, ..., v_1^q, v_2^q) = (v_3^1, ..., v_3^q)$,
$\text{MPC}_b(bop, v_1'^1, v_2'^1, ..., v_1'^q, v_2'^q) = (v_3'^1, ..., v_3'^q)$, $\{v_1^p = v_1'^p\}_{p=1}^q$, *and* $\{v_2^p = v_2'^p\}_{p=1}^q$
then $\{v_3^p = v_3'^p\}_{p=1}^q$.

Axiom 5.8 (MPC$_{cmp}$). *Given values $\{v_1^p, v_2^p, v_3^p, v_1'^p, v_2'^p, v_3'^p\}_{p=1}^q$ and binary operation $bop \in \{==, !=, <\}$,*
if $\text{MPC}_{cmp}(bop, v_1^1, v_2^1, ..., v_1^q, v_2^q) = (v_3^1, ..., v_3^q)$,
$\text{MPC}_{cmp}(bop, v_1'^1, v_2'^1, ..., v_1'^q, v_2'^q) = (v_3'^1, ..., v_3'^q)$, $\{v_1^p = v_1'^p\}_{p=1}^q$, *and* $\{v_2^p = v_2'^p\}_{p=1}^q$
then $\{v_3^p = v_3'^p\}_{p=1}^q$.

Axiom 5.9 (MPC$_u$). *Given values $\{n_1^p, n_1'^p\}_{p=1}^q$ and binary operation $uop \in \{++\}$,*
if $\text{MPC}_u(uop, n_1^1, ..., n_1^q) = (n_2^1, ..., n_2^q)$,
$\text{MPC}_u(uop, n_1'^1, ..., n_1'^q) = (n_2'^1, ..., n_2'^q)$, *and* $\{n_1^p = n_1'^p\}_{p=1}^q$,
then $\{n_2^p = n_2'^p\}_{p=1}^q$.

Axiom 5.10 (MPC$_{resolve}$). *Given values $\{n^1, n'^p, [(v_{t1}^p, v_{e1}^p), ..., (v_{tm}^p, v_{em}^p)], [(v_{t1}'^1, v_{e1}'^1), ..., (v_{tm}'^1, v_{em}'^1)]\}_{p=1}^q$,*
if $\text{MPC}_{resolve}([n^1, ..., n^q], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^q, v_{e1}^q), ..., (v_{tm}^q, v_{em}^q)]])$
$= [[v_1^1, ..., v_m^1], ..., [v_1^q, ..., v_m^q]]$
$\text{MPC}_{resolve}([n'^1, ..., n'^q], [[(v_{t1}'^1, v_{e1}'^1), ..., (v_{tm}'^1, v_{em}'^1)], ..., [(v_{t1}'^q, v_{e1}'^q), ..., (v_{tm}'^q, v_{em}'^q)]])$
$= [[v_1'^1, ..., v_m'^1], ..., [v_1'^q, ..., v_m'^q]]$,
$\{n^p = n'^p\}_{p=1}^q$ *and* $\{[(v_{t1}^p, v_{e1}^p), ..., (v_{tm}^p, v_{em}^p)] = [(v_{t1}'^p, v_{e1}'^p), ..., (v_{tm}'^p, v_{em}'^p)]\}_{p=1}^q$,
then $\{[v_1^p, ..., v_m^p] = [v_1^p, ..., v_m^p]\}_{p=1}^q$.

Axiom 5.11 (MPC$_{dv}$). *Given values $\{[n_0^p, ..., n_{\alpha-1}^p]\}_{p=1}^q$, and tag lists $\{J^p, J'^p\}_{p=1}^q$,*
if $\text{MPC}_{dv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^q, ..., n_{\alpha-1}^q]], [J^1, ..., J^q]) = (n^1, ..., n^q)$,
$\text{MPC}_{dv}([[n_0'^1, ..., n_{\alpha'-1}'^1], ..., [n_0'^q, ..., n_{\alpha'-1}'^q]], [J'^1, ..., J'^q]) = (n'^1, ..., n'^q)$,

$\{[n_0^{\mathrm{p}}, ..., n_{\alpha-1}^{\mathrm{p}}] = [n_0'^{\mathrm{p}}, ..., n_{\alpha'-1}'^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and* $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$
then $\{n^{\mathrm{p}} = n'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Axiom 5.12 (MPC$_{dp}$). *Given values* $\{\forall i \in \{0...\alpha-1\}[\alpha_i,\ L_i^{\mathrm{p}},\ J_i^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\forall j \in \{0...\alpha'-1\}[\alpha_j',\ L_j'^{\mathrm{p}},\ J_j'^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$ *and tag lists* $\{J^{\mathrm{p}}, J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$,
if $\mathrm{MPC}_{dp}([[[\alpha_0,\ L_0^1,\ J_0^1], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1]], ..., [[\alpha_0,\ L_0^{\mathrm{q}},\ J_0^{\mathrm{q}}], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{q}},\ J_{\alpha-1}^{\mathrm{q}}]]], [J^1, ..., J^{\mathrm{q}}]) = ([[\alpha_\alpha,\ L_\alpha^1,\ J_\alpha^1], ..., [\alpha_\alpha,\ L_\alpha^{\mathrm{q}},\ J_\alpha^{\mathrm{q}}]])$,
$\mathrm{MPC}_{dp}([[[\alpha_0',\ L_0'^1,\ J_0'^1], ..., [\alpha_{\alpha'-1}',\ L_{\alpha'-1}'^1, J_{\alpha'-1}'^1]], ..., [[\alpha_0',\ L_0'^{\mathrm{q}},\ J_0'^{\mathrm{q}}], ..., [\alpha_{\alpha'-1}',\ L_{\alpha'-1}'^{\mathrm{q}},\ J_{\alpha'-1}'^{\mathrm{q}}]]], [J'^1, ..., J'^{\mathrm{q}}]) = ([[\alpha_{\alpha'}',\ L_{\alpha'}'^1,\ J_{\alpha'}'^1], ..., [\alpha_{\alpha'}',\ L_{\alpha'}'^{\mathrm{q}},\ J_{\alpha'}'^{\mathrm{q}}]])$,
$\alpha = \alpha'$, $i = i'$, $\{\forall i \in \{0...\alpha-1\}, [\alpha_i,\ L_i^{\mathrm{p}},\ J_i^{\mathrm{p}}] = [\alpha_i',\ L_i'^{\mathrm{p}},\ J_i'^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and* $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$
then $\{[\alpha_\alpha,\ L_\alpha^{\mathrm{p}},\ J_\alpha^{\mathrm{p}}] = [\alpha_{\alpha'}',\ L_{\alpha'}'^{\mathrm{p}},\ J_{\alpha'}'^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Axiom 5.13 (MPC$_{free}$). *Given number* α, α', *bytes* $\{[\omega_0^{\mathrm{p}}, ..., \omega_{\alpha-1}^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{[\omega_0''^{\mathrm{p}}, ..., \omega_{\alpha'-1}''^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and tag lists* $\{J^{\mathrm{p}}, J''^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$,
if $\mathrm{MPC}_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^{\mathrm{q}}, ..., \omega_{\alpha-1}^{\mathrm{q}}]], [J^1, ... J^{\mathrm{q}}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^{\mathrm{q}}, ..., \omega_{\alpha-1}'^{\mathrm{q}}]], [J'^1, ..., J'^{\mathrm{q}}])$,
$\mathrm{MPC}_{free}([[\omega_0''^1, ..., \omega_{\alpha'-1}''^1], ..., [\omega_0''^{\mathrm{q}}, ..., \omega_{\alpha'-1}''^{\mathrm{q}}]], [J''^1, ... J''^{\mathrm{q}}]) = ([[\omega_0'''^1, ..., \omega_{\alpha'-1}'''^1], ..., [\omega_0'''^{\mathrm{q}}, ..., \omega_{\alpha'-1}'''^{\mathrm{q}}]], [J'''^1, ..., J'''^{\mathrm{q}}])$, $\alpha = \alpha'$, $\{[\omega_0^{\mathrm{p}}, ..., \omega_{\alpha-1}^{\mathrm{p}}] = [\omega_0''^{\mathrm{p}}, ..., \omega_{\alpha'-1}''^{\mathrm{p}}]]\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and* $\{J^{\mathrm{p}} = J''^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$
then $\{[\omega_0'^{\mathrm{p}}, ..., \omega_{\alpha-1}'^{\mathrm{p}}] = [\omega_0'''^{\mathrm{p}}, ..., \omega_{\alpha'-1}'''^{\mathrm{p}}]]\}_{\mathrm{p}=1}^{\mathrm{q}}$ *and* $\{J'^{\mathrm{p}} = J'''^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Axiom 5.14 (MPC$_{wdv}$). *Given list of values* $\{[n_0^{\mathrm{p}}, ..., n_{\alpha-1}^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{[n_0''^{\mathrm{p}}, ..., n_{\alpha'-1}''^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, *number* α, α', $\{n^{\mathrm{p}}, n'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and tag list* $\{J^{\mathrm{p}}, J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$,
if $\mathrm{MPC}_{wdv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^{\mathrm{q}}, ..., n_{\alpha-1}^{\mathrm{q}}]], [n^1, ..., n^{\mathrm{q}}], [J^1, ..., J^{\mathrm{q}}]) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^{\mathrm{q}}, ..., n_{\alpha-1}'^{\mathrm{q}}])$,
$\mathrm{MPC}_{wdv}([[n_0''^1, ..., n_{\alpha'-1}''^1], ..., [n_0''^{\mathrm{q}}, ..., n_{\alpha'-1}''^{\mathrm{q}}]], [n'^1, ..., n'^{\mathrm{q}}], [J'^1, ..., J'^{\mathrm{q}}]) = ([n_0'''^1, ..., n_{\alpha'-1}'''^1], ..., [n_0'''^{\mathrm{q}}, ..., n_{\alpha'-1}'''^{\mathrm{q}}])$, $\{[n_0^{\mathrm{p}}, ..., n_{\alpha-1}^{\mathrm{p}}] = [n_0''^{\mathrm{p}}, ..., n_{\alpha'-1}''^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\alpha = \alpha'$, $\{n^{\mathrm{p}} = n'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$ *and* $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$
then $\{[n_0'^{\mathrm{p}}, ..., n_{\alpha-1}'^{\mathrm{p}}] = [n_0'''^{\mathrm{p}}, ..., n_{\alpha'-1}'''^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Axiom 5.15 (MPC$_{wdp}$). *Given location list* $\{[\alpha_e, L_e^{\mathrm{p}}, J_e^{\mathrm{p}}, i], [\alpha_e', L_e'^{\mathrm{p}}, J_e'^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\forall m \in \{0...\alpha-1\},\ [\alpha_m,\ L_m^{\mathrm{p}},\ J_m^{\mathrm{p}}, i]\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\forall m' \in \{0...\alpha''-1\},\ [\alpha_{m'}'',\ L_{m'}''^{\mathrm{p}},\ J_{m'}''^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and tag list* $\{J^{\mathrm{p}}, J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$,
if $\mathrm{MPC}_{wdp}([[[\alpha_e, L_e^1, J_e^1, i], [\alpha_0,\ L_0^1,\ J_0^1, i], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^1,\ J_{\alpha-1}^1, i]], ..., [[\alpha_e, L_e^{\mathrm{q}}, J_e^{\mathrm{q}}, i], [\alpha_0,\ L_0^{\mathrm{q}},\ J_0^{\mathrm{q}}, i], ..., [\alpha_{\alpha-1},\ L_{\alpha-1}^{\mathrm{q}},\ J_{\alpha-1}^{\mathrm{q}}, i]]], [J^1, ..., J^{\mathrm{q}}]) = [[[\alpha_0',\ L_0'^1,\ J_0'^1, i], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^1,\ J_{\alpha-1}'^1, i]], ..., [[\alpha_0', L_0'^{\mathrm{q}}, J_0'^{\mathrm{q}}, i], ..., [\alpha_{\alpha-1}',\ L_{\alpha-1}'^{\mathrm{q}}, J_{\alpha-1}'^{\mathrm{q}}, i]]]$, $\mathrm{MPC}_{wdp}([[[\alpha_e', L_e'^1, J_e'^1, i'], [\alpha_0'',\ L_0''^1,\ J_0''^1, i'], ..., [\alpha_{\alpha''-1}'',\ L_{\alpha''-1}''^1,\ J_{\alpha''-1}''^1, i']], ..., [[\alpha_e', L_e'^{\mathrm{q}}, J_e'^{\mathrm{q}}, i'], [\alpha_0'',\ L_0''^{\mathrm{q}},\ J_0''^{\mathrm{q}}, i'], ..., [\alpha_{\alpha''-1}'',\ L_{\alpha''-1}''^{\mathrm{q}},\ J_{\alpha''-1}''^{\mathrm{q}}, i']]], [J'^1, ..., J'^{\mathrm{q}}]) = [[[\alpha_0''',\ L_0'''^1,\ J_0'''^1, i'], ..., [\alpha_{\alpha''-1}''',\ L_{\alpha''-1}'''^1,\ J_{\alpha''-1}'''^1, i']], ..., [[\alpha_0''',\ L_0'''^{\mathrm{q}},\ J_0'''^{\mathrm{q}}, i'], ..., [\alpha_{\alpha''-1}''',\ L_{\alpha''-1}'''^{\mathrm{q}},\ J_{\alpha''-1}'''^{\mathrm{q}}, i']]]$, $\alpha = \alpha''$, $\{[\alpha_e, L_e^{\mathrm{p}}, J_e^{\mathrm{p}}, i] = [\alpha_e', L_e'^{\mathrm{p}}, J_e'^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\forall m \in \{0...\alpha-1\},\ [\alpha_m,\ L_m^{\mathrm{p}},\ J_m^{\mathrm{p}}, i] = [\alpha_{m'}'',\ L_{m'}''^{\mathrm{p}},\ J_{m'}''^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$, *and* $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$
then $\{\forall m \in \{0...\alpha-1\},\ [\alpha_m',\ L_m'^{\mathrm{p}},\ J_m'^{\mathrm{p}}, i] = [\alpha_{m'}''',\ L_{m'}'''^{\mathrm{p}},\ J_{m'}'''^{\mathrm{p}}, i']\}_{\mathrm{p}=1}^{\mathrm{q}}$.

5.3 Location Access Tracking Supporting Metatheory

Definition 5.8 (Location Access). A location in memory (l, μ) is defined to have been accessed if we look up memory block identifier l in memory σ and obtain or modify the data that is stored at offset μ from within memory block.

Definition 5.9 ($L = L'$). Two location lists are equivalent, in symbols $L = L'$, if and only if $(l, \mu) \in L \iff (l, \mu) \in L'$.

Definition 5.10 ($\mathcal{L} = \mathcal{L}'$). Two party-wise location lists are equivalent, in symbols $\mathcal{L} = \mathcal{L}'$, if and only if $(\mathrm{p}, L) \in \mathcal{L} \iff (\mathrm{p}, L) \in \mathcal{L}'$.

Lemma 5.44 ($\mathcal{L}_1, \mathcal{L}_2$). *Given two party-wise location lists* $\mathcal{L}_1, \mathcal{L}_2$, *if* $\mathcal{L}_1$ *was accessed first and* $\mathcal{L}_2$ *accessed second, then we have* $\mathcal{L}_1 :: \mathcal{L}_2$.

Proof. By the definition of Algorithm 30 and analysis of all rule cases. □

Lemma 5.45 ($(\mathrm{p}, L_1) :: (\mathrm{p}, L_1)$). *Given two party-wise location lists* $(\mathrm{p}, L_1), (\mathrm{p}, L_2)$, *if* $(\mathrm{p}, L_1) :: (\mathrm{p}, L_2)$, *then* $(\mathrm{p}, L_1 :: L_2)$.

Proof. By the definition of Algorithm 30. □

Lemma 5.46 ($\{(\mathrm{p}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}} :: \{(\mathrm{p}, L_2^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$). *Given* $\{(\mathrm{p}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ *and* $\{(\mathrm{p}, L_2^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ *if* $\{(\mathrm{p}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}} :: \{(\mathrm{p}, L_2^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$ *then* $(1, L_1^1 :: L_2^1) \parallel \ldots \parallel (\mathrm{q}, L_1^{\mathrm{q}} :: L_2^{\mathrm{q}})$.

Proof. By the definition of Algorithm 30. □

Lemma 5.47 ($\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$). *Given* $\mathcal{L}_1 :: \mathcal{L}_2$, $\mathcal{L}_1' :: \mathcal{L}_2'$, *if* $\mathcal{L}_1 = \mathcal{L}_1'$ *and* $\mathcal{L}_2' = \mathcal{L}_2'$ *then* $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Proof. By the definition of Algorithm 5.10. □

Lemma 5.48 (Free Location Access). *Given memory* σ *and memory block identifier* l, *if* $\mathrm{Free}(\sigma,\ l) = (\sigma_1, (l, 0))$ *then* $(l, 0)$ *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Free. □

Lemma 5.49 (ReadOOB Location Access). *Given index* i, *number of elements* α, *type* ty, *memory* σ *and memory block identifier* l_1, *if* $\mathrm{ReadOOB}(i, \alpha, l_1, ty, \sigma) = (n, 1, (l_2, \mu))$ *then* (l_2, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm ReadOOB. □

Lemma 5.50 (WriteOOB Location Access). *Given index* i, *numbers* n, α, *type* ty, *memory* σ_1, *location map* Δ_1, *and memory block identifier* l_1, *if* $\mathrm{WriteOOB}(n,\ i,\ \alpha,\ l_1,\ ty,\ \sigma_1,\ \Delta_1,\ \mathrm{acc}) = (\sigma_2,\ \Delta_2, j, (l_2, \mu))$ *then* (l_2, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm WriteOOB. □

Lemma 5.51 (Memory Addition Location Access). *Given memory* σ, *memory block identifier* l, *bytes* ω, *number* α, *type* ty *and privacy label* a, *and permission* p, *if* $\sigma_1 = \sigma[l\ \rightarrow\ (\omega,\ ty,\ \alpha,\ \mathrm{PermL}(p, ty, a, \alpha))]$ *then the location* $(l, 0)$ *has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.52 (Memory Modification Location Access). *Given memory* σ, *memory block identifier* l, *bytes* ω, ω', *number* α, *type* ty *and privacy label* a, *and permission* p, *if* $\sigma = \sigma_1[l\ \rightarrow\ (\omega,\ ty,\ \alpha,\ \mathrm{PermL}(p, ty, a, \alpha))]$ *and* $\sigma_2 = \sigma_1[l\ \rightarrow\ (\omega',\ ty,\ \alpha,\ \mathrm{PermL}(p, ty, a, \alpha))]$ *then the location* $(l, 0)$ *has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.53 (InitializeVariables Location Access). *Given variable list* x_{list}, *environment* γ_1, *memory* σ_1, *value* n, *and accumulator* acc, *if* $\mathrm{InitializeVariables}(x_{list},\ \gamma_1,\ \sigma_1,\ n,\ \mathrm{acc}) = (\gamma_2, \sigma_2, L)$ *then the locations* L *have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm InitializeVariables. □

Lemma 5.54 (RestoreVariables Location Access). *Given environment* γ, *memory* σ_1, *variable list* x_{list}, *and accumulator* acc, *if* $\mathrm{RestoreVariables}(x_{list}, \gamma, \sigma_1, \mathrm{acc}) = (\sigma_2, L)$ *then the locations* L *have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm RestoreVariables. □

Lemma 5.55 (ResolveVariables_Retrieve Location Access). *Given environment* γ, *memory* σ, *variable list* x_{list}, *and accumulator* acc, *if* $\mathrm{ResolveVariables\_Retrieve}(x_{list}, \mathrm{acc}, \gamma, \sigma) = ([(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})], n, L)$ *then the locations* L *have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm ResolveVariables_Retrieve. □

Lemma 5.56 (ResolveVariables_Store Location Access). *Given environment γ, memory σ_1, variable list x_{list}, values $[v_1, ..., v_m]$, and accumulator* acc, *if* ResolveVariables_Store$(x_{list}, \text{acc}, \gamma, \sigma_1, [v_1, ..., v_n]) = (\sigma_2, L)$ *then the locations L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm ResolveVariables_Store. □

Lemma 5.57 (Initialize Location Access). *Given location map Δ_1, variable list x_{list}, environment γ_1, memory σ_1, value n, and accumulator* acc, *if* Initialize$(\Delta_1, x_{list}, \gamma_1, \sigma_1, n, \text{acc}) = (\gamma_2, \sigma_2, \Delta_2, L)$ *then the locations L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Initialize. □

Lemma 5.58 (Restore Location Access). *Given memory σ_1, location map Δ_1, and accumulator* acc, *if* Restore$(\sigma_1, \Delta_1, \text{acc}) = (\sigma_2, \Delta_2, L)$ *then the locations L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Restore. □

Lemma 5.59 (Resolve_Retrieve Location Access). *Given environment γ, memory σ, location map Δ, and accumulator* acc, *if* Resolve_Retrieve$(\gamma, \sigma, \Delta, \text{acc}) = ([(v_{t1}, v_{e1}), ..., (v_{tm}, v_{em})], n, L)$ *then the locations L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Resolve_Retrieve. □

Lemma 5.60 (Resolve_Store Location Access). *Given memory σ_1, location map Δ_1, values $[v_1, ..., v_m]$, and accumulator* acc, *if* Resolve_Store$(\Delta_1, \sigma_1, \text{acc}, [v_1, ..., v_m]) = (\sigma_2, \Delta_2, L)$ *then the locations L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Resolve_Store. □

Lemma 5.61 (DynamicUpdate Location Access). *Given memory σ, location map Δ_1, location list L_1, and type $ty \in \{\text{private } a\ bty, \text{private } a\ bty*\}$, if* DynamicUpdate$(\Delta_1, \sigma, L_1, \text{acc}, ty) = (\Delta_2, L_2)$ *then the locations L_2 have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm DynamicUpdate. □

Lemma 5.62 (Pointer Data Location Access). *Given memory σ, memory block identifier l, and $ty \in \{a\ bty*, \text{const } a\ bty*\}$, if $\sigma(l) = (\omega, ty, \alpha, \text{PermL\_Ptr}(\text{Freeable } ty, a, \alpha))$ and* DecodePtr$(ty, \alpha, \omega) = [\alpha, L, J, i]$, *then the location $(l, 0)$ has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.63 (Array Data Location Access). *Given memory σ and memory block identifier l, if $\sigma(l) = (\omega, a\ bty, \alpha, \text{PermL}(\text{Freeable } a\ bty, a, \alpha))$ and* DecodeArr$(a\ bty, i, \omega) = n$, *then the location (l, i) has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.64 (Data Location Access). *Given memory σ and memory block identifier l, if $\sigma(l) = (\omega, a\ bty, 1, \text{PermL}(\text{Freeable } a\ bty, a, 1))$ and* DecodeVal$(a\ bty, \omega) = n$, *then the location $(l, 0)$ has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.65 (Function Data Location Access). *Given memory σ and memory block identifier l, if $\sigma(l) = (\omega, tyL \to ty, 1, \text{PermL\_Fun}(\text{public}))$ and* DecodeFun$(\omega) = (s, n, P)$, *then the location $(l, 0)$ has been accessed.*

Proof. By Definition 5.8. □

Lemma 5.66 (UpdateVal Location Access). *Given memory σ_1, memory block identifier l, value n, and type $a\ bty$, if* UpdateVal$(\sigma_1, l, n, a\ bty) = \sigma_2$, *then the location $(l, 0)$ has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdateVal. □

Lemma 5.67 (UpdateArr Location Access). *Given memory σ_1, memory block identifier l, index i, value n, and type a bty, if* $\mathrm{UpdateArr}(\sigma_1,\ (l, i),\ n,\ a\ bty) = \sigma_2$ *then the location* (l, i) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdateArr. □

Lemma 5.68 (UpdatePtr Location Access). *Given memory σ_1, location (l, μ), pointer data structure $[\alpha,\ L,\ J,\ i]$, and type a $bty*$,*
if $\mathrm{UpdatePtr}(\sigma_1,\ (l, \mu),\ [\alpha,\ L,\ J,\ i],\ a\ bty*) = (\sigma_2,\ j)$ *then the location* (l, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdatePtr. □

Lemma 5.69 (UpdateOffset Location Access). *Given memory σ_1, location (l, μ), number n and type a bty, if* $\mathrm{UpdateOffset}(\sigma_1,\ (l, \mu),\ n,\ a\ bty) = (\sigma_2, j)$ *then the location* (l, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdateOffset. □

Lemma 5.70 (DerefPtr Location Access). *Given memory σ, location (l, μ), and type ty,*
if $\mathrm{DerefPtr}(\sigma, ty, (l, \mu)) = (n, j)$ *then the location* (l, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm DerefPtr. □

Lemma 5.71 (DerefPtrHLI Location Access). *Given memory σ, location (l, μ), and type ty,*
if $\mathrm{DerefPtrHLI}(\sigma,\ ty, (l, \mu)) = ([\alpha, L, J, i], j)$ *then the location* (l, μ) *has been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm DerefPtrHLI. □

Lemma 5.72 (Retrieve_vals Location Access). *Given number α, location list L, type ty, and memory σ, if* $\mathrm{Retrieve\_vals}(\alpha, L, ty, \sigma) = ([v_0, \ldots v_{\alpha'-1}], j)$ *then all locations in L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm Retrieve_vals. □

Lemma 5.73 (CheckFreeable Location Access). *Given environment γ, location list L, tag list J, and memory σ, if* $\mathrm{CheckFreeable}(\gamma, L, J, \sigma) = j$ *then all locations in L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm CheckFreeable. □

Lemma 5.74 (UpdateBytesFree Location Access). *Given location list L, byte representations $[\omega_0, \ldots, \omega_{\alpha-1}]$, and memory σ_1, if* $\mathrm{UpdateBytesFree}(\sigma_1, L, [\omega_0, \ldots, \omega_{\alpha-1}]) = \sigma_2$ *then all locations in L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdateBytesFree. □

Lemma 5.75 (UpdatePointerLocations Location Access). *Given location list L, tag list J, and memory σ_1 if* $\mathrm{UpdatePointerLocations}(\sigma_1, L[1 : \alpha - 1], J[1 : \alpha - 1], L[0], J[0]) = (\sigma_2, L_1)$, *then all locations in L_1 have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdatePointerLocations. □

Lemma 5.76 (UpdateDerefVals Location Access). *Given number α, location list L, list of values $[v_0, \ldots, v_{\alpha-1}]$, type ty, and memory σ_1,*
if $\mathrm{UpdateDerefVals}(\alpha, L, [v_0, \ldots, v_{\alpha-1}], ty, \sigma_1) = \sigma_2$ *then all locations in L have been accessed.*

Proof. By Definition 5.8 and the definition of Algorithm UpdateDerefVals. □

5.4 Multiparty Noninterference Theorem

Theorem 5.1 (Multiparty Noninterference). *For every environment* $\{\gamma^{\mathrm{p}}, \gamma_1^{\mathrm{p}}, \gamma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$*; memory* $\{\sigma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, \sigma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \in \mathrm{Mem}$*; location map*$\{\Delta^{\mathrm{p}}, \Delta_1^{\mathrm{p}}, \Delta_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$*; accumulator* $\{\mathrm{acc}^{\mathrm{p}}, \mathrm{acc}_1^{\mathrm{p}}, \mathrm{acc}_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}} \in \mathbb{N}$*; statement s, values* $\{v^{\mathrm{p}}, v'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$*; step evaluation code lists* $\mathcal{D}, \mathcal{D}'$ *and their corresponding lists of locations accessed* $\mathcal{L}, \mathcal{L}'$*, party* $\mathrm{p} \in \{1 \ldots \mathrm{q}\}$*;*

if $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, s))$
$\quad \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((1, \gamma_1^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}_1^1, v^1) \parallel \ldots \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}_1^{\mathrm{q}}, v^{\mathrm{q}}))$
and $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}^1, s) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}^{\mathrm{q}}, s))$
$\quad \Downarrow_{\mathcal{D}'}^{\mathcal{L}'} ((1, \gamma_1'^1, \sigma_1'^1, \Delta_1'^1, \mathrm{acc}_1'^1, v'^1) \parallel \ldots \parallel (\mathrm{q}, \gamma_1'^{\mathrm{q}}, \sigma_1'^{\mathrm{q}}, \Delta_1'^{\mathrm{q}}, \mathrm{acc}_1'^{\mathrm{q}}, v'^{\mathrm{q}}))$
then $\{\gamma_1^{\mathrm{p}} = \gamma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\sigma_1^{\mathrm{p}} = \sigma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\Delta_1^{\mathrm{p}} = \Delta_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{\mathrm{acc}_1^{\mathrm{p}} = \mathrm{acc}_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\{v^{\mathrm{p}} = v'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, $\mathcal{D} = \mathcal{D}'$, $\mathcal{L} = \mathcal{L}'$, $\Pi \simeq_L \Sigma$.

Proof. *Proof Sketch:* By induction over all SMC^2 semantic rules. We make the assumption that both evaluation traces are over the same program (this is given by having the same *s* in the starting states) and all public data will remain the same, including data read as input during the evaluation of the program. A portion of the complexity of this proof is within ensuring that memory accesses within our semantics remain data oblivious. Several rules follow fairly simply and leverage similar ideas, which we will discuss first, and then we will provide further intuition behind the more complex cases.

For all rules leveraging helper algorithms, we must reason about the helper algorithms, and that they behave deterministically by definition and have data-oblivious memory accesses. Given this and that these helper algorithms do no modify the private data, we maintain the properties of noninterference of this theorem. First we reason that our helper algorithms to translate values into their byte representation will do so deterministically, and therefore maintain indistinguishability between the value and byte representation. We can then reason that our helper algorithms that take these byte values and store them into memory will also do so deterministically, so that when we later access the data in memory we will obtain the same indistinguishable values we had stored.

It is also important to take note here our functions to help us retrieve data from memory, particularly in cases such as when reading out of bounds of an array. When proving these cases to maintain noninterference, we leverage our definition of how memory blocks are assigned in a monotonically increasing fashion, and how the algorithms for choosing which memory block to read into after the current one are deterministic. This, as well as our original assumptions of having identical public input, allows us to reason that if we access out of bounds (including accessing data at a non-aligned position, such as a chunk of bytes in the middle of a memory block), we will be pulling from the same set of bytes each time, and therefore we will end up with the same interpretation of the data as we continue to evaluate the remainder of the program. It is important to note again here that by definition, our semantics will always interpret bytes of data as the type it is expected to be, not the type it actually is (i.e., reading bytes of data that marked private in memory by overshooting a public array will not decrypt the bytes of data, but instead give you back a garbage public value). To reiterate this point, even when reading out of bounds, we will not reveal anything about private data, as the results of these helper algorithms will be indistinguishable.

To reason about the multiparty protocols, we leverage Axioms, such as Axiom 5.7, to reason that the protocols will maintain our definition of noninterference. With each of these Axioms, we ensure that over two different evaluations, if the values of the first run $(v_1^{\mathrm{p}}, v_2^{\mathrm{p}})$ are not distinguishable from those of the second $(v_1'^{\mathrm{p}}, v_2'^{\mathrm{p}})$, then the resulting values are also not distinguishable ($v_3^{\mathrm{p}} = v_3'^{\mathrm{p}}$). These Axioms should be proven by a library developer to ensure the completeness of the formal model.

For private pointers, it is important to note that the obtaining multiple locations is deterministic based upon the program that is being evaluated. A pointer can initially gain multiple locations through the evaluation of a private if else. Once there exists a pointer that has obtained multiple locations in such a way, it can be assigned to another pointer to give that pointer multiple locations. The other case for a pointer to gain multiple location is through the use of `pfree` on a pointer with multiple locations (i.e., the case where a pointer has locations l_1, l_2, l_3 and we free l_1) - when this occurs, if another pointer had referred to only l_1, it will now gain locations in order to mask whether we had to move the true location or not. When reasoning about pointers with multiple locations, we maintain that given the tags for which location is the true location are indistinguishable, then it is not possible to distinguish between them by their usage as defined in the rules or helper algorithms using them. Additionally, to reason about `pfree`, we leverage that the definitions of the helper algorithms are deterministic, and that (wlog), we will be freeing the same location. We will then leverage our Axiom about the multiparty protocol MPC_{free}. After the evaluation of MPC_{free}, it will deterministically update memory and all other pointers as we mentioned in the brief example above.

For both Private If Else rules, the most important element we must leverage is how values are resolved, showing that given our resolution style, we are not able to distinguish between the ending values. In order to do this, we also must reason about the entirety of the rule, including all of if else helper algorithms. First, we note that the evaluation of the `then` branches follows by induction, as does the evaluation of the `else` branch once we have reasoned through the restoration phase. For variable tracking, it is clear from the definitions of Extract, InitializeVariables, and RestoreVariables that the behavior of these algorithms is deterministic and given the same program, we will be extracting, initializing, and restoring the same set variables every time we evaluate the program. For location tracking, Initialize is also immediately clear that it will be initializing the same locations each time. We must then reason about DynamicUpdate, and how given a program, we will deterministically find the pointer dereference writes and array writes at public indices at corresponding positions in memory and add them to our tracking structure Δ. Then we can reason that the behavior of Restore will deterministically perform the same updates, because Δ will contain the same information in every evaluation. Now, we are able to move on to reasoning about resolution, and show that given all of this and the definitions of the resolution helper algorithms and rule, we are not able to distinguish between the ending values.

One of the main complexities of this proof revolves around ensuring *data-oblivious memory accesses* (i.e. that we always access locations deliberately based on public information), particularly when handling arrays and pointers. Within the proof, we must consider all helper algorithms, and what locations are accessed within the algorithms as well as within the rules. What locations are accessed within the algorithms follows deterministically from the definition of the algorithms, and we return from the algorithms which locations were accessed in order to properly reason about the entire evaluation trace of the program. Our semantics are designed in such a way that we give the multiparty protocols all of the information they need, with all memory accesses being completed within the rule itself or our helper algorithms. This also helps show that memory accesses are purely local, not distributed operations. Within the array rules, the main concern is in reading from and writing at a private index. We currently handle this complexity within our rules by accessing all locations within the array in rules Multiparty Array Read Private Index and Multiparty Array Write Private Index. In Multiparty Array Read Private Index, we clearly read data from every index of the array ($\{\forall i \in \{0...\alpha - 1\} \quad \text{DecodeArr}(a\ bty, i, \omega_1^{\text{p}}) = n_i^{\text{p}}\}_{\text{p}=1}^{\text{q}}$), then that data is passed to the multiparty protocol. Similarly, in Multiparty Array Write Private Index, we read data from every index of the array, pass it to the multiparty protocol, then proceed to update every index of the array with what was returned from the protocol. Within the multiparty protocols used in these two rules, we will ensure the usage of the data is data-oblivious within the main noninterference proof in the following subsection. All other array rules use public indices, and in turn only access that publicly known location. Within the pointer rules, our main concern is that we access all locations that are referred to by a private pointer when we have multiple locations. For this, we will reason about the contents of the rules and the helper algorithms used by the pointer rules, which can be shown to deterministically do so.

□

Proof.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, x[e]) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, x[e])) \Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpra])}^{\mathcal{L}_1::\mathcal{L}_2} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n^{\text{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}^1, x[e]) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}^{\text{q}}, x[e])) \Downarrow_{\mathcal{D}_1::(\text{ALL}, [mpra])}^{\mathcal{L}_1::\mathcal{L}_2} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n^{\text{q}}))$, by rule Multiparty Array Read Private Index we have $\{(e) \vdash \gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $\{(n^{\text{p}}) \vdash$

$\gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, (B) $\{\gamma^\text{p}(x) = (l^\text{p}, \text{const } a\ bty*)\}_{\text{p}=1}^{\text{q}}$, (C) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, i^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_1^\text{q}, \Delta_1^\text{q}, \text{acc}, i^\text{q}))$, (D) $\{\sigma_1^\text{p}(l^\text{p}) = (\omega^\text{p}, a \text{ const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, a \text{ const } bty*, a, 1))\}_{\text{p}=1}^{\text{q}}$, (E) $\{\text{DecodePtr}(a \text{ const } bty*, 1, \omega^\text{p}) = [1, [(l_1^\text{p}, 0)], [1], 1]\}_{\text{p}=1}^{\text{q}}$, (F) $\{\sigma_1^\text{p}(l_1^\text{p}) = (\omega_1^\text{p}, a\ bty, \alpha, \text{PermL}(\text{Freeable}, a\ bty, a, \alpha))\}_{\text{p}=1}^{\text{q}}$, (G) $\{\forall i \in \{0...\alpha-1\} \quad \text{DecodeArr}(a\ bty, i, \omega_1^\text{p}) = n_i^\text{p}\}_{\text{p}=1}^{\text{q}}$, (H) $\text{MPC}_{ar}((i^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^\text{q}, [n_0^\text{q}, ..., n_{\alpha-1}^\text{q}])) = (n^1, ..., n^\text{q})$, and (I) $\mathcal{L}_2 = (1, [(l^1, 0), (l_1^1, 0), ..., (l_1^1, \alpha-1)]) \parallel ... \parallel (\text{q}, [(l^\text{q}, 0), (l_1^\text{q}, 0), ..., (l_1^\text{q}, \alpha-1)])$.

Given (J) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}^1, x[e]) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}^\text{q}, x[e])) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\text{ALL},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'} ((1, \gamma_2'^1, \sigma_2'^1, \Delta_2'^1, \text{acc}^1, v'^1) \parallel ... \parallel (\text{q}, \gamma_2'^\text{q}, \sigma_2'^\text{q}, \Delta_2'^\text{q}, \text{acc}^\text{q}, v'^\text{q}))$ and (A), by Lemma 4.87 we have (K) $d = mpra$.

Given (J) and (K), by SMC^2 rule Multiparty Array Read Private Index we have $\{(e) \vdash \gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, $\{(n'^\text{p}) \vdash \gamma^\text{p}\}_{\text{p}=1}^{\text{q}}$, (L) $\{\gamma^\text{p}(x) = (l'^\text{p}, \text{const } a'\ bty'*)\}_{\text{p}=1}^{\text{q}}$, (M) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, e)) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \text{acc}, i'^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_1'^\text{q}, \Delta_1'^\text{q}, \text{acc}, i'^\text{q}))$, (N) $\{\sigma_1'^\text{p}(l'^\text{p}) = (\omega'^\text{p}, a' \text{ const } bty'*, 1, \text{PermL\_Ptr}(\text{Freeable}, a' \text{ const } bty'*, a', 1))\}_{\text{p}=1}^{\text{q}}$, (O) $\{\text{DecodePtr}(a' \text{ const } bty'*, 1, \omega'^\text{p}) = [1, [(l_1'^\text{p}, 0)], [1], 1]\}_{\text{p}=1}^{\text{q}}$, (P) $\{\sigma_1'^\text{p}(l_1'^\text{p}) = (\omega_1'^\text{p}, a'\ bty', \alpha', \text{PermL}(\text{Freeable}, a'\ bty', a', \alpha'))\}_{\text{p}=1}^{\text{q}}$, (Q) $\{\forall i' \in \{0...\alpha'-1\} \quad \text{DecodeArr}(a'\ bty', i', \omega_1'^\text{p}) = n_{i'}'^\text{p}\}_{\text{p}=1}^{\text{q}}$, (R) $\text{MPC}_{ar}((i'^1, [n_0'^1, ..., n_{\alpha'-1}'^1]), ..., (i'^\text{q}, [n_0'^\text{q}, ..., n_{\alpha'-1}'^\text{q}])) = (n'^1, ..., n'^\text{q})$, and (S) $\mathcal{L}_2' = (1, [(l'^1, 0), (l_1'^1, 0), ..., (l_1'^1, \alpha'-1)]) \parallel ... \parallel (\text{q}, [(l'^\text{q}, 0), (l_1'^\text{q}, 0), ..., (l_1'^\text{q}, \alpha'-1)])$.

Given (B) and (L), by Definition 5.3 we have (T) $\{l^\text{p} = l'^\text{p}\}_{\text{p}=1}^{\text{q}}$, (U) $a = a'$, and (V) $bty = bty'$.

Given (C) and (M), by the inductive hypothesis we have (W) $\{\sigma_1^\text{p} = \sigma_1'^\text{p}\}_{\text{p}=1}^{\text{q}}$, (X) $\{\Delta_1^\text{p} = \Delta_1'^\text{p}\}_{\text{p}=1}^{\text{q}}$, (Y) $\{i^\text{p} = i'^\text{p}\}_{\text{p}=1}^{\text{q}}$, (Z) $\mathcal{L}_1 = \mathcal{L}_1'$, and (A1) $\mathcal{D}_1 = \mathcal{D}_1'$.

Given (D), (N), (W), and (T), by Definition 5.4 we have (B1) $\{\omega^\text{p} = \omega'^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Given (E), (O), (U), (V), and (B1), by Lemma 5.26 we have (C1) $\{l_1^\text{p} = l_1'^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Given (F), (P), (W), and (C1), by Definition 5.4 we have (D1) $\{\omega_1^\text{p} = \omega_1'^\text{p}\}_{\text{p}=1}^{\text{q}}$ and (E1) $\alpha = \alpha'$.

Given (G), (Q), and (E1), we have $i = i'$. Given (U), (V), and (D1), by Lemma 5.27 we have (F1) $\forall i \in \{0...\alpha - 1\}\{n_i^\text{p} = n_i'^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Given (H), (R), (Y), and (F1), by Axiom 5.5 we have (G1) $\{n^\text{p} = n'^\text{p}\}_{\text{p}=1}^{\text{q}}$.

Given (D) and (E), by Lemma 5.62 we have accessed locations (H1) $\{(\text{p}, [(l^\text{p}, 0)])\}_{\text{p}=1}^{\text{q}}$. Given (F) and (G), by Lemma 5.63 we have accessed locations (I1) $\{(\text{p}, [(l_1^\text{p}, 0), ..., (l_1^\text{p}, \alpha-1)])\}_{\text{p}=1}^{\text{q}}$. Given (H1) and (I1), by Lemmas 5.44 and 5.46 we have (I).

Given (N) and (O), by Lemma 5.62 we have accessed locations (J1) $\{(\text{p}, [(l^\text{p}, 0)])\}_{\text{p}=1}^{\text{q}}$. Given (P) and (Q), by Lemma 5.63 we have accessed locations (K1) $\{(\mathrm{p}, [(l_1'^1, 0), ..., (l_1'^1, \alpha' - 1)])\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (J1) and (K1), by Lemmas 5.44 and 5.46 we have (S).

Given (T), (C1), (E1), (I), and (S), by Definition 5.10 we have (L1) $\mathcal{L}_2 = \mathcal{L}_2'$. Given (Z) and (L1), by Lemma 5.47 we have (M1) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (A1) and (ALL, $[mpra]$), by Lemma 5.38 we have (N1) $\mathcal{D}_1 ::$ (ALL, $[mpra]$) $= \mathcal{D}_1' ::$ (ALL, $[mpra]$).

Given (W), (X), (G1), (M1), and (N1), by Definition 5.2, we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, x[e_1] = e_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, x[e_1] = e_2)) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{ALL}, [mpwa])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3} ((1, \gamma^1, \sigma_{3+\alpha-1}^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_{3+\alpha-1}^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, x[e_1] = e_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, x[e_1] = e_2)) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{ALL}, [mpwa])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3} ((1, \gamma^1, \sigma_{3+\alpha-1}^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_{3+\alpha-1}^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Multiparty Array Write Private Index, we have (B) $\{(e_1) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e_1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e_1)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, i^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, i^{\mathrm{q}}))$, (D) $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, e_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}\mathrm{acc}, e_2)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$, (E) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private\ const}\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\{\sigma_2^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \mathrm{private\ const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private\ const}\ bty*, \mathrm{private}, 1))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $\{\mathrm{DecodePtr}(\mathrm{private\ const}\ bty*, 1, \omega^{\mathrm{p}}) = [1, [(l_1^{\mathrm{p}}, 0)], [1], 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $\{\sigma_2^{\mathrm{p}}(l_1^{\mathrm{p}}) = (\omega_1^{\mathrm{p}}, \mathrm{private}\ bty, \alpha, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private}\ bty, \mathrm{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (I) $\{\forall j \in \{0...\alpha - 1\}\ \mathrm{DecodeArr}(\mathrm{private}\ bty, j, \omega_1^{\mathrm{p}}) = n_j^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\mathrm{MPC}_{aw}((i^1, n^1, [n_0^1, ..., n_{\alpha-1}^1]), ..., (i^{\mathrm{q}}, n^{\mathrm{q}}, [n_0^{\mathrm{q}}, ..., n_{\alpha-1}^{\mathrm{q}}])) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^{\mathrm{q}}, ..., n_{\alpha-1}'^{\mathrm{q}}])$, (K) $\{\forall j \in \{0...\alpha{-}1\}\ \mathrm{UpdateArr}(\sigma_{2+j}^{\mathrm{p}}, (l_1^{\mathrm{p}}, j), n_j'^{\mathrm{p}}, \mathrm{private}\ bty) = \sigma_{3+j}^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (L) $\mathcal{L}_3 = (1, [(l^{\mathrm{p}}, 0), (l_1^{\mathrm{p}}, 0), ..., (l_1^{\mathrm{p}}, \alpha{-}1)]) \parallel ... \parallel (\mathrm{q}, [(l^{\mathrm{p}}, 0), (l_1^{\mathrm{p}}, 0), ..., (l_1^{\mathrm{p}}, \alpha - 1)])$.

Given (M) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, x[e_1] = e_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, x[e_1] = e_2)) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{ALL}, [d])}^{\mathcal{L}_1'::\mathcal{L}_2'::\mathcal{L}_3'} ((1, \gamma^1, \sigma_{3+\alpha-1}'^1, \Delta_2'^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_{3+\alpha-1}'^{\mathrm{q}}, \Delta_2'^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ and (A), by Lemma 4.87 we have (N) $d = mpwa$.

Given (M) and (N), by SMC^2 rule Multiparty Array Write Private Index, we have (O) $\{(e_1) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (P) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e_1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e_1)) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \mathrm{acc}, i'^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1'^{\mathrm{q}}, \Delta_1'^{\mathrm{q}}, \mathrm{acc}, i'^{\mathrm{q}}))$, (Q) $((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \mathrm{acc}, e_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1'^{\mathrm{q}}, \Delta_1'^{\mathrm{q}}\mathrm{acc}, e_2)) \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((1, \gamma^1, \sigma_2'^1, \Delta_2'^1, \mathrm{acc}, n'^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2'^{\mathrm{q}}, \Delta_2'^{\mathrm{q}}, \mathrm{acc}, n'^{\mathrm{q}}))$, (R) $\{\gamma^{\mathrm{p}}(x) = (l'^{\mathrm{p}}, \mathrm{private\ const}\ bty'*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (S) $\{\sigma_2'^{\mathrm{p}}(l'^{\mathrm{p}}) = (\omega'^{\mathrm{p}}, \mathrm{private\ const}\ bty'*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private\ const}\ bty'*, \mathrm{private}, 1))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (T) $\{\mathrm{DecodePtr}(\mathrm{private\ const}\ bty'*, 1, \omega'^{\mathrm{p}}) = [1, [(l_1'^{\mathrm{p}}, 0)], [1], 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (U) $\{\sigma_2'^{\mathrm{p}}(l_1'^{\mathrm{p}}) = (\omega_1'^{\mathrm{p}}, \mathrm{private}\ bty', \alpha', \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private}\ bty', \mathrm{private}, \alpha'))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (V) $\{\forall j' \in \{0...\alpha' - 1\}\ \mathrm{DecodeArr}(\mathrm{private}\ bty', j', \omega_1'^{\mathrm{p}}) = n_j''^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (W) $\mathrm{MPC}_{aw}((i'^1, n'^1, [n_0''^1, ..., n_{\alpha'-1}''^1]), ..., (i'^{\mathrm{q}}, n'^{\mathrm{q}}, [n_0''^{\mathrm{q}}, ..., n_{\alpha'-1}''^{\mathrm{q}}])) = ([n_0'''^1, ..., n_{\alpha'-1}'''^1], ..., [n_0'''^{\mathrm{q}}, ..., n_{\alpha'-1}'''^{\mathrm{q}}])$, (X) $\{\forall j' \in \{0...\alpha' - 1\}\ \mathrm{UpdateArr}(\sigma_{2+j'}'^{\mathrm{p}},$

$(l_1^{\prime\text{p}}, j'), n_{j'}^{\prime\prime\text{p}}, \text{private } bty') = \sigma_{3+j'}^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, and (Y) $\mathcal{L}_3' = (1, [(l^{\prime\text{p}}, 0), (l_1^{\prime\text{p}}, 0), ..., (l_1^{\prime\text{p}}, \alpha' - 1)]) \parallel ... \parallel (\text{q}, [(l^{\prime\text{p}}, 0), (l_1^{\prime\text{p}}, 0), ..., (l_1^{\prime\text{p}}, \alpha' - 1)])$.

Given (C) and (P), by the inductive hypothesis we have (Z) $\{\sigma_1^{\text{p}} = \sigma_1^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (A1) $\{\Delta_1^{\text{p}} = \Delta_1^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (B1) $\{i^{\text{p}} = i^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (C1) $\mathcal{L}_1 = \mathcal{L}_1'$, and (D1) $\mathcal{D}_1 = \mathcal{D}_1'$.

Given (D), (Q), (Z), and (A1), by the inductive hypothesis we have (E1) $\{\sigma_2^{\text{p}} = \sigma_2^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (F1) $\{\Delta_2^{\text{p}} = \Delta_2^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (G1) $\{n^{\text{p}} = n^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$, (H1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (I1) $\mathcal{D}_2 = \mathcal{D}_2'$.

Given (E) and (R), by Definition 5.3 we have (J1) $\{l^{\text{p}} = l^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$ and (K1) $bty = bty'$.

Given (F), (S), (E1) and (J1), by Definition 5.4 we have (L1) $\{\omega^{\text{p}} = \omega^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (G), (T), (K1), and (L1), by Lemma 5.26 we have (M1) $\{l_1^{\text{p}} = l_1^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (H), (U), (E1), and (M1), by Definition 5.4 we have (N1) $\{\omega_1^{\text{p}} = \omega_1^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$ and (O1) $\alpha = \alpha'$.

Given (I), (V), (O1), we have (P1) $j = j'$. Given (I), (V), (K1), (O1), (P1), and (N1), by Lemma 5.27 we have (Q1) $\forall j \in \{0...\alpha - 1\}\{n_j^{\text{p}} = n_j^{\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (J), (W), (B1), (G1), and (Q1), by Axiom 5.6 we have (R1) $\{n^{\prime\text{p}} = n^{\prime\prime\prime\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (K), (X), (P1), (O1), (M1), (E1), (L1), and (R1), by Lemma 5.35 we have (S1) $\forall j, j' \in \{0...\alpha - 1\}$ such that $j = j'$, $\sigma_{2+j} = \sigma_{2+j'}'$ and (T1) $\sigma_{3+j} = \sigma_{3+j'}'$.

Given (F) and (G), by Lemma 5.62 we have accessed locations (U1) $\{(\text{p}, [(l^{\text{p}}, 0)])\}_{\text{p}=1}^{\text{q}}$. Given (H) and (I), by Lemma 5.63 we have accessed locations (V1) $\{(\text{p}, [(l_1^{\text{p}}, 0), ..., (l_1^{\text{p}}, \alpha - 1)])\}_{\text{p}=1}^{\text{q}}$. Given (K), by Lemma 5.67 we have accessed locations (W1) $\{(\text{p}, [(l_1^{\text{p}}, 0), ..., (l_1^{\text{p}}, \alpha - 1)])\}_{\text{p}=1}^{\text{q}}$. Given (U1), (V1), and (W1), by Lemmas 5.44 and 5.46 we have (L).

Given (S) and (T), by Lemma 5.62 we have accessed locations (X1) $\{(\text{p}, [(l^{\prime\text{p}}, 0)])\}_{\text{p}=1}^{\text{q}}$. Given (U) and (V), by Lemma 5.63 we have accessed locations (Y1) $\{(\text{p}, [(l_1^{\prime\text{p}}, 0), ..., (l_1^{\prime\text{p}}, \alpha' - 1)])\}_{\text{p}=1}^{\text{q}}$. Given (X), by Lemma 5.67 we have accessed locations (Z1) $\{(\text{p}, [(l_1^{\prime\text{p}}, 0), ..., (l_1^{\prime\text{p}}, \alpha' - 1)])\}_{\text{p}=1}^{\text{q}}$. Given (X1), (Y1), and (Z1), by Lemmas 5.44 and 5.46 we have (Y).

Given (J1), (M1), (O1), (L), and (Y), by Definition 5.10 we have (A2) $\mathcal{L}_3 = \mathcal{L}_3'$. Given (C1), (H1), and (A2), by Lemma 5.47 we have (B2) $\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3 = \mathcal{L}_1' :: \mathcal{L}_2' :: \mathcal{L}_3'$.

Given (D1), (I1), and (ALL, $[mpwa]$), by Lemma 5.38 we have (C2) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpwa]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{ALL}, [mpwa])$.

Given (T1), (F1), (C2), and (B2), by Definition 5.2, we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1 \ bop \ e_2) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1 \ bop \ e_2)) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpb])}^{\mathcal{L}_1 :: \mathcal{L}_2}$
$((1, \gamma_2^1, \sigma_2^1, \Delta_2^1, \text{acc}, n_3^1) \parallel ... \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}, n_3^{\text{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1 \ bop \ e_2) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1 \ bop \ e_2)) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpb])}^{\mathcal{L}_1 :: \mathcal{L}_2}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, n_3^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}, n_3^{\text{q}}))$, by SMC[2] rule Multiparty Binary Operation we have

$\{(e_1, e_2) \vdash \gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $bop \in \{\cdot, +, -, \div\}$, (B) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n_1^1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n_1^{\text{q}}))$, (C) $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, e_2) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, e_2)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, n_2^1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}, n_2^{\text{q}}))$, and (D) $\text{MPC}_b(bop, [n_1^1, \ldots, n_1^{\text{q}}], [n_2^1, \ldots, n_2^{\text{q}}]) = (n_3^1, \ldots, n_3^{\text{q}})$.

Given (E) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1\ bop\ e_2) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1\ bop\ e_2)) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{ALL}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2'} ((1, \gamma^1, \sigma_2'^1, \Delta_2'^1, \text{acc}, n_3'^1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_2'^{\text{q}}, \Delta_2'^{\text{q}}, \text{acc}, n_3'^{\text{q}}))$ and (A), by Lemma 4.87 we have (F) $d = mpb$.

Given (E) and (F), by SMC^2 rule Multiparty Binary Operation we have $\{(e_1, e_2) \vdash \gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, $bop \in \{\cdot, +, -, \div\}$, (G) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1)) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \text{acc}, n_1'^1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1'^{\text{q}}, \Delta_1'^{\text{q}}, \text{acc}, n_1'^{\text{q}}))$,
(H) $((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \text{acc}, e_2) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1'^{\text{q}}, \Delta_1'^{\text{q}}, \text{acc}, e_2)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((1, \gamma^1, \sigma_2'^1, \Delta_2'^1, \text{acc}, n_2'^1) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma_2'^{\text{q}}, \Delta_2'^{\text{q}}, \text{acc}, n_2'^{\text{q}}))$, and (I) $\text{MPC}_b(bop, [n_1'^1, \ldots, n_1'^{\text{q}}], [n_2'^1, \ldots, n_2'^{\text{q}}]) = (n_3'^1, \ldots, n_3'^{\text{q}})$.

Given (B) and (G), by the inductive hypothesis we have (J) $\{\sigma_1^{\text{p}} = \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (K) $\{\Delta_1^{\text{p}} = \Delta_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (L) $\{n_1^{\text{p}} = n_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$.

Given (C), (H), (J), and (K), by the inductive hypothesis we have (O) $\{\sigma_2^{\text{p}} = \sigma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (P) $\{\Delta_2^{\text{p}} = \Delta_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (Q) $\{n_2^{\text{p}} = n_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (R) $\mathcal{D}_2 = \mathcal{D}_2'$, (S) $\mathcal{L}_2 = \mathcal{L}_2'$.

Given (D), (I), (L), and (Q), by Axiom 5.7 we have (T) $\{n_3^{\text{p}} = n_3'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (M), (R), and $(\text{ALL}, [mpb])$, by Lemma 5.38 we have (U) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpb]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{ALL}, [mpb])$.

Given (N) and (S), by Lemma 5.47 we have (V) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (O), (P), (T), (U), and (V), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}^1, e_1\ bop\ e_2) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}^{\text{q}}, e_1\ bop\ e_2)) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpcmp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((1, \gamma_2^1, \sigma_2^1, \Delta_2^1, \text{acc}^1, n_3^1) \parallel \ldots \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}^{\text{q}}, n_3^{\text{q}}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e_1\ bop\ e_2) \parallel \ldots \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e_1\ bop\ e_2)) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{ALL}, [mpb])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((1, \gamma_2^1, \sigma_2^1, \Delta_2^1, \text{acc}, n_3^1) \parallel \ldots \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc}, n_3^{\text{q}}))$. The main difference is using Axiom 5.8 in place of Axiom 5.7.

Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ n_3) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ n_3) \parallel C_2)$ by SMC^2 rule Public Addition, we have $(e_1, e_2) \nvdash \gamma$, (B) $((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \text{acc}, \ n_1) \parallel C_1)$, (C) $((\text{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \text{acc}, \ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ n_2) \parallel C_2)$, and (D) $n_1 + n_2 = n_3$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2'} ((\text{p}, \gamma, \ \sigma_2', \ \Delta_2', \ \text{acc}, \ n_3') \parallel C_2')$ and (A), by Lemma 4.87 we have (F) $d = bp$.

Given (E) and (F), by SMC^2 rule Public Addition, we have $(e_1, e_2) \nvdash \gamma$, (G) $((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e_1) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'}$

$((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ n_1') \parallel C_1')$, (H) $((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ e_2') \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\text{p}, \gamma,\ \sigma_2',\ \Delta_2',\ \text{acc},\ n_2') \parallel C_2')$, and (I) $n_1' + n_2' = n_3'$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $n_1 = n_1'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C), (H), (J), (K), and (O), by the inductive hypothesis we have (P) $\sigma_2 = \sigma_2'$, (Q) $\Delta_2 = \Delta_2'$, (R) $n_2 = n_2'$, (S) $\mathcal{D}_2 = \mathcal{D}_2'$, (T) $\mathcal{L}_2 = \mathcal{L}_2'$, and (U) $C_2 = C_2'$.

Given (D), (I), (L), and (R), we have (V) $n_3 = n_3'$.

Given (M), (S), and $(\text{p}, [bp])$, by Lemma 5.38 we have (W) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [bp])$.

Given (N) and (T), by Lemma 5.47 we have (X) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (P), (Q), (U), (V), (W), and (X), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 \cdot e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bm])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 - e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bs])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 \div e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bd])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ n_3) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1 + e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [bp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, n_3) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ltt])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ltt])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$ by SMC^2 rule Public Less Than True, we have $(e_1, e_2) \nvDash \gamma$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)$, (C) $((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ n_2) \parallel C_2)$, and (D) $(n_1 < n_2) = 1$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2'} ((\text{p}, \gamma,\ \sigma_2',\ \Delta_2',\ \text{acc},\ 1) \parallel C_2')$ and (A), by Lemma 4.87 we have (F) $d = ltt$.

Given (E) and (F), by SMC^2 rule Public Less Than True we have $(e_1, e_2) \nvDash \gamma$, (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1) \parallel C)$

$\Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'}$ ((p, γ, σ_1', Δ_1', acc, n_1') $\|$ C_1'), (H) ((p, γ, σ_1', Δ_1', acc, e_2) $\|$ C_1') $\Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'}$ ((p, γ, σ_2', Δ_2, acc, n_2') $\|$ C_2'), and (I) $(n_1' < n_2') = 1$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $n_1 = n_1'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C), (H), (J), (K), and (O), by the inductive hypothesis we have (P) $\sigma_2 = \sigma_2'$, (Q) $\Delta_2 = \Delta_2'$, (R) $n_2 = n_2'$, (S) $\mathcal{D}_2 = \mathcal{D}_2'$, (T) $\mathcal{L}_2 = \mathcal{L}_2'$, and (U) $C_2 = C_2'$.

Given (D), (I), (L), and (R), we have (V) $(n_1 < n_2) = (n_1' < n_2') = 1$.

Given (M), (S), and (p, $[ltt]$), by Lemma 5.38 we have (W) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ltt]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [ltt])$.

Given (N) and (T), by Lemma 5.47 we have (X) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (P), (Q), (U), (V), (W), and (X), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[eqt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 == e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[eqf])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 ! = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[net])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 ! = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[nef])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 0) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e_1 < e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[ltt])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ 1) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x) \parallel C) \Downarrow_{(\text{p},[dv])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x) \parallel C) \Downarrow_{(\text{p},[dv])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ by SMC^2 rule Public Declaration, we have $(ty = \text{public } bty)$, acc $= 0$ (B) $l = \phi()$, (C) $\gamma_1 = \gamma[x\ \rightarrow\ (l,\ ty)]$, (D) $\omega = \text{EncodeVal}(ty, \text{NULL})$, and (E) $\sigma_1 = \sigma[l\ \rightarrow\ (\omega,\ ty,\ 1,\ \text{PermL}(\text{Freeable}, ty, \text{public}, 1))]$.

Given (F) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x) \parallel C) \Downarrow_{(\text{p},[dv])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1',\ \sigma_1',\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ and (A), by Lemma 4.87 we have (G) $d = dv$.

Given (F) and (G), by SMC^2 rule Public Declaration, we have $(ty = \text{public } bty)$, acc $= 0$ (H) $l' = \phi()$, (I)

$\gamma_1' = \gamma[x \to (l', ty)]$, (J) $\omega' = \text{EncodeVal}(ty, \text{NULL})$, and (K) $\sigma_1' = \sigma[l' \to (\omega', ty, 1, \text{PermL}(\text{Freeable}, ty, \text{public}, 1))]$.

Given (B) and (H), by Axiom 5.4 we have (L) $l = l'$.

Given (C), (I), and (L), by Definition 5.3 we have (M) $\gamma_1 = \gamma_1'$.

Given (D) and (J), by Lemma 5.30 we have (N) $\omega = \omega'$.

Given (E), (K), (L), and (N), by Definition 5.4 we have (O) $\sigma_1 = \sigma_1'$.

Given (E), by Lemma 5.51 we have accessed (P) $(\text{p}, [(l, 0)])$. Given (K), by Lemma 5.51 we have accessed (Q) $(\text{p}, [(l', 0)])$. Given (P), (Q), and (L), we have (R) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (A), (F), and (G) we have (S) $(\text{p}, [dv]) = (\text{p}, [dv])$.

Given (M), (O), (R), and (S), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow_{(\text{p}, [d1])}^{(\text{p}, [(l, 0)])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow_{(\text{p}, [dv])}^{(\text{p}, [(l, 0)])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1; s_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, v_2) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1; s_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, v_2) \parallel C_2)$ by SMC^2 rule Statement Sequencing, we have (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, v_1) \parallel C_1)$, and (C) $((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, s_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, v_2) \parallel C_2)$.

Given (D) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1; s_2) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [ss])}^{\mathcal{L}_1' :: \mathcal{L}_2'} ((\text{p}, \gamma_2', \sigma_2', \Delta_2', \text{acc}, v_2') \parallel C_2')$ and (A), by Lemma 4.87 we have (E) $d = ss$.

Given (D) and (E) by SMC^2 rule Statement Sequencing, we have (F) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma_1', \sigma_1', \Delta_1', \text{acc}, v_1') \parallel C_1')$, and (G) $((\text{p}, \gamma_1', \sigma_1', \Delta_1', \text{acc}, s_2) \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\text{p}, \gamma_2', \sigma_2', \Delta_2', \text{acc}, v_2') \parallel C_2')$.

Given (B) and (F), by the inductive hypothesis we have (H) $\gamma_1 = \gamma_1'$, (I) $\sigma_1 = \sigma_1'$, (J) $\Delta_1 = \Delta_1'$, (K) $v_1 = v_1'$, (L) $\mathcal{D}_1 = \mathcal{D}_1'$, (M) $\mathcal{L}_1 = \mathcal{L}_1'$, and (N) $C_1 = C_1'$.

Given (C), (G), (H), (I), (J), and (N), by the inductive hypothesis we have (O) $\gamma_2 = \gamma_2'$, (P) $\sigma_2 = \sigma_2'$, (Q) $\Delta_2 = \Delta_2'$, (R) $v_2 = v_2'$, (S) $\mathcal{D}_2 = \mathcal{D}_2'$, (T) $\mathcal{L}_2 = \mathcal{L}_2'$, and (U) $C_2 = C_2'$.

Given (L), (S), and $(\text{p}, [ss])$, by Lemma 5.38 we have (V) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [ss]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [ss])$.

Given (M) and (T), by Lemma 5.47 we have (W) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (O), (P), (Q), (R), (U), (V), and (W), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \{s\}) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[sb])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \{s\}) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[sb])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$ by SMC^2 rule Statement Block, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ v) \parallel C_1)$.

Given (C) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \{s\}) \parallel C) \Downarrow_{\mathcal{D}_1'::(\mathrm{p},[d])}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (D) $d = sb$.

Given (C) and (D), by SMC^2 rule Statement Block, we have (E) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ s) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma_1',\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ v') \parallel C_1')$.

Given (B) and (E), by the inductive hypothesis we have (F) $\gamma_1 = \gamma_1'$, (G) $\sigma_1 = \sigma_1'$, (H) $\Delta_1 = \Delta_1'$, (I) $v_1 = v_1'$, (J) $\mathcal{D}_1 = \mathcal{D}_1'$, (K) $\mathcal{L}_1 = \mathcal{L}_1'$, and (L) $C_1 = C_1'$.

Given (J) and $(\mathrm{p}, [sb])$, by Lemma 5.38 we have (M) $\mathcal{D}_1 :: (\mathrm{p}, [sb]) = \mathcal{D}_1' :: (\mathrm{p}, [sb])$.

Given (G), (H), (L), (K), and (M), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[ep])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ v) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[ep])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ v) \parallel C_1)$ by SMC^2 rule Parentheses, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ v) \parallel C_1)$.

Given (C) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (e)) \parallel C) \Downarrow_{\mathcal{D}_1'::(\mathrm{p},[d])}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ v') \parallel C_1')$ and (A), by Lemma 4.87 we have (D) $d = ep$.

Given (C) and (D), by SMC^2 rule Parentheses, we have (E) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ v') \parallel C_1')$.

Given (B) and (E), by the inductive hypothesis we have (F) $\sigma_1 = \sigma_1'$, (G) $\Delta_1 = \Delta_1'$, (H) $v = v'$, (I) $\mathcal{D}_1 = \mathcal{D}_1'$, (J) $\mathcal{L}_1 = \mathcal{L}_1'$, and (K) $C_1 = C_1'$.

Given (I) and $(\mathrm{p}, [ep])$, by Lemma 5.38 we have (L) $\mathcal{D}_1 :: (\mathrm{p}, [ep]) = \mathcal{D}_1' :: (\mathrm{p}, [ep])$.

Given (F), (G), (H), (J), (K), and (L), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ds])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ds])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$ by SMC^2 rule

Declaration Assignment, we have (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$, and (C) $((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ x = e) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Given (D) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{p},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'} ((\mathrm{p}, \gamma_1',\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (E) $d = ds$.

Given (D) and (E) by SMC^2 rule Declaration Assignment, we have (F) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma_1',\ \sigma_1',\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1')$, and (G) $((\mathrm{p}, \gamma_1',\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ x = e) \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\mathrm{p}, \gamma_1',\ \sigma_2',\ \Delta_2',\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2')$.

Given (B) and (F), by the inductive hypothesis we have (H) $\gamma_1 = \gamma_1'$, (I) $\sigma_1 = \sigma_1'$, (J) $\Delta_1 = \Delta_1'$, (K) $\mathcal{D}_1 = \mathcal{D}_1'$, (L) $\mathcal{L}_1 = \mathcal{L}_1'$, and (M) $C_1 = C_1'$.

Given (C), (G), (H), (I), (J), and (N), by the inductive hypothesis we have (N) $\sigma_2 = \sigma_2'$, (O) $\Delta_2 = \Delta_2'$, (P) $\mathcal{D}_2 = \mathcal{D}_2'$, (Q) $\mathcal{L}_2 = \mathcal{L}_2'$, and (R) $C_2 = C_2'$.

Given (K), (P), and $(\mathrm{p}, [ds])$, by Lemma 5.38 we have (S) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [ds]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\mathrm{p}, [ds])$.

Given (L) and (Q), by Lemma 5.47 we have (T) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (H), (N), (O), (S), and (T), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[das])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_1,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ ty\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ds])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma_1,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[r])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ v) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow_{(\mathrm{p},[r])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ v) \parallel C)$ by SMC^2 rule Read Public Variable, we have (B) $\gamma(x) = (l,\ \text{public}\ bty)$, (C) $\sigma(l) = (\omega,\ \text{public}\ bty,\ 1,\ \text{PermL}(\text{Freeable}, \text{public}\ bty, \text{public}, 1))$, and (D) $\text{DecodeVal}(\text{public}\ bty,\ \omega) = v$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ v') \parallel C)$ and (A), by Lemma 4.87 we have (F) $d = r$.

Given (E) and (F), by SMC^2 rule Read Public Variable, we have (G) $\gamma(x) = (l',\ \text{public}\ bty')$, (H) $\sigma(l') = (\omega',\ \text{public}\ bty',\ 1,\ \text{PermL}(\text{Freeable}, \text{public}\ bty', \text{public}, 1))$, and (I) $\text{DecodeVal}(\text{public}\ bty',\ \omega') = v'$.

Given (B) and (G), by Definition 5.3 we have (J) $l = l'$, and (K) $bty = bty'$.

Given (C), (H), and (J), by Definition 5.4 we have (L) $\omega = \omega'$.

Given (D), (I), (K), and (L), by Lemma 5.29 we have (M) $v = v'$.

Given (C) and (D), by Lemma 5.64 we have accessed location $(\text{p}, [(l, 0)])$. Given (H) and (I), by Lemma 5.64 we have accessed location $(\text{p}, [(l', 0)])$. Given (J), we have (N) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (A), (E), (F), (M), and (N), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[r1])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ v) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[r])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ v) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Write Public Variable, we have $(e) \nvdash \gamma$, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{public}\ bty)$, and (D) $\text{UpdateVal}(\sigma_1,\ l,\ n,\ \text{public}\ bty) = \sigma_2$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma_2',\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (F) $d = w$.

Given (E) and (F), by SMC^2 rule Write Public Variable, we have $(e) \nvdash \gamma$, (G) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, n') \parallel C_1')$, (H) $\gamma(x) = (l',\ \text{public}\ bty')$, and (I) $\text{UpdateVal}(\sigma_1',\ l',\ n',\ \text{public}\ bty') = \sigma_2'$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $n = n'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C) and (H), by Definition 5.3 we have (P) $l = l'$ and (Q) $bty = bty'$.

Given (D), (I), (J), (P), (L), and (Q), by Lemma 5.34 we have (R) $\sigma_2 = \sigma_2'$.

Given (M) and $(\text{p}, [w])$, by Lemma 5.38 we have (S) $\mathcal{D}_1 :: (\text{p}, [w]) = \mathcal{D}_1' :: (\text{p}, [w])$.

Given (D), by Lemma 5.66 we have accessed location $(\text{p}, [(l, 0)])$. Given (I), by Lemma 5.66 we have accessed location (p, $[(l', 0)]$). Given (P), we have (T) (p, $[(l, 0)]$) = (p, $[(l', 0)]$). Given (N) and (T), by Lemma 5.47 we have (U) $\mathcal{L}_1$ :: (p, $[(l, 0)]$) = $\mathcal{L}_1'$ :: (p, $[(l', 0)]$).

Given (R), (K), (O), (S), and (U), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w2])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[w2])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Write Private Variable Public Value, we have $(e) \nvdash \gamma$, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{public}\ bty)$, and (D) UpdateVal$(\sigma_1,\ l,\ \text{encrypt}(n),\ \text{public}\ bty) = \sigma_2$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma_2',\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (F) $d = w2$.

Given (E) and (F), by SMC^2 rule Write Private Variable Public Value, we have $(e) \nvdash \gamma$, (G) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, n') \parallel C_1')$, (H) $\gamma(x) = (l',\ \text{public}\ bty')$, and (I) UpdateVal$(\sigma_1',\ l',\ \text{encrypt}(n'),\ \text{public}\ bty') = \sigma_2'$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $n = n'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C) and (H), by Definition 5.3 we have (P) $l = l'$ and (Q) $bty = bty'$.

Given (L) and encrypt(n) and encrypt(n'), by Axiom 5.1 we have (R) encrypt(n) = encrypt(n').

Given (D), (I), (J), (P), (R), and (Q), by Lemma 5.34 we have (S) $\sigma_2 = \sigma_2'$.

Given (M) and (p, $[w]$), by Lemma 5.38 we have (T) $\mathcal{D}_1$ :: (p, $[w]$) = $\mathcal{D}_1'$ :: (p, $[w]$).

Given (D), by Lemma 5.66 we have accessed location (p, $[(l, 0)]$). Given (I), by Lemma 5.66 we have accessed location (p, $[(l', 0)]$). Given (P), we have (U) (p, $[(l, 0)]$) = (p, $[(l', 0)]$). Given (N) and (U), by Lemma 5.47 we have (V) $\mathcal{L}_1$ :: (p, $[(l, 0)]$) = $\mathcal{L}_1'$ :: (p, $[(l', 0)]$).

Given (S), (K), (O), (T), and (V), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{smcinput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[inp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{smcinput}(x,\ e)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[inp])}^{\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$ by SMC^2 rule SMC Input Public Value, we have $(e) \nvdash \gamma$, acc $= 0$, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc},$

n) $\parallel C_1$), (C) $\gamma(x) = (l, \text{public } bty)$, (D) $\text{InputValue}(x, n) = n_1$, (E) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, x = n_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$.

Given (F) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e)) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2'} ((\text{p}, \gamma, \sigma_2', \Delta_2', \text{acc}, \text{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (G) $d = inp$.

Given (F) and (G), by SMC^2 rule SMC Input Public Value, we have $(e) \nvDash \gamma$, acc = 0, (H) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, n') \parallel C_1')$, (I) $\gamma(x) = (l', \text{public } bty')$, (J) $\text{InputValue}(x, n') = n_1'$, (K) $((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, x = n_1') \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\text{p}, \gamma, \sigma_2', \Delta_2', \text{acc}, \text{skip}) \parallel C_2')$.

Given (B) and (H), by the inductive hypothesis we have (L) $\sigma_1 = \sigma_1'$, (M) $\Delta_1 = \Delta_1'$, (N) $n = n'$, (O) $\mathcal{D}_1 = \mathcal{D}_1'$, (P) $\mathcal{L}_1 = \mathcal{L}_1'$, and (Q) $C_1 = C_1'$.

Given (C) and (I), by Definition 5.3 we have (R) $l = l'$, and (S) $bty = bty'$.

Given (D), (J), and (N), by Axiom 5.2 we have (T) $n_1 = n_1'$.

Given (E), (K), (L), (M), (Q), and (T), by the inductive hypothesis we have (U) $\sigma_2 = \sigma_2'$, (V) $\Delta_2 = \Delta_2'$, (W) $\mathcal{D}_2 = \mathcal{D}_2'$, (X) $\mathcal{L}_2 = \mathcal{L}_2'$, and (Y) $C_2 = C_2'$.

Given (O), (W), and (p, $[inp]$), by Lemma 5.38 we have (Z) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [inp]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [inp])$.

Given (P) and (X), by Lemma 5.47 we have (A1) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (U), (V), (Y), (Z), and (A1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [inp2])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [inp])}^{\mathcal{L}_1 :: \mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_1)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [inp3])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_1)) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [inp3])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)$ by SMC^2 rule SMC Input Private Array, we have $(e_1, e_2) \nvDash \gamma$, acc = 0, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (C) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \alpha) \parallel C_2)$, (D) $\gamma(x) = (l, \text{private const } bty*)$, (E) $\text{InputArray}(x, n, \alpha) = [m_0, ..., m_\alpha]$, and (F) $((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, x = [m_0, ..., m_\alpha]) \parallel C_2) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_3} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_1)) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: \mathcal{D}_3' :: (\text{p}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2' :: \mathcal{L}_3'} ((\text{p}, \gamma, \sigma_3', \Delta_3', \text{acc}, \text{skip}) \parallel C_3')$ and (A), by Lemma 4.87 we have (H) $d = inp3$.

Given (G) and (H), by SMC^2 rule SMC Input Private Array, we have $(e_1, e_2) \nvDash \gamma$, acc = 0, (I) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, n') \parallel C_1')$, (J) $((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, e_2) \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\text{p}, \gamma, \sigma_2', \Delta_2', \text{acc}, \alpha') \parallel C_2')$, (K) $\gamma(x) =$

$(l', \text{private const } bty'*)$, (L) $\text{InputArray}(x, n', \alpha') = [m'_0, ..., m'_{\alpha'}]$, and (M) $((\text{p}, \gamma, \sigma'_2, \Delta'_2, \text{acc}, x = [m'_0, ..., m'_{\alpha'}]) \parallel C'_2) \Downarrow^{\mathcal{L}'_3}_{\mathcal{D}'_3} ((\text{p}, \gamma, \sigma'_3, \Delta'_3, \text{acc}, \text{skip}) \parallel C'_3)$.

Given (B) and (I), by the inductive hypothesis we have (N) $\sigma_1 = \sigma'_1$, (O) $\Delta_1 = \Delta'_1$, (P) $n = n'$, (Q) $\mathcal{D}_1 = \mathcal{D}'_1$, (R) $\mathcal{L}_1 = \mathcal{L}'_1$, and (S) $C_1 = C'_1$.

Given (C), (J), (N), (O), and (S), by the inductive hypothesis we have (T) $\sigma_2 = \sigma'_2$, (U) $\Delta_2 = \Delta'_2$, (V) $\alpha = \alpha'$, (W) $\mathcal{D}_2 = \mathcal{D}'_2$, (X) $\mathcal{L}_2 = \mathcal{L}'_2$, and (Y) $C_2 = C'_2$.

Given (D) and (K), by Definition 5.3 we have (Z) $l = l'$, and (A1) $bty = bty'$.

Given (E), (L), (P), and (V), by Axiom 5.3 we have (B1) $[m_0, ..., m_{n_1}] = [m'_0, ..., m'_{n'_1}]$.

Given (F), (M), (T), (U), (Y), and (B1), by the inductive hypothesis we have (C1) $\sigma_3 = \sigma'_3$, (D1) $\Delta_3 = \Delta'_3$, (E1) $\mathcal{D}_3 = \mathcal{D}'_3$, (F1) $\mathcal{L}_3 = \mathcal{L}'_3$, and (G1) $C_3 = C'_3$.

Given (O), (W), (E1) and $(\text{p}, [inp3])$, by Lemma 5.38 we have (H1) $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [inp3]) = \mathcal{D}'_1 :: \mathcal{D}'_2 :: \mathcal{D}'_3 :: (\text{p}, [inp3])$.

Given (R), (X), and (F1), by Lemma 5.47 we have (I1) $\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3 = \mathcal{L}'_1 :: \mathcal{L}'_2 :: \mathcal{L}'_3$.

Given (C1), (D1), (G1), (H1), and (I1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_2)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\text{p},[inp1])} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcinput}(x, e_1, e_1)) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3}_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\text{p},[inp3])} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_3)$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcoutput}(x, e)) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[out])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcoutput}(x, e)) \parallel C) \Downarrow^{\mathcal{L}_1::(\text{p},[(l,0)])}_{\mathcal{D}_1::(\text{p},[out])} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)$ by SMC^2 rule SMC Output Public Value, we have $(e) \nvdash \gamma$, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, n) \parallel C_1)$, (C) $\gamma(x) = (l, \text{public } bty)$, (D) $\sigma_1(l) = (\omega, \text{public } bty, 1, \text{PermL}(\text{Freeable}, \text{public } bty, \text{public}, 1))$, (E) $\text{DecodeVal}(\text{public } bty, \omega) = n_1$, and (F) $\text{OutputValue}(x, n, n_1)$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, \text{smcoutput}(x, e)) \parallel C) \Downarrow^{\mathcal{L}'_1::(\text{p},[(l',0)])}_{\mathcal{D}'_1::(\text{p},[d])} ((\text{p}, \gamma, \sigma'_1, \Delta'_1, \text{acc}, \text{skip}) \parallel C'_1)$ and (A), by Lemma 4.87 we have (H) $d = out$.

Given (G) and (H), by SMC^2 rule SMC Output Public Value, we have $(e) \nvdash \gamma$, (I) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e) \parallel C)$

$\Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'}$ ((p, γ, σ_1', Δ_1', acc, n') $\|$ C_1'), (J) $\gamma(x) = (l', \text{public } bty')$, (K) $\sigma_1'(l') = (\omega', \text{ public } bty', 1, \text{PermL(Freeable, public } bty', \text{public}, 1))$, (L) DecodeVal(public bty', ω') = n_1', and (M) OutputValue(x, n', n_1').

Given (B) and (I), by the inductive hypothesis we have (N) $\sigma_1 = \sigma_1'$, (O) $\Delta_1 = \Delta_1'$, (P) $n = n'$, (Q) $\mathcal{D}_1 = \mathcal{D}_1'$, (R) $\mathcal{L}_1 = \mathcal{L}_1'$, and (S) $C_1 = C_1'$.

Given (C) and (J), by Definition 5.3 we have (T) $l = l'$, and (U) $bty = bty'$.

Given (D), (K), (N), and (T), by Definition 5.4 we have (V) $\omega = \omega'$.

Given (E), (L), (U), and (V), by Lemma 5.29 we have (W) $n_1 = n_1'$.

Given (F), (M), (P), and (W), by Lemma 5.1 we have identical output going to the same parties.

Given (Q) and (p, $[out]$), by Lemma 5.38 we have (X) $\mathcal{D}_1 :: (\text{p}, [out]) = \mathcal{D}_1' :: (\text{p}, [out])$.

Given (D) and (E), by Lemma 5.64 we have accessed location (p, $[(l, 0)]$). Given (K) and (L), by Lemma 5.64 we have accessed location (p, $[(l', 0)]$). Given (R) and (T), by Lemma 5.47 we have (Y) $\mathcal{L}_1 :: (\text{p}, [(l, 0)]) = \mathcal{L}_1' :: (\text{p}, [(l', 0)])$.

Given (N), (O), (S), (X), and (Y), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, smcoutput(x, e)) $\|$ C) $\Downarrow_{\mathcal{D}_1::(\text{p},[out2])}^{\mathcal{L}_1::(\text{p},[(l,0)])}$ ((p, γ, σ_1, Δ_1, acc, skip) $\|$ C_1)

This case is similar to Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, smcoutput(x, e)) $\|$ C) $\Downarrow_{\mathcal{D}_1::(\text{p},[out])}^{\mathcal{L}_1::(\text{p},[(l,0)])}$ ((p, γ, σ_1, Δ_1, acc, skip) $\|$ C_1).

Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, smcoutput(x, e_1, e_2)) $\|$ C) $\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[out3])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_1,0),\dots,(l_1,\alpha-1)])}$ ((p, γ, σ_2, Δ_2, acc, skip) $\|$ C_2)

Given (A) $\Pi \triangleright$ ((p, γ, σ, Δ, acc, smcoutput(x, e_1, e_2)) $\|$ C) $\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[out3])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_1,0),\dots,(l_1,\alpha-1)])}$ ((p, γ, σ_2, Δ_2, acc, skip) $\|$ C_2) by SMC^2 rule SMC Output Private Array, we have $(e_1, e_2) \nvdash \gamma$, (B) ((p, γ, σ, Δ, acc, e_1) $\|$ C) $\Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}$ ((p, γ, σ_1, Δ_1, acc, n) $\|$ C_1), (C) ((p, γ, σ_1, Δ_1, acc, e_2) $\|$ C_1) $\Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2}$ ((p, γ, σ_2, Δ_2, acc, α) $\|$ C_2), (D) $\gamma(x) = (l, \text{private const } bty*)$, (E) $\sigma_2(l) = (\omega, \text{private const } bty*, 1, \text{PermL\_Ptr(Freeable, private const } bty*, \text{private}, 1))$, (F) DecodePtr(private const $bty*$, 1, ω) = $[1, [(l_1, 0)], [1], \text{private } bty, 1]$, (G) $\sigma_2(l_1) = (\omega_1, \text{private } bty, \alpha, \text{PermL(Freeable, private } bty, \text{private}, \alpha))$, (H) $\forall i \in \{0, ..., \alpha - 1\}$ DecodeArr(private bty, i, ω_1) = m_i, and (I) OutputArray(x, n, $[m_0, ..., m_{\alpha-1}]$).

Given (J) $\Sigma \triangleright$ ((p, γ, σ, Δ, acc, smcoutput(x, e_1, e_2)) $\|$ C) $\Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\text{p},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'::(\text{p},[(l',0),(l_1',0),\dots,(l_1',\alpha-1)])}$ ((p, γ, σ_2', Δ_2', acc, skip) $\|$ C_2') and (A), by Lemma 4.87 we have (K) $d = out3$.

Given (J) and (K), by SMC^2 rule SMC Output Private Array, we have $(e_1, e_2) \nvdash \gamma$, (L) ((p, γ, σ, Δ, acc, e_1) $\|$ C) $\Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'}$ ((p, γ, σ_1', Δ_1', acc, n') $\|$ C_1'), (M) ((p, γ, σ_1', Δ_1', acc, e_2) $\|$ C_1') $\Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'}$ ((p, γ, σ_2', Δ_2', acc, α') $\|$ C_2'), (N) $\gamma(x) = (l', \text{private const } bty'*)$, (O) $\sigma_2'(l') = (\omega', \text{ private const } bty'*, 1, \text{PermL\_Ptr(Freeable, private const } bty'*, \text{private}, 1))$, (P) DecodePtr(private const $bty'*$, 1, ω') = $[1, [(l_1', 0)], [1], \text{private } bty', 1]$, (Q) $\sigma_2'(l_1')$ =

$(\omega_1', \text{private } bty', \alpha', \text{PermL}(\text{Freeable}, \text{private } bty', \text{private}, \alpha'))$, (R) $\forall i' \in \{0, ..., \alpha'-1\}$ DecodeArr(private bty', i', ω_1') $= m_{i'}'$, and (S) OutputArray(x, n', $[m_0', ..., m_{\alpha'-1}']$).

Given (B) and (L), by the inductive hypothesis we have (T) $\sigma_1 = \sigma_1'$, (U) $\Delta_1 = \Delta_1'$, (V) $n = n'$, (W) $\mathcal{D}_1 = \mathcal{D}_1'$, (X) $\mathcal{L}_1 = \mathcal{L}_1'$, and (Y) $C_1 = C_1'$.

Given (C) and (M), by the inductive hypothesis we have (Z) $\sigma_2 = \sigma_2'$, (A1) $\Delta_2 = \Delta_2'$, (B1) $\alpha = \alpha'$, (C1) $\mathcal{D}_2 = \mathcal{D}_2'$, (D1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (E1) $C_2 = C_2'$.

Given (D) and (N), by Definition 5.3 we have (F1) $l = l'$, and (G1) $bty = bty'$.

Given (E), (O), (Z), and (F1), by Definition 5.4 we have (H1) $\omega = \omega'$.

Given (F), (P), (G1), and (H1), by Lemma 5.26 we have (I1) $l_1 = l_1'$.

Given (G), (Q), (Z), and (I1), by Definition 5.4 we have (J1) $\omega_1 = \omega_1'$ and (K1) $\alpha = \alpha'$.

Given (R), (H), (K1), we have $i = i'$. Given (G1) and (J1), by Lemma 5.27 we have (L1) $\forall i \in \{0...\alpha - 1\} m_i = m_i'$.

Given (I), (S), (V), (K1), and (L1), by Lemma 5.2 we have identical output going to the same parties.

Given (W), (C1) and (p, $[out3]$), by Lemma 5.38 we have (M1) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [out3]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [out3])$.

Given (E) and (F), by Lemma 5.62 we have accessed locations (N1) $(\text{p}, [(l, 0)])$. Given (G) and (H), by Lemma 5.63 we have accessed locations (O1) $(\text{p}, [(l_1, 0), ..., (l_1, \alpha - 1)])$. Given (N1) and (O1), by Lemmas 5.44 and 5.45 we have (P1) $(\text{p}, [(l, 0), (l_1, 0), ..., (l_1, \alpha - 1)])$ Given (O) and (P), by Lemma 5.62 we have accessed locations (Q1) $(\text{p}, [(l', 0)])$. Given (Q) and (R), by Lemma 5.63 we have accessed locations (R1) $(\text{p}, [(l_1', 0), ..., (l_1', \alpha' - 1)])$. Given (Q1) and (R1), by Lemmas 5.44 and 5.45 we have (S1) $(\text{p}, [(l', 0), (l_1', 0), ..., (l_1', \alpha' - 1)])$.

Given (X), (D1), (F1), (I1), (P1), and (S1) by Lemma 5.47 we have (T1) $\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_1, 0), ..., (l_1, \alpha-1)]) = \mathcal{L}_1' :: \mathcal{L}_2' :: (\text{p}, [(l', 0), (l_1', 0), ..., (l_1', \alpha' - 1)])$.

Given (Z), (A1), (E1), (M1), and (T1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{smcoutput}(x,\ e_1,\ e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [out1])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p}, [(l,0), (l_1,0), ..., (l_1, \alpha-1)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{smcoutput}(x,\ e_1,\ e_2)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p}, [out3])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p}, [(l,0), (l_1,0), ..., (l_1, \alpha-1)])}$ $((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},\ \text{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p}, [fpd])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p}, [fpd])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ by SMC^2 rule Function Definition, we have acc $= 0$, $x \notin \gamma$, (B) $l = \phi()$, (C) GetFunTypeList$(P) = tyL$, (D) $\gamma_1 = \gamma[x \to (l,\ tyL \to ty)]$,

(E) CheckPublicEffects$(s,\ x,\ \gamma, \sigma) = n$, (F) EncodeFun$(s,\ n,\ P) = \omega$, and (G) $\sigma_1 = \sigma[l \to (\omega,\ tyL \to ty,\ 1,\ \text{PermL\_Fun(public)})]$.

Given (H) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma_1',\ \sigma_1',\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ and (A), by Lemma 4.87 we have (I) $d = fpd$.

Given (H) and (I), by SMC[2] rule Function Definition, we have $\text{acc} = 0, x \notin \gamma$, (J) $l' = \phi()$, (K) GetFunTypeList$(P) = tyL'$, (L) $\gamma_1' = \gamma[x \to (l',\ tyL' \to ty)]$, (M) CheckPublicEffects$(s,\ x,\ \gamma, \sigma) = n'$, (N) EncodeFun$(s,\ n',\ P) = \omega'$, and (O) $\sigma_1' = \sigma[l' \to (\omega',\ tyL' \to ty,\ 1,\ \text{PermL\_Fun(public)})]$.

Given (B) and (J), by Axiom 5.4 we have (P) $l = l'$.

Given (C) and (K), by Lemma 5.3 we have (Q) $tyL = tyL'$.

Given (D), (L), (P), and (Q), by Definition 5.3 we have (R) $\gamma_1 = \gamma_1'$.

Given (E) and (M), by Lemma 5.5 we have (S) $n = n'$.

Given (F), (N), and (S), by Lemma 5.33 we have (T) $\omega = \omega'$.

Given (G), (O), (P), (Q), and (T), by Definition 5.4 we have (U) $\sigma_1 = \sigma_1'$.

Given (G), by Lemma 5.51 we have accessed (V) $(\text{p}, [(l, 0)])$. Given (O), by Lemma 5.51 we have accessed (W) $(\text{p}, [(l', 0)])$. Given (V), (W), and (P), we have (X) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (A), (H), and (I) we have (Y) $(\text{p}, [fpd]) = (\text{p}, [fpd])$.

Given (R), (U), (X), and (Y), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)) \parallel C) \Downarrow_{(\text{p},[df])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fpd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[fd])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ by SMC[2] rule Pre-Declared Function Definition, we have $\text{acc} = 0, x \in \gamma$, (B) $\gamma(x) = (l,\ tyL \to ty)$, (C) CheckPublicEffects$(s,\ x,\ \gamma, \sigma) = n$, (D) $\sigma = \sigma_1[l \to (\text{NULL},\ tyL \to ty,\ 1,\ \text{PermL\_Fun(public)})]$, (E) EncodeFun$(s,\ n,\ P) = \omega$, and (F) $\sigma_2 = \sigma_1[l \to (\omega,\ tyL \to ty,\ 1,\ \text{PermL\_Fun(public)})]$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x(P)\{s\}) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma_2',\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = fd$.

Given (G) and (H), by SMC[2] rule Pre-Declared Function Definition, we have $\text{acc} = 0$, $x \in \gamma$, (I) $\gamma(x) = (l',\ tyL' \to ty)$, (J) CheckPublicEffects$(s,\ x,\ \gamma, \sigma) = n'$,

(K) $\sigma = \sigma_1'[l' \rightarrow (\text{NULL}, tyL' \rightarrow ty, 1, \text{PermL\_Fun(public)})]$, (L) $\text{EncodeFun}(s, n', P) = \omega'$, and (M) $\sigma_2' = \sigma_1'[l' \rightarrow (\omega', tyL' \rightarrow ty, 1, \text{PermL\_Fun(public)})]$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$ and (O) $tyL = tyL'$.

Given (C) and (J), by Lemma 5.5 we have (P) $n = n'$.

Given (D), (K), (N), and (O), by Definition 5.4 we have (Q) $\sigma_1 = \sigma_1'$.

Given (E), (L), and (P), by Lemma 5.33 we have (R) $\omega = \omega'$.

Given (F), (M), (N), (O), (Q), and (R), by Definition 5.4 we have (S) $\sigma_2 = \sigma_2'$.

Given (D) and (F), by Lemma 5.52 we have accessed (T) $(\text{p}, [(l, 0)])$. Given (K) and (M), by Lemma 5.52 we have accessed (U) $(\text{p}, [(l', 0)])$. Given (T), (U), and (N), we have (V) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (A), (G), and (H) we have (W) $(\text{p}, [fd]) = (\text{p}, [fd])$.

Given (S), (V), and (W), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc1])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[fc1])}^{(\text{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$ by SMC^2 rule Function Call Without Public Side Effects, we have (B) $\gamma(x) = (l, tyL \rightarrow ty)$, (C) $\sigma(l) = (\omega, tyL \rightarrow ty, 1, \text{PermL\_Fun(public)})$, (D) $\text{DecodeFun}(\omega) = (s, n, P)$, (E) $\text{GetFunParamAssign}(P, E) = s_1$, (F) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, s_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, \text{skip}) \parallel C_1)$, (G) $n = 0$, and (H) $((\text{p}, \gamma_1, \sigma_1, \Delta_1, \text{acc}, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma_2, \sigma_2, \Delta_2, \text{acc}, \text{skip}) \parallel C_2)$.

Given (I) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x(E)) \parallel C) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\text{p},[d])}^{(\text{p},[(l',0)])::\mathcal{L}_1'::\mathcal{L}_2'} ((\text{p}, \gamma, \sigma_2', \Delta_2', \text{acc}, \text{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (J) $d = fc1$.

Given (I) and (J), by SMC^2 rule Function Call Without Public Side Effects, we have (K) $\gamma(x) = (l', tyL' \rightarrow ty')$, (L) $\sigma(l') = (\omega', tyL' \rightarrow ty', 1, \text{PermL\_Fun(public)})$, (M) $\text{DecodeFun}(\omega') = (s', n', P')$,

(N) $\text{GetFunParamAssign}(P', E) = s_1'$, (O) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ s_1') \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma_1',\ \sigma_1',\ \Delta_1',\ \text{acc},\ \text{skip}) \parallel C_1')$, (P) $n' = 0$, and (Q) $((\text{p}, \gamma_1',\ \sigma_1',\ \Delta_1',\ \text{acc},\ s) \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\text{p}, \gamma_2',\ \sigma_2',\ \Delta_2',\ \text{acc},\ \text{skip}) \parallel C_2')$.

Given (B) and (K), by Definition 5.3 we have (R) $l = l'$, (S) $tyL = tyL'$, and (T) $ty = ty'$.

Given (C), (L), and (R), by Definition 5.4 we have (U) $\omega = \omega'$.

Given (D), (M), and (U), by Lemma 5.28 we have (V) $s = s'$, (W) $n = n'$, and (X) $P = P'$.

Given (E), (N), and (X), by Lemma 5.4 we have (Y) $s_1 = s_1'$.

Given (F), (O), and (Y), by the inductive hypothesis we have (Z) $\gamma_1 = \gamma_1'$, (A1) $\sigma_1 = \sigma_1'$, (B1) $\Delta_1 = \Delta_1'$, (C1) $\mathcal{D}_1 = \mathcal{D}_1'$, (D1) $\mathcal{L}_1 = \mathcal{L}_1'$, and (E1) $C_1 = C_1'$.

Given (H), (Q), (Z), (A1), (B1), (E1), by the inductive hypothesis we have (F1) $\gamma_2 = \gamma_2'$, (G1) $\sigma_2 = \sigma_2'$, (H1) $\Delta_2 = \Delta_2'$, (I1) $\mathcal{D}_2 = \mathcal{D}_2'$, (J1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (K1) $C_2 = C_2'$.

Given (C1), (I1), and $(\text{p}, [fc1])$, by Lemma 5.38 we have (L1) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [fc1]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [fc1])$.

Given (C) and (D), by Lemma 5.65 we have accessed (M1) $(\text{p}, [(l, 0)])$. Given (L) and (M), by Lemma 5.65 we have accessed (N1) $(\mathrm{p}, [(l', 0)])$. Given (M1), (N1), and (R), we have (O1) $(\mathrm{p}, [(l, 0)]) = (\mathrm{p}, [(l', 0)])$. Given (D1), (J1), and (O1), by Lemma 5.47 we have (P1) $\mathcal{L}_1 :: \mathcal{L}_2 :: (\mathrm{p}, [(l, 0)]) = \mathcal{L}_1' :: \mathcal{L}_2' :: (\mathrm{p}, [(l', 0)])$.

Given (G1), (H1), (K1), (L1), and (P1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[fc])}^{(\mathrm{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x(E)) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[fc1])}^{(\mathrm{p},[(l,0)])::\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \Downarrow_{(\mathrm{p},[ty])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \Downarrow_{(\mathrm{p},[ty])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$ by SMC^2 rule Size of Type, we have (B) $n = \tau(ty)$ and $(ty) \nvdash \gamma$.

Given (C) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{sizeof}(ty)) \parallel C) \Downarrow_{(\mathrm{p},[d])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n') \parallel C)$ and (A), by Lemma 4.87 we have (D) $d = ty$.

Given (C) and (D), by SMC^2 rule Size of Type, we have (E) $n' = \tau(ty)$ and $(ty) \nvdash \gamma$.

Given (B) and (E), by Lemma 5.6 we have (F) $n = n'$.

Given (A), (C), and (D), we have (G) $(\mathrm{p}, [ty]) = (\mathrm{p}, [ty])$.

Given (F) and (G), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \Downarrow_{(\mathrm{p},[loc])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l, 0)) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \Downarrow_{(\mathrm{p},[loc])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l, 0)) \parallel C)$ by SMC^2 rule Address Of, we have (B) $\gamma(x) = (l,\ ty)$.

Given (C) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \&x) \parallel C) \Downarrow_{(\mathrm{p},[d])}^{\epsilon} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l', 0)) \parallel C)$ and (A), by Lemma 4.87 we have (D) $d = loc$.

Given (C) and (D), by SMC^2 rule Address Of, we have (E) $\gamma(x) = (l',\ ty')$.

Given (B) and (E), by Definition 5.3 we have (F) $l = l'$ and $ty = ty'$.

Given (A), (C), and (D), we have (G) $(\mathrm{p}, [loc]) = (\mathrm{p}, [loc])$.

Given (F) and (G), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[cv])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[cv])}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_1) \parallel C_1)$ by SMC^2 rule Cast Public

Value, we have $(e) \nvdash \gamma$, $(ty = \text{public } bty)$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$, and (C) $n_1 = \text{Cast}(\text{public},\ ty,\ n)$.

Given (D) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ n_1') \parallel C_1')$ and (A), by Lemma 4.87 we have (E) $d = cv$.

Given (D) and (E), by SMC^2 rule Cast Public Value, we have $(e) \nvdash \gamma$, $(ty = \text{public } bty)$, (F) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ n') \parallel C_1')$, and (G) $n_1' = \text{Cast}(\text{public},\ ty,\ n')$.

Given (B) and (F), by the inductive hypothesis we have (H) $\sigma_1 = \sigma_1'$, (I) $\Delta_1 = \Delta_1'$, (J) $n = n'$, (K) $\mathcal{D}_1 = \mathcal{D}_1'$, (L) $\mathcal{L}_1 = \mathcal{L}_1'$, and (M) $C_1 = C_1'$.

Given (C), (G), and (J), by Lemma 5.7 we have (N) $n_1 = n_1'$.

Given (K) and $(\text{p}, [cv])$, by Lemma 5.38 we have (O) $\mathcal{D}_1 :: (\text{p}, [cv]) = \mathcal{D}_1' :: (\text{p}, [cv])$.

Given (H), (I), (N), (M), (L), and (O), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cv1])}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cv])}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_1) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$ by SMC^2 rule Cast Private Location, we have $(ty = \text{private } bty*)$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$, (C) $\sigma_1 = \sigma_2\big[l \rightarrow \big(\omega,\ \text{void},\ n, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, n)\big)\big]$, and (D) $\sigma_3 = \sigma_2\big[l \rightarrow \big(\omega,\ ty,\ \frac{n}{\tau(ty)}, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, \frac{n}{\tau(ty)})\big)\big]$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma_3',\ \Delta_1',\ \text{acc},\ (l', 0)) \parallel C_1')$ and (A), by Lemma 4.87 we have (F) $d = cl1$.

Given (E) and (F), by SMC^2 rule Cast Private Location, we have $(ty = \text{private } bty*)$, (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ (l', 0)) \parallel C_1')$, (H) $\sigma_1' = \sigma_2'\big[l' \rightarrow \big(\omega',\ \text{void},\ n', \text{PermL\_Ptr}(\text{Freeable}, \text{void}, \text{private}, n')\big)\big]$, and (I) $\sigma_3' = \sigma_2'\big[l' \rightarrow \big(\omega',\ ty,\ \frac{n'}{\tau(ty)}, \text{PermL\_Ptr}(\text{Freeable}, ty, \text{private}, \frac{n'}{\tau(ty)})\big)\big]$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $l = l'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C), (H), (J), and (L), by Definition 5.4 we have (P) $\sigma_2 = \sigma_2'$, (Q) $\omega = \omega'$, and (R) $n = n'$.

Given (D), (I), (P), (L), (Q), and (R), by Definition 5.4 we have (S) $\sigma_3 = \sigma_3'$.

Given (C) and (D), by Lemma 5.52 we have accessed (T) $(\text{p}, [(l, 0)])$. Given (H) and (I), by Lemma 5.52 we have accessed (U) (p, $[(l', 0)]$). Given (T), (U), and (L), we have (V) (p, $[(l, 0)]$) = (p, $[(l', 0)]$). Given (N) and (V), by Lemma 5.47 we have (W) $\mathcal{L}_1 ::$ (p, $[(l, 0)]$) = $\mathcal{L}_1' ::$ (p, $[(l', 0)]$).

Given (A), (E), (F), and (M), we have (X) $\mathcal{D}_1 ::$ (p, $[cl1]$) = $\mathcal{D}_1' ::$ (p, $[cl1]$).

Given (S), (K), (L), (O), (W), and (X), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (ty)\ e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[cl1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{free}(e)) \parallel C) \Downarrow_{(\text{p},[fre])}^{(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{free}(e)) \parallel C) \Downarrow_{(\text{p},[fre])}^{(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ by SMC^2 rule Single Location Free, we have acc = 0, (B) $\gamma(x) = (l,\ \text{public}\ bty*)$, (C) $\sigma(l) = (\omega, \text{public}\ bty*, 1, \text{PermL}(\text{Freeable}, \text{public}\ bty*, \text{public}, 1))$, (D) $\text{DecodePtr}(\text{public}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (E) $\text{CheckFreeable}(\gamma, [(l_1, 0)], [1], \sigma) = 1$, and (F) $\text{Free}(\sigma,\ l_1) = (\sigma_1, (l_1, 0))$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{free}(e)) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0),(l_1',0)])} ((\text{p}, \gamma,\ \sigma_1',\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = fre$.

Given (G) and (H), by SMC^2 rule Single Location Free, we have acc = 0, (I) $\gamma(x) = (l',\ \text{public}\ bty'*)$, (J) $\sigma(l') = (\omega', \text{public}\ bty'*, 1, \text{PermL}(\text{Freeable}, \text{public}\ bty'*, \text{public}, 1))$, (K) $\text{DecodePtr}(\text{public}\ bty'*,\ 1,\ \omega') = [1,\ [(l_1', 0)],\ [1],\ 1]$, (L) $\text{CheckFreeable}(\gamma, [(l_1', 0)], [1], \sigma) = 1$, and (M) $\text{Free}(\sigma,\ l_1') = (\sigma_1', (l_1', 0))$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$ and (O) $bty = bty'$.

Given (C), (J), and (N), by Definition 5.4 we have (P) $\omega = \omega'$.

Given (D), (K), (O), (P), by Lemma 5.26 we have (Q) $l_1 = l_1'$.

Given (F), (M), and (Q), by Lemma 5.8 we have (R) $\sigma_1 = \sigma_1'$.

Given (C) and (D), by Lemma 5.62 we have accessed (S) (p, $[(l, 0)]$). Given (F), by Lemma 5.48 we have accessed location (T) (p, $[(l_1, 0)]$ Given (J) and (K), by Lemma 5.62 we have accessed (U) (p, $[(l', 0)]$). Given (M), by Lemma 5.48 we have accessed location (V) (p, $[(l_1', 0)]$

Given (S), (T), (U), and (V), by Lemmas 5.44 and 5.45 we have (W) (p, $[(l, 0), (l_1, 0)]$) and (X) (p, $[(l', 0), (l_1', 0)]$). Given (W), (X), (N), and (Q) by Definition 5.10 we have (Y) (p, $[(l, 0), (l_1, 0)]) = ($p, $[(l', 0), (l_1', 0)]$).

Given (A), (G), and (H), we have (Z) (p, $[fre]) = ($p, $[fre]$).

Given (R), (Y), and (Z), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{pfree}(x)) \parallel C) \Downarrow_{(\text{p},[pfre])}^{(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{free}(e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[fre])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])]} ((\text{p}, \gamma,\ \sigma_2,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[malp])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[malp])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ (l, 0)) \parallel C_1)$ by SMC^2 rule Private Malloc, we have $(e) \nvdash \gamma$, acc $= 0$, $(ty = \text{private } bty*) \vee (ty = \text{private } bty)$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \parallel C_1)$, (C) $l = \phi()$, and (D) $\sigma_2 = \sigma_1\big[l \to \big(\text{NULL},\ \text{void}*,\ n \cdot \tau(ty), \text{PermL}(\text{Freeable}, \text{void}*, \text{private}, n \cdot \tau(ty))\big)\big]$.

Given (E) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ \text{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma_2',\ \Delta_1',\ \text{acc},\ (l', 0)) \parallel C_1')$ by Lemma 4.87 we have (F) $d = malp$.

Given (E) and (F), by SMC^2 rule Private Malloc, we have $(e) \nvdash \gamma$, acc $= 0$, $(ty = \text{private } bty*) \vee (ty = \text{private } bty)$, (G) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1,\ \text{acc},\ n') \parallel C_1')$, (H) $l' = \phi()$, and (I) $\sigma_2' = \sigma_1'\big[l' \to \big(\text{NULL},\ \text{void}*,\ n' \cdot \tau(ty), \text{PermL}(\text{Freeable}, \text{void}*, \text{private}, n' \cdot \tau(ty))\big)\big]$.

Given (B) and (G), by the inductive hypothesis we have (J) $\sigma_1 = \sigma_1'$, (K) $\Delta_1 = \Delta_1'$, (L) $n = n'$, (M) $\mathcal{D}_1 = \mathcal{D}_1'$, (N) $\mathcal{L}_1 = \mathcal{L}_1'$, and (O) $C_1 = C_1'$.

Given (C) and (H), by Axiom 5.4 we have (P) $l = l'$.

Given (D), (I), (J), (P), and (L), by Definition 5.4 we have (Q) $\sigma_2 = \sigma_2'$.

Given (D), by Lemma 5.51 we have accessed location (R) (p, $[(l, 0)]$). Given (I), by Lemma 5.51 we have accessed location (S) (p, $[(l', 0)]$). Given (N), (P), (R), and (S), by Lemma 5.47 we have (T) $\mathcal{L}_1 ::$ (p, $[(l, 0)]) = \mathcal{L}'_1 ::$ (p, $[(l', 0)]$).

Given (O) and (p, $[malp]$), by Lemma 5.38 we have (U) $\mathcal{D}_1 ::$ (p, $[malp]) = \mathcal{D}'_1 ::$ (p, $[malp]$).

Given (Q), (K), (P), (T), (U) and (O), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{malloc}(e)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[mal])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{pmalloc}(e,\ ty)) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[malp])}^{\mathcal{L}_1::(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \mathrm{acc},\ (l, 0)) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, {++}\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin3])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, n_2) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, {++}\ x) \parallel C) \Downarrow_{(\mathrm{p},[pin3])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma, \sigma_1, \Delta, \mathrm{acc}, n_2) \parallel C)$, by SMC^2 rule Pre-Increment Private Int Variable, we have (B) $\gamma(x) = (l, \mathrm{private\ int})$, (C) $\sigma(l) = (\omega, \mathrm{private\ int}, 1, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private\ int}, \mathrm{private}, 1))$, (D) $\mathrm{DecodeVal}(\mathrm{private\ int}, \omega) = n_1$, (E) $n_2 = n_1 + \mathrm{encrypt}(1)$, and (F) $\mathrm{UpdateVal}(\sigma, l, n_2, \mathrm{private\ int}) = \sigma_1$.

Given (G) $\Sigma \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, {++}\ x) \parallel C) \Downarrow_{(\mathrm{p},[d])}^{(\mathrm{p},[(l',0)])} ((\mathrm{p}, \gamma, \sigma'_1, \Delta, \mathrm{acc}, v'_2) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = pin3$.

Given (G) and (H), by SMC^2 rule Pre-Increment Private Int Variable, we have (I) $\gamma(x) = (l', \mathrm{private\ int})$, (J) $\sigma(l') = (\omega', \mathrm{private\ int}, 1, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{private\ int}, \mathrm{private}, 1))$, (K) $\mathrm{DecodeVal}(\mathrm{private\ int}, \omega') = n'_1$, (L) $n'_2 = n'_1 + \mathrm{encrypt}(1)$, and (M) $\mathrm{UpdateVal}(\sigma, l', n'_2, \mathrm{private\ int}) = \sigma'_1$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$.

Given (C), (J), and (N), by Definition 5.4 we have (O) $\omega = \omega'$.

Given (D), (K), and (O), by Lemma 5.29 we have (P) $n_1 = n'_1$.

By Axiom 5.1, we have (Q) $\mathrm{encrypt}(1) = \mathrm{encrypt}(1)$. Given (E), (L), (P), and (Q), we have (R) $n_2 = n'_2$.

Given (F), (M), (N), and (R), by Lemma 5.34 we have (S) $\sigma_1 = \sigma'_1$.

Given (A), (G), and (H), we have (T) (p, $[pin3]$) = (p, $[pin3]$).

Given (C) and (D), by Lemma 5.64 and Lemma 5.66 we have accessed location (U) (p, $[(l, 0)]$). Given (J) and (K), by Lemma 5.64 and Lemma 5.66 we have accessed location (V) (p, $[(l', 0)]$). Given (U), (V), and (N), we have (W) (p, $[(l, 0)]$) = (p, $[(l', 0)]$).

Given (S), (R), (T), and (W) by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p}, [pin])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ n_1) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, {++}\ x) \parallel C) \Downarrow_{(\text{p}, [pin3])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma, \sigma_1, \Delta, \text{acc}, n_2) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p}, [pin5])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ [\alpha,\ L_1,\ J,\ i]) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p}, [pin5])}^{(\text{p}, [(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ [\alpha,\ L_1,\ J,\ i]) \parallel C)$ by SMC^2 rule Pre-Increment Private Pointer Multiple Locations, we have (B) $\gamma(x) = (l,\ \text{private}\ bty*)$, (C) $\sigma(l) = (\omega, \text{private}\ bty*, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, \alpha))$, (D) $\text{DecodePtr}(\text{private}\ bty*,\ \alpha,\ \omega) = [\alpha, L, J, i]$, (E) $\text{IncrementList}(L,\ \tau(\text{private}\ bty*), \sigma) = (L_1, 1)$, and (F) $\text{UpdatePtr}(\sigma,\ (l, 0),\ [\alpha,\ L_1,\ J,\ i],\ \text{private}\ bty*) = (\sigma_1,\ 1)$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {++}\ x) \parallel C) \Downarrow_{(\text{p}, [d])}^{(\text{p}, [(l',0)])} ((\text{p}, \gamma,\ \sigma_1',\ \Delta,\ \text{acc},\ [\alpha',\ L_1',\ J',\ i']) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = pin5$.

Given (G) and (H), by SMC^2 rule Pre-Increment Private Pointer Multiple Locations, we have (I) $\gamma(x) = (l',\ \text{private}\ bty'*)$, (J) $\sigma(l') = (\omega',\ \text{private}\ bty'*,\ \alpha',\ \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty'*, \text{private}, \alpha'))$, (K)

DecodePtr(private $bty'*,\ \alpha',\ \omega'$) = $[\alpha',\ L',\ J',\ i']$, (L) IncrementList($L',\ \tau$(private $bty'*$)$,\sigma$) = $(L'_1, 1)$, and (M) UpdatePtr($\sigma,\ (l', 0),\ [\alpha',\ L'_1,\ J',\ i']$, private $bty'*$) = $(\sigma'_1,\ 1)$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$ and (O) $bty = bty'$.

Given (C), (J), and (N), by Definition 5.4 we have (P) $\omega = \omega'$ and (Q) $\alpha = \alpha'$.

Given (D), (K), (O), (P), and (Q), by Lemma 5.26 we have (R) $L = L'$, (S) $J = J'$, and (T) $i = i'$.

Given (E), (L), (R), and (O), by Lemma 5.9 we have (U) $L_1 = L'_1$.

Given (F), (M), (N), (O), (Q), (S), (T), and (U), by Lemma 5.36 we have (V) $\sigma_1 = \sigma'_1$.

Given (A), (G), and (H), we have (W) (p, [$pin5$]) = (p, [$pin5$]).

Given (C) and (D), by Lemma 5.62 we have accessed location (X) (p, $[(l, 0)]$). Given (J) and (K), by Lemma 5.62 we have accessed location (Y) (p, $[(l', 0)]$). Given (X), (Y), and (N), we have (Z) (p, $[(l, 0)]$) = (p, $[(l', 0)]$).

Given (V), (Q), (U), (S), (T), (W) and (Z) by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \parallel C) \Downarrow_{(\text{p},[pin4])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ [n,\ L_1,\ J,\ 1]) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \parallel C) \Downarrow_{(\text{p},[pin5])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ [\alpha,\ L_1,\ J,\ i]) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \parallel C) \Downarrow_{(\text{p},[pin2])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \parallel C) \Downarrow_{(\text{p},[pin2])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ (l_2, \mu_2)) \parallel C)$ by SMC^2 rule Pre-Increment Public Pointer Higher Level Indirection Single Location, we have $i > 1$, (B) $\gamma(x) = (l,$ public $bty*)$, (C) $\sigma(l) = (\omega,$ public $bty*$, 1, PermL_Ptr(Freeable, public $bty*$, public, 1)), (D) DecodePtr(public $bty*$, 1, ω) = $[1,\ [(l_1, \mu_1)],\ [1],\ i]$, (E) $((l_2, \mu_2), 1)$ = GetLocation($(l_1, \mu_1), \tau$(public $bty*$)$, \sigma$), and (F) UpdatePtr($\sigma,\ (l, 0),\ [1,\ [(l_2, \mu_2)],\ [1],\ i]$, public bty) = $(\sigma_1,\ 1)$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ++\ x) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma'_1,\ \Delta,\ \text{acc},\ (l'_2, \mu'_2)) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = pin2$.

Given (G) and (H), by SMC^2 rule Pre-Increment Public Pointer Higher Level Indirection Single Location, we have $i' > 1$, (I) $\gamma(x) = (l',$ public $bty'*)$, (J) $\sigma(l') = (\omega',$ public $bty'*$, 1, PermL_Ptr(Freeable, public $bty'*$, public, 1)),

(K) DecodePtr(public $bty'*$, 1, ω') = [1, [(l_1', μ_1')], [1], i'], (L) $((l_2', \mu_2'), 1)$ = GetLocation((l_1', μ_1'), τ(public $bty'*$), σ), and (M) UpdatePtr(σ, $(l', 0)$, [1, [(l_2', μ_2')], [1], i'], public bty') = $(\sigma_1', 1)$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$ and (O) $bty = bty'$.

Given (C), (J), and (N), by Definition 5.4 we have (P) $\omega = \omega'$.

Given (D), (K), (O), and (P), by Lemma 5.26 we have (Q) $l_1 = l_1'$, (R) $\mu_1 = \mu_1'$, and (S) $i = i'$.

Given (E), (L), (O), (Q), and (R), by Lemma 5.10 we have (T) $l_2 = l_2'$ and (U) $\mu_2 = \mu_2'$.

Given (F), (M), (N), (O), (S), (T), and (U), by Lemma 5.36 we have (V) $\sigma_1 = \sigma_1'$.

Given (A), (G), and (H), we have (W) $(\mathrm{p}, [pin2]) = (\mathrm{p}, [pin2])$.

Given (C) and (D), by Lemma 5.62 we have accessed location (X) $(\mathrm{p}, [(l, 0)])$. Given (J) and (K), by Lemma 5.62 we have accessed location (Y) $(\mathrm{p}, [(l', 0)])$. Given (X), (Y), and (N), we have (Z) $(\mathrm{p}, [(l, 0)]) = (\mathrm{p}, [(l', 0)])$.

Given (V), (T), (U), (W) and (Z) by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin1])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin2])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin6])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin2])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin7])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ {+}{+}\, x) \parallel C) \Downarrow_{(\mathrm{p},[pin2])}^{(\mathrm{p},[(l,0)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$ by SMC^2

rule Public If Else True, we have $(e) \nvDash \gamma, n \neq 0$, (B)$((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$, and (C) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, s_1) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)$.

Given (D) $\Sigma \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{p},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'} ((\mathrm{p}, \gamma, \sigma_2', \Delta_2', \mathrm{acc}, \mathrm{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (E) $d = iet$.

Given (D) and (E), by SMC^2 rule Public If Else True, we have $(e) \nvDash \gamma, n' \neq 0$ (F)$((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, n') \parallel C_1')$, and (G) $((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, s_1') \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\mathrm{p}, \gamma_1', \sigma_2', \Delta_2', \mathrm{acc}, \mathrm{skip}) \parallel C_2')$.

Given (B) and (F), by the inductive hypothesis we have (H) $\sigma_1 = \sigma_1'$, (I) $\Delta_1 = \Delta_1'$, (J) $n = n'$, (K) $\mathcal{D}_1 = \mathcal{D}_1'$, (L) $\mathcal{L}_1 = \mathcal{L}_1'$, and (M) $C_1 = C_1'$.

Given (C), (G), (H), (I), and (M), by the inductive hypothesis we have (N) $\gamma_2 = \gamma_2'$, (O) $\sigma_2 = \sigma_2'$, (P) $\Delta_2 = \Delta_2'$, (Q) $\mathcal{D}_2 = \mathcal{D}_2'$, (R) $\mathcal{L}_2 = \mathcal{L}_2'$, and (S) $C_2 = C_2'$.

Given (K), (Q), and $(\mathrm{p}, [iet])$, by Lemma 5.38 we have (T) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [iet]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\mathrm{p}, [iet])$.

Given (L) and (R), by Lemma 5.47 we have (U) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (O), (P), (S), (T), and (U), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[ief])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[iet])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}::(\mathrm{p},[wle])}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}::(\mathrm{p},[wle])}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$ by SMC^2 rule While End, we have $(e) \nvDash \gamma, n = 0$, and (B) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}}^{\mathcal{L}} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$.

Given (C) $\Sigma \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}'::(\mathrm{p},[d])}^{\mathcal{L}'} ((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, \mathrm{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (D) $d = wle$.

Given (C) and (D), by SMC^2 rule While End, we have $(e) \nvDash \gamma, n' = 0$, and (E) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}'}^{\mathcal{L}'} ((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, n') \parallel C_1')$.

Given (B) and (E), by the inductive hypothesis we have (F) $\sigma_1 = \sigma_1'$, (G) $\Delta_1 = \Delta_1'$, (H) $n = n'$, (I) $\mathcal{D} = \mathcal{D}'$, (J) $\mathcal{L} = \mathcal{L}'$, and (K) $C_1 = C_1'$.

Given (I) and $(\mathrm{p}, [wle])$, by Lemma 5.38 we have (L) $\mathcal{D} :: (\mathrm{p}, [wle]) = \mathcal{D}' :: (\mathrm{p}, [wle])$.

Given (F), (G), (J), (L), and (K), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wlc])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wlc])}^{\mathcal{L}_1::\mathcal{L}_2} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C_2)$ by SMC^2

rule While Continue, we have $(e) \nvdash \gamma, n \neq 0$, (B) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$, and (C) $((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, s) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma_1, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_2)$.

Given (D) $\Sigma \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{p},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'} ((\mathrm{p}, \gamma, \sigma_2', \Delta_2', \mathrm{acc}, \mathrm{while}\ (e)\ s) \parallel C_2')$ and (A), by Lemma 4.87 we have (E) $d = wlc$.

Given (D) and (E), by SMC^2 rule While Continue, we have $(e) \nvdash \gamma, n \neq 0$, (F) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, n') \parallel C_1')$, and (G) $((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, s) \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\mathrm{p}, \gamma_1', \sigma_2', \Delta_2', \mathrm{acc}, \mathrm{skip}) \parallel C_2')$.

Given (B) and (F), by the inductive hypothesis we have (H) $\sigma_1 = \sigma_1'$, (I) $\Delta_1 = \Delta_1'$, (J) $n = n'$, (K) $\mathcal{D}_1 = \mathcal{D}_1'$, (L) $\mathcal{L}_1 = \mathcal{L}_1'$, and (M) $C_1 = C_1'$.

Given (C), (G), (H), (I), and (M), by the inductive hypothesis we have (N) $\gamma_1 = \gamma_1'$, (O) $\sigma_2 = \sigma_2'$, (P) $\Delta_2 = \Delta_2'$, (Q) $\mathcal{D}_2 = \mathcal{D}_2'$, (R) $\mathcal{L}_2 = \mathcal{L}_2'$, and (S) $C_2 = C_2'$.

Given (K), (Q), and $(\mathrm{p}, [wlc])$, by Lemma 5.38 we have (T) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wlc]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\mathrm{p}, [wlc])$.

Given (L) and (R), by Lemma 5.47 we have (U) $\mathcal{L}_1 :: \mathcal{L}_2 = \mathcal{L}_1' :: \mathcal{L}_2'$.

Given (O), (P), (S), (T), and (U), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[iep])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7}$ $((1, \gamma^1, \sigma_6^1, \Delta_3^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2))$ $\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[iep])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7} ((1, \gamma^1, \sigma_6^1, \Delta_3^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Private If Else (Variable Tracking), we have $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}$ $((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$, (C) $\{\mathrm{Extract}(s_1, s_2, \gamma^{\mathrm{p}}) = (x_{list}, 0)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (D) $\{\mathrm{InitializeVariables}(x_{list}, \gamma^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, n^{\mathrm{p}}, \mathrm{acc}+1) = (\gamma_1^{\mathrm{p}}, \sigma_2^{\mathrm{p}}, L_2^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $((1, \gamma_1^1, \sigma_2^1, \Delta_1^1, \mathrm{acc}+1, s_1) \parallel ... \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}+1, s_1)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_3} ((1, \gamma_2^1, \sigma_3^1, \Delta_2^1, \mathrm{acc}+1, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma_2^{\mathrm{q}}, \sigma_3^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, \mathrm{skip}))$, (F) $\{\mathrm{RestoreVariables}(x_{list}, \gamma_1^{\mathrm{p}}, \sigma_3^{\mathrm{p}}, \mathrm{acc}+1) = (\sigma_4^{\mathrm{p}}, L_4^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $((1, \gamma_1^1, \sigma_4^1, \Delta_2^1, \mathrm{acc}+1, s_2) \parallel ... \parallel (\mathrm{q}, \gamma_1^{\mathrm{q}}, \sigma_4^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}+1, s_2)) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_5} ((1, \gamma_3^1, \sigma_5^1, \Delta_3^1, \mathrm{acc}+1, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma_3^{\mathrm{q}}, \sigma_5^{\mathrm{q}}, \Delta_3^{\mathrm{q}}, \mathrm{acc}+1, \mathrm{skip}))$, (H) $\{\mathrm{ResolveVariables\_Retrieve}(x_{list}, \mathrm{acc}+1, \gamma_1^{\mathrm{p}}, \sigma_5^{\mathrm{p}}) = ([(v_{t1}^{\mathrm{p}}, v_{e1}^{\mathrm{p}}), ..., (v_{tm}^{\mathrm{p}}, v_{em}^{\mathrm{p}})], n_1^{\mathrm{p}}, L_6^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (I) $\mathrm{MPC}_{resolve}([n_1^1, ..., n_1^{\mathrm{q}}], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^{\mathrm{q}}, v_{e1}^{\mathrm{q}}), ..., (v_{tm}^{\mathrm{q}}, v_{em}^{\mathrm{q}})]]) = [[v_1^1, ..., v_m^1], ..., [v_1^{\mathrm{q}}, ..., v_m^{\mathrm{q}}]]$ (J) $\{\mathrm{ResolveVariables\_Store}(x_{list}, \mathrm{acc}+1, \gamma_1^{\mathrm{p}}, \sigma_5^{\mathrm{p}}, [v_1^{\mathrm{p}}, ..., v_m^{\mathrm{p}}]) = (\sigma_6^{\mathrm{p}}, L_7^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (K) $\mathcal{L}_2 = (1, L_2^1) \parallel ... \parallel (\mathrm{q}, L_2^{\mathrm{q}})$, (L) $\mathcal{L}_4 = (1, L_4^1) \parallel ... \parallel (\mathrm{q}, L_4^{\mathrm{q}})$, (M) $\mathcal{L}_6 = (1, L_6^1) \parallel ... \parallel (\mathrm{q}, L_6^{\mathrm{q}})$, and (N) $\mathcal{L}_7 = (1, L_7^1) \parallel ... \parallel (\mathrm{q}, L_7^{\mathrm{q}})$.

Given (O) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2))$ $\Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::\mathcal{D}_3'::(\mathrm{p},[d])}^{\mathcal{L}_1'::\mathcal{L}_2'::\mathcal{L}_3'::\mathcal{L}_4'::\mathcal{L}_5'::\mathcal{L}_6'::\mathcal{L}_7'} ((1, \gamma^1, \sigma_6'^1, \Delta_3'^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6'^{\mathrm{q}}, \Delta_3'^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ and (A), by Lemma 4.87 we have (P) $d = iep$.

Given (O) and (P), by SMC^2 rule Private If Else (Variable Tracking), we have $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Q) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \mathrm{acc}, n'^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1'^{\mathrm{q}}, \Delta_1'^{\mathrm{q}}, \mathrm{acc}, n'^{\mathrm{q}}))$, (R) $\{\mathrm{Extract}(s_1, s_2, \gamma^{\mathrm{p}}) = (x_{list}', 0)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (S) $\{\mathrm{InitializeVariables}(x_{list}', \gamma^{\mathrm{p}}, \sigma_1'^{\mathrm{p}}, n'^{\mathrm{p}}, \mathrm{acc}+1) = (\gamma_1'^{\mathrm{p}}, \sigma_2'^{\mathrm{p}}, L_2'^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (T) $((1, \gamma_1'^1, \sigma_2'^1,$

$\Delta_1'^1$, acc + 1, s_1) ∥ … ∥ (q, $\gamma_1'^q$, $\sigma_2'^q$, $\Delta_1'^q$, acc + 1, s_1)) $\Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_3'}$ ((1, $\gamma_2'^1$, $\sigma_3'^1$, $\Delta_2'^1$, acc + 1, skip) ∥ … ∥ (q, $\gamma_2'^q$, $\sigma_3'^q$, $\Delta_2'^q$, acc + 1, skip)), (U) {RestoreVariables(x'_{list}, $\gamma_1'^p$, $\sigma_3'^p$, acc + 1) = $(\sigma_4'^p, L_4'^p)\}_{p=1}^q$, (V) ((1, $\gamma_1'^1$, $\sigma_4'^1$, $\Delta_2'^1$, acc + 1, s_2) ∥ … ∥ (q, $\gamma_1'^q$, $\sigma_4'^q$, $\Delta_2'^q$, acc + 1, s_2)) $\Downarrow_{\mathcal{D}_3'}^{\mathcal{L}_5'}$ ((1, $\gamma_3'^1$, $\sigma_5'^1$, $\Delta_3'^1$, acc + 1, skip) ∥ … ∥ (q, $\gamma_3'^q$, $\sigma_5'^q$, $\Delta_3'^q$, acc + 1, skip)), (W) {ResolveVariables_Retrieve(x'_{list}, acc+1, $\gamma_1'^p$, $\sigma_5'^p$) = ([($v_{t1}'^p$, $v_{e1}'^p$), …, ($v_{tm}'^p$, $v_{em}'^p$)], $n_1'^p$, $L_6'^p$)$\}_{p=1}^q$, (X) $\text{MPC}_{resolve}$([$n_1'^1$, …, $n_1'^q$], [[($v_{t1}'^1$, $v_{e1}'^1$), …, ($v_{tm}'^1$, $v_{em}'^1$)], …, [($v_{t1}'^q$, $v_{e1}'^q$), …, ($v_{tm}'^q$, $v_{em}'^q$)]]) = [[$v_1'^1$, …, $v_m'^1$], …, [$v_1'^q$, …, $v_m'^q$]] (Y) {ResolveVariables_Store(x'_{list}, acc + 1, $\gamma_1'^p$, $\sigma_5'^p$, [$v_1'^p$, …, $v_m'^p$]) = $(\sigma_6'^p, L_7'^p)\}_{p=1}^q$, (Z) $\mathcal{L}_2' = (1, L_2'^1)$

$\parallel \ldots \parallel (\mathrm{q}, L_2'^{\mathrm{q}})$, (A1) $\mathcal{L}_4' = (1, L_4'^1) \parallel \ldots \parallel (\mathrm{q}, L_4'^{\mathrm{q}})$, (B1) $\mathcal{L}_6' = (1, L_6'^1) \parallel \ldots \parallel (\mathrm{q}, L_6'^{\mathrm{q}})$, and (C1) $\mathcal{L}_7' = (1, L_7'^1) \parallel \ldots \parallel (\mathrm{q}, L_7'^{\mathrm{q}})$.

Given (B) and (Q), by the inductive hypothesis we have (D1) $\{\sigma_1^{\mathrm{p}} = \sigma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E1) $\{\Delta_1^{\mathrm{p}} = \Delta_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F1) $\{n^{\mathrm{p}} = n'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G1) $\mathcal{D}_1 = \mathcal{D}_1'$, and (H1) $\mathcal{L}_1 = \mathcal{L}_1'$.

Given (C) and (R), by Lemma 5.16 we have (I1) $x_{list} = x_{list}'$.

Given (D), (S), (D1), and (F1), by Lemma 5.17 and we have (J1) $\{\gamma_1^{\mathrm{p}} = \gamma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (K1) $\{\sigma_2^{\mathrm{p}} = \sigma_2'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (L1) $\{L_2^{\mathrm{p}} = L_2'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (L1), (K), and (Z), by Lemma 5.53 and Definition 5.10 we have (M1) $\mathcal{L}_2 = \mathcal{L}_2'$.

Given (E), (T), (J1), (K1), and (E1), by the inductive hypothesis we have (N1) $\{\gamma_2^{\mathrm{p}} = \gamma_2'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (O1) $\{\sigma_3^{\mathrm{p}} = \sigma_3'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (P1) $\{\Delta_2^{\mathrm{p}} = \Delta_2'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Q1) $\mathcal{D}_2 = \mathcal{D}_2'$, and (R1) $\mathcal{L}_3 = \mathcal{L}_3'$.

Given (F), (U), (I1), (J1), and (O1), by Lemma 5.18 we have (S1) $\{\sigma_4^{\mathrm{p}} = \sigma_4'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (T1) $\{L_4^{\mathrm{p}} = L_4'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (T1), (L), and (A1), by Lemma 5.54 and Definition 5.10 we have (U1) $\mathcal{L}_4 = \mathcal{L}_4'$.

Given (G), (V), (J1), (S1), and (P1), by the inductive hypothesis we have (V1) $\{\gamma_3^{\mathrm{p}} = \gamma_3'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (W1) $\{\sigma_5^{\mathrm{p}} = \sigma_5'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (X1) $\{\Delta_3^{\mathrm{p}} = \Delta_3'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Y1) $\mathcal{D}_3 = \mathcal{D}_3'$, and (Z1) $\mathcal{L}_5 = \mathcal{L}_5'$.

Given (H), (W), (J1), (W1), (F1), and (I1), by Lemma 5.19 we have (A2) $\{[(v_{t1}^{\mathrm{p}}, v_{e1}^{\mathrm{p}}), \ldots, (v_{tm}^{\mathrm{p}}, v_{em}^{\mathrm{p}})] = [(v_{t1}'^{\mathrm{p}}, v_{e1}'^{\mathrm{p}}), \ldots, (v_{tm}'^{\mathrm{p}}, v_{em}'^{\mathrm{p}})]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (B2) $\{n_1^{\mathrm{p}} = n_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (C2) $\{L_6^{\mathrm{p}} = L_6'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (M), (B1), and (C2), by Lemma 5.55 and Definition 5.10 we have (D2) $\mathcal{L}_6 = \mathcal{L}_6'$.

Given (I), (X), (B2), and (A2), by Axiom 5.10 we have $[[v_1^1, \ldots, v_m^1], \ldots, [v_1^{\mathrm{q}}, \ldots, v_m^{\mathrm{q}}]] = [[v_1'^1, \ldots, v_m'^1], \ldots, [v_1'^{\mathrm{q}}, \ldots, v_m'^{\mathrm{q}}]]$ and therefore (E2) $\{[v_1^{\mathrm{p}}, \ldots, v_m^{\mathrm{p}}] = [v_1'^{\mathrm{p}}, \ldots, v_m'^{\mathrm{p}}]\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (J), (Y), (I1), (J1), (W1), and (E2), by Lemma 5.20 we have (F2) $\{\sigma_6^{\mathrm{p}} = \sigma_6'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (G2) $\{L_7^{\mathrm{p}} = L_7'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (N), (C1), and (G2), by Lemma 5.56 and Definition 5.10 we have (H2) $\mathcal{L}_7 = \mathcal{L}_7'$.

Given (G1), (Q1), (Y1), and (P), by Lemma 5.38 we have (I2) $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\mathrm{p}, [iep]) = \mathcal{D}_1 :: \mathcal{D}_2' :: \mathcal{D}_3'(\mathrm{p}, [iep])$.

Given (H1), (M1), (R1), (U1), (Z1), (D2), and (H2), by Lemma 5.47 we have (J2) $\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3 :: \mathcal{L}_4 :: \mathcal{L}_5 :: \mathcal{L}_6 :: \mathcal{L}_7 = \mathcal{L}_1' :: \mathcal{L}_2' :: \mathcal{L}_3' :: \mathcal{L}_4' :: \mathcal{L}_5' :: \mathcal{L}_6' :: \mathcal{L}_7'$.

Given (F2), (X1), (J2), and (I2), by Definition 5.2, we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[iepd])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7} ((1, \gamma^1, \sigma_6^1, \Delta_6^1, \mathrm{acc}, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_6^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, \mathrm{if}\ (e)\ s_1\ \mathrm{else}\ s_2)) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::\mathcal{D}_3::(\mathrm{p},[iepd])}^{\mathcal{L}_1::\mathcal{L}_2::\mathcal{L}_3::\mathcal{L}_4::\mathcal{L}_5::\mathcal{L}_6::\mathcal{L}_7} ((1, \gamma^1, \sigma_6^1, \Delta_6^1, \mathrm{acc}, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_6^{\mathrm{q}}, \Delta_6^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Private If Else (Location Tracking), we have $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (B) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1}$

$((1, \gamma^1, \sigma_1^1, \Delta_1^1, \text{acc}, n^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1^{\text{q}}, \Delta_1^{\text{q}}, \text{acc}, n^{\text{q}}))$, (C) $\{\text{Extract}(s_1, s_2, \gamma^{\text{p}}) = (x_{list}, 1)\}_{\text{p}=1}^{\text{q}}$, (D) $\{\text{Initialize}(\Delta_1^{\text{p}}, x_{list}, \gamma^{\text{p}}, \sigma_1^{\text{p}}, n^{\text{p}}, \text{acc} + 1) = (\gamma_1^{\text{p}}, \sigma_2^{\text{p}}, \Delta_2^{\text{p}}, L_2^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (E) $((1, \gamma_1^1, \sigma_2^1, \Delta_2^1, \text{acc} + 1, s_1) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_2^{\text{q}}, \Delta_2^{\text{q}}, \text{acc} + 1, s_1)) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_3} ((1, \gamma_2^1, \sigma_3^1, \Delta_3^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_2^{\text{q}}, \sigma_3^{\text{q}}, \Delta_3^{\text{q}}, \text{acc} + 1, \text{skip}))$, (F) $\{\text{Restore}(\sigma_3^{\text{p}}, \Delta_3^{\text{p}}, \text{acc}+1) = (\sigma_4^{\text{p}}, \Delta_4^{\text{p}}, L_4^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (G) $((1, \gamma_1^1, \sigma_4^1, \Delta_4^1, \text{acc}+1, s_2) \parallel ... \parallel (\text{q}, \gamma_1^{\text{q}}, \sigma_4^{\text{q}}, \Delta_4^{\text{q}}, \text{acc}+1, s_2)) \Downarrow_{\mathcal{D}_3}^{\mathcal{L}_5} ((1, \gamma_3^1, \sigma_5^1, \Delta_5^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_3^{\text{q}}, \sigma_5^{\text{q}}, \Delta_5^{\text{q}}, \text{acc} + 1, \text{skip}))$, (H) $\{\text{Resolve\_Retrieve}(\gamma_1^{\text{p}}, \sigma_5^{\text{p}}, \Delta_5^{\text{p}}, \text{acc} + 1) = ([(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})], n_1^{\text{p}}, L_6^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (I) $\text{MPC}_{resolve}([n_1^1, ..., n_1^{\text{q}}], [[(v_{t1}^1, v_{e1}^1), ..., (v_{tm}^1, v_{em}^1)], ..., [(v_{t1}^{\text{q}}, v_{e1}^{\text{q}}), ..., (v_{tm}^{\text{q}}, v_{em}^{\text{q}})]]) = [[v_1^1, ..., v_m^1], ..., [v_1^{\text{q}}, ..., v_m^{\text{q}}]]$ (J) $\{\text{Resolve\_Store}(\Delta_5^{\text{p}}, \sigma_5^{\text{p}}, \text{acc} + 1, [v_1^{\text{p}}, ..., v_m^{\text{p}}]) = (\sigma_6^{\text{p}}, \Delta_6^{\text{p}}, L_7^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (K) $\mathcal{L}_2 = (1, L_2^1) \parallel ... \parallel (\text{q}, L_2^{\text{q}})$, (L) $\mathcal{L}_4 = (1, L_4^1) \parallel ... \parallel (\text{q}, L_4^{\text{q}})$, (M) $\mathcal{L}_6 = (1, L_6^1) \parallel ... \parallel (\text{q}, L_6^{\text{q}})$, and (N) $\mathcal{L}_7 = (1, L_7^1) \parallel ... \parallel (\text{q}, L_7^{\text{q}})$.

Given (O) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{if } (e)\ s_1 \text{ else } s_2) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, \text{if } (e)\ s_1 \text{ else } s_2)) \Downarrow_{\mathcal{D}'_1::\mathcal{D}'_2::\mathcal{D}'_3::(\text{p},[d])}^{\mathcal{L}'_1::\mathcal{L}'_2::\mathcal{L}'_3::\mathcal{L}'_4::\mathcal{L}'_5::\mathcal{L}'_6::\mathcal{L}'_7} ((1, \gamma^1, \sigma_6'^1, \Delta_6'^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_6'^{\text{q}}, \Delta_6'^{\text{q}}, \text{acc}, \text{skip}))$ and (A), by Lemma 4.87 we have (P) $d = iepd$.

Given (O) and (P), by SMC^2 rule Private If Else (Location Tracking), we have $\{(e) \vdash \gamma^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (Q) $((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, e) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma^{\text{q}}, \Delta^{\text{q}}, \text{acc}, e)) \Downarrow_{\mathcal{D}'_1}^{\mathcal{L}'_1} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \text{acc}, n'^1) \parallel ... \parallel (\text{q}, \gamma^{\text{q}}, \sigma_1'^{\text{q}}, \Delta_1'^{\text{q}}, \text{acc}, n'^{\text{q}}))$, (R) $\{\text{Extract}(s_1, s_2, \gamma^{\text{p}}) = (x'_{list}, 1)\}_{\text{p}=1}^{\text{q}}$, (S) $\{\text{Initialize}(\Delta_1'^{\text{p}}, x'_{list}, \gamma'^{\text{p}}, \sigma_1'^{\text{p}}, n'^{\text{p}}, \text{acc}+1) = (\gamma_1'^{\text{p}}, \sigma_2'^{\text{p}}, \Delta_2'^{\text{p}}, L_2'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (T) $((1, \gamma_1'^1, \sigma_2'^1, \Delta_2'^1, \text{acc} + 1, s_1) \parallel ... \parallel (\text{q}, \gamma_1'^{\text{q}}, \sigma_2'^{\text{q}}, \Delta_2'^{\text{q}}, \text{acc} + 1, s_1)) \Downarrow_{\mathcal{D}'_2}^{\mathcal{L}'_3} ((1, \gamma_2'^1, \sigma_3'^1, \Delta_3'^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_2'^{\text{q}}, \sigma_3'^{\text{q}}, \Delta_3'^{\text{q}}, \text{acc} + 1, \text{skip}))$, (U) $\{\text{Restore}(\sigma_3'^{\text{p}}, \Delta_3'^{\text{p}}, \text{acc} + 1) = (\sigma_4'^{\text{p}}, \Delta_4'^{\text{p}}, L_4'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (V) $((1, \gamma_1'^1, \sigma_4'^1, \Delta_4'^1, \text{acc} + 1, s_2) \parallel ... \parallel (\text{q}, \gamma_1'^{\text{q}}, \sigma_4'^{\text{q}}, \Delta_4'^{\text{q}}, \text{acc} + 1, s_2)) \Downarrow_{\mathcal{D}'_3}^{\mathcal{L}'_5} ((1, \gamma_3'^1, \sigma_5'^1, \Delta_5'^1, \text{acc} + 1, \text{skip}) \parallel ... \parallel (\text{q}, \gamma_3'^{\text{q}}, \sigma_5'^{\text{q}}, \Delta_5'^{\text{q}}, \text{acc} + 1, \text{skip}))$, (W) $\{\text{Resolve\_Retrieve}(\gamma_1'^{\text{p}}, \sigma_5'^{\text{p}}, \Delta_5'^{\text{p}}, \text{acc} + 1) = ([(v_{t1}'^{\text{p}}, v_{e1}'^{\text{p}}), ..., (v_{tm}'^{\text{p}}, v_{em}'^{\text{p}})], n_1'^{\text{p}}, L_6'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (X) $\text{MPC}_{resolve}([n_1'^1, ..., n_1'^{\text{q}}], [[(v_{t1}'^1, v_{e1}'^1), ..., (v_{tm}'^1, v_{em}'^1)], ..., [(v_{t1}'^{\text{q}}, v_{e1}'^{\text{q}}), ..., (v_{tm}'^{\text{q}}, v_{em}'^{\text{q}})]]) = [[v_1'^1, ..., v_m'^1], ..., [v_1'^{\text{q}}, ..., v_m'^{\text{q}}]]$

(Y) $\{\text{Resolve\_Store}(\Delta_5'^{\text{p}}, \sigma_5'^{\text{p}}, \text{acc}+1, [v_1'^{\text{p}}, ..., v_m'^{\text{p}}]) = (\sigma_6'^{\text{p}}, \Delta_6'^{\text{p}}, L_7'^{\text{p}})\}_{\text{p}=1}^{\text{q}}$, (Z) $\mathcal{L}_2' = (1, L_2'^1) \parallel ... \parallel (\text{q}, L_2'^{\text{q}})$, (A1) $\mathcal{L}_4' = (1, L_4'^1) \parallel ... \parallel (\text{q}, L_4'^{\text{q}})$, (B1) $\mathcal{L}_6' = (1, L_6'^1) \parallel ... \parallel (\text{q}, L_6'^{\text{q}})$, and (C1) $\mathcal{L}_7' = (1, L_7'^1) \parallel ... \parallel (\text{q}, L_7'^{\text{q}})$.

Given (B) and (Q), by the inductive hypothesis we have (D1) $\{\sigma_1^{\text{p}} = \sigma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (E1) $\{\Delta_1^{\text{p}} = \Delta_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (F1) $\{n^{\text{p}} = n'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (G1) $\mathcal{D}_1 = \mathcal{D}_1'$, and (H1) $\mathcal{L}_1 = \mathcal{L}_1'$.

Given (C) and (R), by Lemma 5.16 we have (I1) $x_{list} = x_{list}'$.

Given (D), (S), (D1), (E1), and (F1), by Lemma 5.21 we have (J1) $\{\gamma_1^{\text{p}} = \gamma_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (K1) $\{\sigma_2^{\text{p}} = \sigma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (L1) $\{\Delta_2^{\text{p}} = \Delta_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and (M1) $\{L_2^{\text{p}} = L_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (M1), (K), and (Z), by Lemma 5.57 and Definition 5.10 we have (N1) $\mathcal{L}_2 = \mathcal{L}_2'$.

Given (E), (T), (J1), (K1), and (L1), by the inductive hypothesis we have $\{\gamma_2^{\text{p}} = \gamma_2'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (O1) $\{\sigma_3^{\text{p}} = \sigma_3'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (P1) $\{\Delta_3^{\text{p}} = \Delta_3'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (Q1) $\mathcal{D}_2 = \mathcal{D}_2'$, and (R1) $\mathcal{L}_3 = \mathcal{L}_3'$.

Given (F), (U), (P1), and (O1), by Lemma 5.22 we have (S1) $\{\sigma_4^{\text{p}} = \sigma_4'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (T1) $\{\Delta_4^{\text{p}} = \Delta_4'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and (U1) $\{L_4^{\text{p}} = L_4'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (U1), (L), and (A1), by Lemma 5.58 and Definition 5.10 we have (V1) $\mathcal{L}_4 = \mathcal{L}_4'$.

Given (G), (V), (J1), (S1), and (T1), by the inductive hypothesis we have $\{\gamma_3^{\text{p}} = \gamma_3'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (W1) $\{\sigma_5^{\text{p}} = \sigma_5'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (X1) $\{\Delta_5^{\text{p}} = \Delta_5'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (Y1) $\mathcal{D}_3 = \mathcal{D}_3'$, and (Z1) $\mathcal{L}_5 = \mathcal{L}_5'$.

Given (H), (W), (J1), (W1), and (X1), by Lemma 5.23 we have (A2) $\{[(v_{t1}^{\text{p}}, v_{e1}^{\text{p}}), ..., (v_{tm}^{\text{p}}, v_{em}^{\text{p}})] = [(v_{t1}'^{\text{p}}, v_{e1}'^{\text{p}}), ..., (v_{tm}'^{\text{p}}, v_{em}'^{\text{p}})]\}_{\text{p}=1}^{\text{q}}$, (B2) $\{n_1^{\text{p}} = n_1'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and (C2) $\{L_6^{\text{p}} = L_6'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (M), (B1), and (C2), by Lemma 5.59 and Definition 5.10 we have (D2) $\mathcal{L}_6 = \mathcal{L}_6'$.

Given (I), (X), (B2), and (A2), by Axiom 5.10 we have $[[v_1^1, ..., v_m^1], ..., [v_1^{\text{q}}, ..., v_m^{\text{q}}]] = [[v_1'^1, ..., v_m'^1], ..., [v_1'^{\text{q}}, ..., v_m'^{\text{q}}]]$ and therefore (E2) $\{[v_1^{\text{p}}, ..., v_m^{\text{p}}] = [v_1'^{\text{p}}, ..., v_m'^{\text{p}}]\}_{\text{p}=1}^{\text{q}}$.

Given (J), (Y), (X1), (W1), and (E2), by Lemma 5.24 we have (F2) $\{\sigma_6^{\text{p}} = \sigma_6'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, (G2) $\{\Delta_6^{\text{p}} = \Delta_6'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$, and (H2) $\{L_7^{\text{p}} = L_7'^{\text{p}}\}_{\text{p}=1}^{\text{q}}$.

Given (N), (C1), and (H2), by Lemma 5.60 and Definition 5.10 we have (I2) $\mathcal{L}_7 = \mathcal{L}_7'$.

Given (G1), (Q1), (Y1), and (P), by Lemma 5.38 we have (J2) $\mathcal{D}_1 :: \mathcal{D}_2 :: \mathcal{D}_3 :: (\text{p}, [iepd]) = \mathcal{D}_1 :: \mathcal{D}_2' :: \mathcal{D}_3'(\text{p}, [iepd])$.

Given (H1), (N1), (R1), (V1), (Z1), (D2), and (I2), by Lemma 5.47 we have (K2) $\mathcal{L}_1 :: \mathcal{L}_2 :: \mathcal{L}_3 :: \mathcal{L}_4 :: \mathcal{L}_5 :: \mathcal{L}_6 :: \mathcal{L}_7 = \mathcal{L}_1' :: \mathcal{L}_2' :: \mathcal{L}_3' :: \mathcal{L}_4' :: \mathcal{L}_5' :: \mathcal{L}_6' :: \mathcal{L}_7'$.

Given (F2), (G2), (J2), and (K2), by Definition 5.2, we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1, \ \sigma_3, \ \Delta, \ \text{acc}, \ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1, \ \sigma_3, \ \Delta, \ \text{acc}, \ \text{skip}) \parallel C_1)$ by SMC[2] rule Public Array Declaration we have $((ty = \text{public } bty) \land ((bty = \text{float}) \lor (bty = \text{char}) \lor (bty = \text{int}))) \lor (ty = \text{char})$, $(e) \nvdash \gamma$, $\alpha > 0$, acc $= 0$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C), \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta,\ \text{acc},\ \alpha) \parallel C_1)$, (C) $l = \phi()$, (D) $l_1 = \phi()$, (E) $\omega_1 = \text{EncodeArr}(\text{public } bty, 0, \alpha, \text{NULL})$, (F) $\gamma_1 = \gamma[x \to (l, \text{public const } bty*)]$, (G) $\omega = \text{EncodePtr}(\text{public const } bty*, [1, [(l_1, 0)], [1], 1])$, (H) $\sigma_2 = \sigma_1[l \to (\omega, \text{public const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const } bty*, \text{public}, 1))]$, and (I) $\sigma_3 = \sigma_2[l_1 \to (\omega_1, \text{public } bty, \alpha, \text{PermL}(\text{Freeable}, \text{public } bty, \text{public}, \alpha))]$.

Given (J) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0),(l_1',0)])} ((\text{p}, \gamma_1',\ \sigma_3',\ \Delta,\ \text{acc},\ \text{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (K) $d = da$.

Given (J) and (K), by SMC^2 rule Public Array Declaration we have $((ty = \text{public } bty) \land ((bty = \text{float}) \lor (bty = \text{char}) \lor (bty = \text{int}))) \lor (ty = \text{char})$, $(e) \nvdash \gamma$, $\alpha' > 0$, acc $= 0$, (L) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C), \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta,\ \text{acc},\ \alpha') \parallel C_1')$, (M) $l' = \phi()$, (N) $l_1' = \phi()$, (O) $\omega_1' = \text{EncodeArr}(\text{public } bty, 0, \alpha', \text{NULL})$, (P) $\gamma_1' = \gamma[x \to (l', \text{public const } bty*)]$, (Q) $\omega' = \text{EncodePtr}(\text{public const } bty*, [1, [(l_1', 0)], [1], 1])$, (R) $\sigma_2' = \sigma_1'[l' \to (\omega', \text{public const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const } bty*, \text{public}, 1))]$, and (S) $\sigma_3' = \sigma_2'[l_1' \to (\omega_1', \text{public } bty, \alpha', \text{PermL}(\text{Freeable}, \text{public } bty, \text{public}, \alpha'))]$.

Given (B) and (L), by the inductive hypothesis we have (T) $\sigma_1 = \sigma_1'$, (U) $\alpha = \alpha'$, (V) $\mathcal{D}_1 = \mathcal{D}_1'$, (W) $\mathcal{L}_1 = \mathcal{L}_1'$, and (X) $C_1 = C_1'$.

Given (C), (D), (M), and (N), by Axiom 5.4 we have (Y) $l = l'$ and (Z) $l_1 = l_1'$.

Given (E) , (O), and (U), by Lemma 5.31 we have (A1) $\omega_1 = \omega_1'$.

Given (F), (P), and (Y), by Definition 5.3 we have (B1) $\gamma_1 = \gamma_1'$.

Given (G), (Q), and (Z), by Lemma 5.32 we have (C1) $\omega = \omega'$.

Given (H), (R), (T), (Y), and (C1), by Definition 5.4 we have (D1) $\sigma_2 = \sigma_2'$.

Given (I), (S), (D1), (Z), and (A1), by Definition 5.4 we have (E1) $\sigma_3 = \sigma_3'$.

Given (X) and $(\text{p}, [da])$, by Lemma 5.38 we have (F1) $\mathcal{D}_1 :: (\text{p}, [da]) = \mathcal{D}_1' :: (\text{p}, [da])$.

Given (H) and (I), by Lemma 5.51 we have accessed (G1) $(\text{p}, [(l, 0)])$ and (H1) $(\text{p}, [(l_1, 0)])$. Given (G1) and (H1), by Lemmas 5.44 and 5.45 we have (I1) $(\text{p}, [(l, 0), (l_1, 0)])$. Given (R) and (S), by Lemma 5.51 we have accessed (J1) $(\text{p}, [(l', 0)])$ and (K1) $(\text{p}, [(l_1', 0)])$. Given (J1) and (K1), by Lemmas 5.44 and 5.45 we have (L1) $(\text{p}, [(l', 0), (l_1', 0)])$. Given (I1), (L1), (Y), (Z), and (W), by Lemma 5.47 we have (M1) $\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_1, 0)]) = \mathcal{L}_1' :: (\text{p}, [(l', 0), (l_1', 0)])$.

Given (B1), (E1), (X), (F1), and (M1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ ty\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[da])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0)])} ((\text{p}, \gamma_1,\ \sigma_3,\ \Delta,\ \text{acc},\ \text{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[ra1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,i)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_i) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[ra1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,i)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n_i) \parallel C_1)$ by SMC^2 rule Private

Array Read Public Index we have $0 \leq i \leq \alpha - 1$, $(e) \nvdash \gamma$, (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{private const } bty*)$, (D) $\sigma_1(l) = (\omega,\ \text{private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable, private const } bty*, \text{private}, 1))$, (E) $\text{DecodePtr}(\text{private const } bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (F) $\sigma_1(l_1) = (\omega_1, \text{private } bty, \alpha, \text{PermL}(\text{Freeable, private } bty, \text{private}, \alpha))$, and (G) $\text{DecodeArr}(\text{private } bty,\ i,\ \omega_1) = n_i$.

Given (H) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1'::(\mathrm{p},[d])}^{\mathcal{L}_1'::(\mathrm{p},[(l',0),(l_1',i')])} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ n_{i'}') \parallel C_1')$ and (A), by Lemma 4.87 we have (I) $d = ra1$.

Given (H) and (I), by SMC[2] rule Private Array Read Public Index we have $0 \leq i' \leq \alpha' - 1$, $(e) \nvdash \gamma$, (J) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ i') \parallel C_1')$, (K) $\gamma(x) = (l',\ \text{private const } bty'*)$, (L) $\sigma_1'(l') = (\omega',\ \text{private const } bty'*, 1, \text{PermL\_Ptr}(\text{Freeable, private const } bty'*, \text{private}, 1))$, (M) $\text{DecodePtr}(\text{private const } bty'*,\ 1,\ \omega') = [1,\ [(l_1', 0)],\ [1],\ 1]$, (N) $\sigma_1'(l_1') = (\omega_1', \text{private } bty', \alpha', \text{PermL}(\text{Freeable, private } bty', \text{private}, \alpha'))$, and (O) $\text{DecodeArr}(\text{private } bty',\ i',\ \omega_1') = n_{i'}'$.

Given (B) and (J), by the inductive hypothesis we have (P) $\sigma_1 = \sigma_1'$, (Q) $\Delta_1 = \Delta_1'$, (R) $i = i'$, (S) $\mathcal{D}_1 = \mathcal{D}_1'$, (T) $\mathcal{L}_1 = \mathcal{L}_1'$, and (U) $C_1 = C_1'$.

Given (C) and (K), by Definition 5.3 we have (V) $l = l'$ and (W) $bty = bty'$.

Given (D), (L), (P), and (V), by Definition 5.4 we have (X) $\omega = \omega'$.

Given (E), (M), (W), and (X), by Lemma 5.26 we have (Y) $l_1 = l_1'$.

Given (F), (N), (P), and (Y), by Definition 5.4 we have (Z) $\omega_1 = \omega_1'$ and (A1) $\alpha = \alpha'$.

Given (G), (O), (W), (R), and (Z), by Lemma 5.27 we have (B1) $n_i = n_{i'}'$.

Given (S) and $(\mathrm{p}, [ra1])$, by Lemma 5.38 we have (C1) $\mathcal{D}_1 :: (\mathrm{p}, [ra1]) = \mathcal{D}_1' :: (\mathrm{p}, [ra1])$.

Given (D) and (E), by Lemma 5.62 we have accessed location (D1) $(\mathrm{p}, [(l, 0)])$. Given (F) and (G), by Lemma 5.63 we have accessed location (E1) $(\mathrm{p}, [(l_1, i)])$. Given (D1) and (E1), by Lemmas 5.44 and 5.45 we have (F1) $(\mathrm{p}, [(l, 0), (l_1, i)])$.

Given (L) and (M), by Lemma 5.62 we have accessed location (G1) $(\mathrm{p}, [(l', 0)])$. Given (N) and (O), by Lemma 5.63 we have accessed location (H1) $(\mathrm{p}, [(l_1', i')])$. Given (D1) and (E1), by Lemmas 5.44 and 5.45 we have (I1) $(\mathrm{p}, [(l', 0), (l_1', i')])$.

Given (F1), (I1), (V), (Y), (R), and (T), by Lemma 5.47 we have (J1) $\mathcal{L}_1 :: (\mathrm{p}, [(l, 0), (l_1, i)]) = \mathcal{L}_1' :: (\mathrm{p}, [(l', 0), (l_1', i')])$.

Given (P), (Q), (B1), (U), (C1), and (J1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[ra])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,i)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[ra1])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,i)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n_i) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[rao])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e]) \parallel C) \Downarrow_{\mathcal{D}_1::(\mathrm{p},[rao])}^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_2,\mu)])} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ n) \parallel C_1)$ by SMC[2] rule Public Array Read Out of Bounds Public Index we have $(e) \nvdash \gamma$, $(i < 0) \vee (i \geq \alpha)$, (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e)$

$\| C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ i) \| C_1)$, (C) $\gamma(x) = (l,\ \text{public const}\ bty*)$, (D) $\sigma_1(l) = (\omega,\ \text{public const}\ bty*, 1,$ $\text{PermL\_Ptr}(\text{Freeable}, \text{public const}\ bty*, \text{public}, 1))$, (E) $\text{DecodePtr}(\text{public const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1, 0)],\ [1],$ $1]$, (F) $\sigma_1(l_1) = (\omega_1, \text{public}\ bty, \alpha, \text{PermL}(\text{Freeable}, \text{public}\ bty, \text{public}, \alpha))$, (G) $\text{ReadOOB}(i, \alpha, l_1, \text{public}\ bty, \sigma_1)$ $= (n, 1, (l_2, \mu))$.

Given (H) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \| C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0),(l_2',\mu')])} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ n') \| C_1')$ and (A), by Lemma 4.87 we have (I) $d = rao$.

Given (H) and (I), by SMC^2 rule Public Array Read Out of Bounds Public Index we have $(e) \nvdash \gamma$, $(i < 0) \vee (i \geq \alpha)$, (J) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \| C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ i') \| C_1')$, (K) $\gamma(x) = (l',\ \text{public const}\ bty'*)$, (L) $\sigma_1'(l') = (\omega',\ \text{public const}\ bty'*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public const}\ bty'*, \text{public}, 1))$, (M) $\text{DecodePtr}(\text{public const}\ bty'*,\ 1,\ \omega') = [1,\ [(l_1', 0)],\ [1],\ 1]$, (N) $\sigma_1'(l_1') = (\omega_1', \text{public}\ bty', \alpha', \text{PermL}(\text{Freeable}, \text{public}\ bty', \text{public}, \alpha'))$, (O) $\text{ReadOOB}(i', \alpha', l_1', \text{public}\ bty', \sigma_1') = (n', 1, (l_2', \mu'))$.

Given (B) and (J), by the inductive hypothesis we have (P) $\sigma_1 = \sigma_1'$, (Q) $\Delta_1 = \Delta_1'$, (R) $i = i'$, (S) $\mathcal{D}_1 = \mathcal{D}_1'$, (T) $\mathcal{L}_1 = \mathcal{L}_1'$, and (U) $C_1 = C_1'$.

Given (C) and (K), by Definition 5.3 we have (V) $l = l'$ and (W) $bty = bty'$.

Given (D), (L), (P), and (V), by Definition 5.4 we have (X) $\omega = \omega'$.

Given (E), (M), (W), and (X), by Lemma 5.26 we have (Y) $l_1 = l_1'$.

Given (F), (N), (P), and (Y), by Definition 5.4 we have (Z) $\omega_1 = \omega_1'$ and (A1) $\alpha = \alpha'$.

Given (G), (O), (R), (A1), (Y), (X), and (P), by Lemma 5.11 we have (B1) $n = n'$ and (C1) $(l_2, \mu) = (l_2', \mu')$.

Given (S) and $(\text{p}, [rao])$, by Lemma 5.38 we have (D1) $\mathcal{D}_1 :: (\text{p}, [rao]) = \mathcal{D}_1' :: (\text{p}, [rao])$.

Given (D) and (E) by Lemma 5.62 we have accessed location (E1) $(\text{p}, [(l, 0)])$. Given (G), by Lemma 5.49 we have accessed location (F1) $(\text{p}, [(l_2, \mu)])$. Given (E1) and (F1), by Lemmas 5.44 and 5.45 we have (G1) $(\text{p}, [(l, 0), (l_2, \mu)])$.

Given (L) and (M) by Lemma 5.62 we have accessed location (H1) $(\text{p}, [(l', 0)])$. Given (O), by Lemma 5.49 we have accessed location (I1) $(\text{p}, [(l_2', \mu')])$. Given (H1) and (I1), by Lemmas 5.44 and 5.45 we have (J1) $(\text{p}, [(l', 0), (l_2', \mu')])$.

Given (G1), (J1), (T), (V), and (C1), by Lemma 5.47 we have (K1) $\mathcal{L}_1 :: (\text{p}, [(l, 0), (l_2, \mu)]) = \mathcal{L}_1' :: (\text{p}, [(l', 0), (l_2', \mu')])$.

Given (P), (Q), (B1), (U), (D1), and (K1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \| C) \Downarrow_{\mathcal{D}_1::(\text{p},[rao1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_2,\mu)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \| C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e]) \| C) \Downarrow_{\mathcal{D}_1::(\text{p},[rao])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_2,\mu)])} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ n) \| C_1)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e_1]\ = e_2) \| C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[wao2])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_2,\mu)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \text{acc},\ \text{skip}) \| C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x[e_1]\ = e_2) \| C) \Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[wao2])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_2,\mu)])} ((\text{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \text{acc},\ \text{skip}) \| C_2)$ by SMC^2 rule Private Array Write Out of Bounds Public Index Private Value we have $(e_1) \nvdash \gamma$, $(e_2) \vdash \gamma$,

$(i < 0) \vee (i \geq \alpha)$ (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1)$, (C) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2)$, (D) $\gamma(x) = (l, \text{private const } bty*)$, (E) $\sigma_2(l) = (\omega, \text{private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1))$, (F) $\text{DecodePtr}(\text{private const } bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (G) $\sigma_2(l_1) = (\omega_1,\ \text{private } bty,\ \alpha,\ \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha))$, and (H) $\text{WriteOOB}(n,\ i,\ \alpha,\ l_1,\ \text{private } bty,\ \sigma_2,\ \Delta_2,\ \mathrm{acc}) = (\sigma_3,\ \Delta_3, 1, (l_2, \mu))$.

Given (I) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{p}, [wao2])}^{\mathcal{L}_1'::\mathcal{L}_2'::(\mathrm{p}, [(l', 0), (l_2', \mu')])} ((\mathrm{p}, \gamma,\ \sigma_3',\ \Delta_3',\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (J) $d = wao2$.

Given (I) and (J), by SMC[2] rule Private Array Write Out of Bounds Public Index Private Value we have $(e_1) \nvdash \gamma, (e_2) \vdash \gamma, (i' < 0) \vee (i' \geq \alpha')$ (K) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ i') \parallel C_1')$, (L) $((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ e_2') \parallel C_1') \Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'} ((\mathrm{p}, \gamma,\ \sigma_2',\ \Delta_2',\ \mathrm{acc},\ n') \parallel C_2')$, (M) $\gamma(x) = (l', \text{private const } bty'*)$, (N) $\sigma_2'(l') = (\omega', \text{private const } bty'*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty'*, \text{private}, 1))$, (O) $\text{DecodePtr}(\text{private const } bty'*, 1, \omega') = [1,\ [(l_1', 0)],\ [1],\ 1]$, (P) $\sigma_2'(l_1') = (\omega_1',\ \text{private } bty',\ \alpha',\ \text{PermL}(\text{Freeable}, \text{private } bty', \text{private}, \alpha'))$, and (Q) $\text{WriteOOB}(n',\ i',\ \alpha',\ l_1',\ \text{private } bty',\ \sigma_2',\ \Delta_2',\ \mathrm{acc}) = (\sigma_3',\ \Delta_3', 1, (l_2', \mu'))$.

Given (B) and (K), by the inductive hypothesis we have (R) $\sigma_1 = \sigma_1'$, (S) $\Delta_1 = \Delta_1'$, (T) $i = i'$, (U) $\mathcal{D}_1 = \mathcal{D}_1'$, (V) $\mathcal{L}_1 = \mathcal{L}_1'$, and (W) $C_1 = C_1'$.

Given (C), (L), (R), (S), and (W), by the inductive hypothesis we have (X) $\sigma_2 = \sigma_2'$, (Y) $\Delta_2 = \Delta_2'$, (Z) $n = n'$, (A1) $\mathcal{D}_2 = \mathcal{D}_2'$, (B1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (C1) $C_2 = C_2'$.

Given (D) and (M), by Definition 5.3 we have (D1) $l = l'$ and (E1) $bty = bty'$.

Given (E), (N), (X), and (D1), by Definition 5.4 we have (F1) $\omega = \omega'$.

Given (F), (O), (E1), and (F1), by Lemma 5.26 we have (G1) $l_1 = l_1'$.

Given (G), (P), (X), and (G1), by Definition 5.4 we have (H1) $\omega_1 = \omega_1'$ and (I1) $\alpha = \alpha'$.

Given (H), (Q), (Z), (T), (I1), (G1), (E1), (X), and (Y), by Lemma 5.12 we have (J1) $\sigma_3 = \sigma_3'$, (K1) $\Delta_3 = \Delta_3'$, and (L1) $(l_2, \mu) = (l_2', \mu')$.

Given (U), (A1), and $(\mathrm{p}, [wao2])$, by Lemma 5.38 we have (M1) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\mathrm{p}, [wao2]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\mathrm{p}, [wao2])$.

Given (E) and (F) by Lemma 5.62 we have accessed location (N1) $(\mathrm{p}, [(l, 0)])$. Given (H), by Lemma 5.50 we have accessed location (O1) $(\mathrm{p}, [(l_2, \mu)])$. Given (N1) and (O1), by Lemmas 5.44 and 5.45 we have (P1) $(\mathrm{p}, [(l, 0), (l_2, \mu)])$.

Given (N) and (O) by Lemma 5.62 we have accessed location (Q1) $(\mathrm{p}, [(l', 0)])$. Given (Q), by Lemma 5.50

we have accessed location (R1) (p, $[(l'_2, \mu')]$). Given (Q1) and (R1), by Lemmas 5.44 and 5.45 we have (S1) (p, $[(l', 0), (l'_2, \mu')]$).

Given (P1), (S1), (V), (B1), (D1), and (L1), by Lemma 5.47 we have (T1) $\mathcal{L}_1 :: \mathcal{L}_2 ::$ (p, $[(l, 0), (l_2, \mu)]$) $= \mathcal{L}'_1 :: \mathcal{L}'_2 ::$ (p, $[(l', 0), (l'_2, \mu')]$).

Given (J1), (K1), (C1), (M1), and (T1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao2])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao1])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_2,\mu)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wao2])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_3,\ \mathrm{acc},\ \mathrm{skip})\ \parallel C_2)$. We use Axiom 5.1 to prove that $\mathrm{encrypt}(n) = \mathrm{encrypt}(n')$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{(\mathrm{p},[rea])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n_0, \ldots, n_{\alpha-1}]) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{(\mathrm{p},[rea])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n_0, \ldots, n_{\alpha-1}]) \parallel C)$ by SMC^2 rule Read Entire Array we have (B) $\gamma(x) = (l, a\ \mathrm{const}\ bty*)$, (C) $\sigma(l) = (\omega, a\ \mathrm{const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a\ \mathrm{const}\ bty*,\ a,\ 1))$, (D) $\mathrm{DecodePtr}(a\ \mathrm{const}\ bty*,\ 1,\ \omega) = [1,\ [(l_1,\ 0)],\ [1],\ 1]$, (E) $\sigma(l_1) = (\omega_1,\ a\ bty,\ \alpha,\ \mathrm{PermL}(\mathrm{Freeable}, a\ bty, a, \alpha))$, and (F) $\forall i \in \{0 \ldots \alpha - 1\}\quad \mathrm{DecodeArr}(a\ bty, i, \omega_1) = n_i$.

Given (G) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x) \parallel C) \Downarrow^{(\mathrm{p},[(l',0),(l'_1,0),\ldots,(l'_1,\alpha'-1)])}_{(\mathrm{p},[d])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [n'_0, \ldots, n'_{\alpha'-1}]) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = rea$.

Given (G) and (H), by SMC^2 rule Read Entire Array we have (I) $\gamma(x) = (l', a'\ \mathrm{const}\ bty'*)$, (J) $\sigma(l') = (\omega, a'\ \mathrm{const}\ bty'*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a'\ \mathrm{const}\ bty'*, a', 1))$, (K) $\mathrm{DecodePtr}(a'\ \mathrm{const}\ bty'*,\ 1,\ \omega') = [1,\ [(l'_1, 0)],\ [1],\ 1]$, (L) $\sigma(l'_1) = (\omega'_1, a'\ bty', \alpha', \mathrm{PermL}(\mathrm{Freeable}, a'\ bty', a', \alpha'))$, and (M) $\forall i' \in \{0 \ldots \alpha' - 1\}$ $\mathrm{DecodeArr}(a'\ bty', i', \omega'_1) = n'_{i'}$.

Given (B) and (I), by Definition 5.3 we have (N) $l = l'$, (O) $a = a'$ and (P) $bty = bty'$.

Given (C), (J), and (N), by Definition 5.4 we have (Q) $\omega = \omega'$.

Given (D), (K), (O), (P) and (Q), by Lemma 5.26 we have $[1,\ [(l_1, 0)],\ [1],\ 1] = [1,\ [(l'_1, 0)],\ [1],\ 1]$ and therefore (R) $l_1 = l'_1$.

Given (E), (L), and (R), by Definition 5.4 we have (S) $\omega_1 = \omega'_1$ and (T) $\alpha = \alpha'$.

Given (T), we have (U) $i = i'$ such that $i \in \{0 \ldots \alpha - 1\}$.

Given (O), (P), (U), and (S), by Lemma 5.27 we have (V) $\forall i, i' \in \{0 \ldots \alpha - 1\}$ such that $i = i'$, $n_i = n'_{i'}$. Therefore, we have (W) $[n_0, \ldots, n_{\alpha-1}] = [n'_0, \ldots, n'_{\alpha'-1}]$.

Given (C) and (D) by Lemma 5.62 we have accessed location (X) (p, $[(l, 0)]$). Given (E) and (F), by Lemma 5.63

we have accessed locations (Y) (p, $[(l_1, 0), ..., (l_1, \alpha - 1)]$). Given (X) and (Y), by Lemmas 5.44 and 5.45 we have (Z) (p, $[(l, 0), (l_1, 0), ..., (l_1, \alpha - 1)]$).

Given (J) and (K) by Lemma 5.62 we have accessed location (A1) (p, $[(l', 0)]$). Given (L) and (M), by Lemma 5.63 we have accessed locations (B1) (p, $[(l_1', 0), ..., (l_1', \alpha' - 1)]$). Given (A1) and (B1), by Lemmas 5.44 and 5.45 we have (C1) (p, $[(l', 0), (l_1', 0), ..., (l_1', \alpha' - 1)]$).

Given (N), (T), (R), (Z), and (C1), we have (D1) (p, $[(l, 0), (l_1, 0), ..., (l_1, \alpha-1)]$) = (p, $[(l', 0), (l_1', 0), ..., (l_1', \alpha'-1)]$).

Given (D1), (W), and (H), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wea])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wea])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,0),...,(l_1,\alpha-1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Public Array Write Entire Array we have $\alpha_e = \alpha$, acc $= 0$, $(e) \nvdash \gamma$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, [n_0, ..., n_{\alpha_e-1}]) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{public const}\ bty*)$, (D) $\sigma_1(l) = (\omega,\ \text{public const}\ bty*, 1,$ PermL_Ptr(Freeable, public const $bty*$, public, 1)), (E) DecodePtr(public const $bty*,\ 1,\ \omega) = [1, [(l_1, 0)], [1], 1]$, (F) $\sigma_1(l_1) = (\omega_1, \text{public}\ bty, \alpha, \text{PermL}(\text{Freeable}, \text{public}\ bty, \text{public}, \alpha))$, and (G) $\forall i \in \{0...\alpha-1\}$ UpdateArr$(\sigma_{1+i}, (l_1, i),\ n_i,\ \text{public}\ bty) = \sigma_{2+i}$.

Given (H) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1'::(\text{p},[d])}^{\mathcal{L}_1'::(\text{p},[(l',0),(l_1',0),...,(l_1',\alpha'-1)])} ((\text{p}, \gamma,\ \sigma_{2+\alpha'-1}',\ \Delta_1',\ \text{acc},\ \text{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (I) $d = wea$.

Given (H) and (I), by SMC^2 rule Public Array Write Entire Array we have $\alpha_e' = \alpha'$, acc $= 0$, $(e) \nvdash \gamma$, (J) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \text{acc},\ [n_0', ..., n_{\alpha_e'-1}']) \parallel C_1')$, (K) $\gamma(x) = (l',\ \text{public const}\ bty'*)$, (L) $\sigma_1'(l') = (\omega',\ \text{public const}\ bty'*, 1,$ PermL_Ptr(Freeable, public const $bty'*$, public, 1)), (M) DecodePtr(public const $bty'*,\ 1,\ \omega') = [1,\ [(l_1', 0)],\ [1],\ 1]$, (N) $\sigma_1'(l_1') = (\omega_1', \text{public}\ bty', \alpha', \text{PermL}(\text{Freeable}, \text{public}\ bty', \text{public}, \alpha'))$, and (O) $\forall i' \in \{0...\alpha' - 1\}$UpdateArr$(\sigma_{1+i'}',\ (l_1', i'),\ n_i',\ \text{public}\ bty) = \sigma_{2+i'}'$.

Given (B) and (J), by the inductive hypothesis we have (P) $\sigma_1 = \sigma_1'$, (Q) $\Delta_1 = \Delta_1'$, (R) $[n_0, ..., n_{\alpha_e-1}] = [n_0', ..., n_{\alpha_e'-1}']$ and therefore (S) $\alpha_e = \alpha_e'$, (T) $\mathcal{D}_1 = \mathcal{D}_1'$, (U) $\mathcal{L}_1 = \mathcal{L}_1'$, and (V) $C_1 = C_1'$.

Given (C) and (K), by Definition 5.3 we have (W) $l = l'$ and (X) $bty = bty'$.

Given (D), (L), (P), and (W), by Definition 5.4 we have (Y) $\omega = \omega'$.

Given (E), (M), (X), and (Y), by Lemma 5.26 we have $[1,\ [(l_1, 0)],\ [1],\ 1] = [1,\ [(l_1', 0)],\ [1],\ 1]$ and therefore (Z) $l_1 = l_1'$.

Given (F), (N), (P), and (Z), by Definition 5.4 we have (A1) $\omega_1 = \omega_1'$ and (B1) $\alpha = \alpha'$.

Given (B1), (S), $\alpha_e = \alpha$, and $\alpha_e' = \alpha'$, we have (C1) $i = i'$ such that $i \in \{0...\alpha - 1\}$.

Given (G), (O), (B1), (C1), (P), (Z), (X), (S), $\alpha_e = \alpha$, $\alpha_e' = \alpha'$, and (R), by Lemma 5.35 we have (D1) $\forall i, i' \in \{0...\alpha - 1\}$ such that $i = i'$, $\sigma_{1+i} = \sigma_{1+i'}'$ and (E1) $\sigma_{2+i} = \sigma_{2+i'}'$.

Given (D) and (E) by Lemma 5.62 we have accessed location (F1) (p, $[(l, 0)]$). Given (G), by Lemma 5.67 we have accessed locations (G1) (p, $[(l_1, 0), ..., (l_1, \alpha - 1)]$). Given (F1) and (G1), by Lemmas 5.44 and 5.45 we have (H1) (p, $[(l, 0), (l_1, 0), ..., (l_1, \alpha - 1)]$).

Given (L) and (M) by Lemma 5.62 we have accessed location (I1) (p, $[(l', 0)]$). Given (O), by Lemma 5.67 we have accessed locations (J1) (p, $[(l_1', 0), ..., (l_1', \alpha' - 1)]$). Given (I1) and (J1), by Lemmas 5.44 and 5.45 we have (K1) (p, $[(l', 0), (l_1', 0), ..., (l_1', \alpha' - 1)]$).

Given (U), (W), (Z), (B1), (H1), and (K1), by Lemma 5.47 we have (L1) $\mathcal{L}_1$ :: (p, $[(l, 0), (l_1, 0), ..., (l_1, \alpha - 1)]$) $= \mathcal{L}_1'$ :: (p, $[(l', 0), (l_1', 0), ..., (l_1', \alpha' - 1)]$).

Given (T) and (I), by Lemma 5.38 we have (M1) $\mathcal{D}_1$ :: (p, $[wea]$) $= \mathcal{D}_1'$ :: (p, $[wea]$).

Given (E1), (V), (L1), (M1), and (V), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea1])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea2])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0),(l_1,0),\ldots,(l_1,\alpha-1)])}_{\mathcal{D}_1::(\mathrm{p},[wea])} ((\mathrm{p}, \gamma,\ \sigma_{2+\alpha-1},\ \Delta_1,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$. We use Axiom 5.1 to prove that $\mathrm{encrypt}(n) = \mathrm{encrypt}(n')$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,i)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wa])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1::\mathcal{L}_2::(\mathrm{p},[(l,0),(l_1,i)])}_{\mathcal{D}_1::\mathcal{D}_2::(\mathrm{p},[wa])} ((\mathrm{p}, \gamma,\ \sigma_3,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2)$ by SMC^2 rule Public Array Write Public Value Public Index we have $(e_1, e_2) \nvdash \gamma$, $0 \leq i \leq \alpha - 1$, $\mathrm{acc} = 0$ (B) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ i) \parallel C_1)$, (C) $((\mathrm{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1) \Downarrow^{\mathcal{L}_2}_{\mathcal{D}_2} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ n) \parallel C_2)$, (D) $\gamma(x) = (l,\ \mathrm{public\ const}\ bty*)$, (E) $\sigma_2(l) = (\omega, \mathrm{public\ const}\ bty*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public\ const}\ bty*, \mathrm{public}, 1))$, (F) $\mathrm{DecodePtr}(\mathrm{public\ const}\ bty*, 1, \omega) = [1,\ [(l_1, 0)],\ [1],\ 1]$, (G) $\sigma_2(l_1) = (\omega_1, \mathrm{public}\ bty, \alpha, \mathrm{PermL}(\mathrm{Freeable}, \mathrm{public}\ bty, \mathrm{public}, \alpha))$, and (H) $\mathrm{UpdateArr}(\sigma_2,\ (l_1, i),\ n,\ \mathrm{public}\ bty) = \sigma_3$.

Given (I) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ x[e_1]\ = e_2) \parallel C) \Downarrow^{\mathcal{L}_1'::\mathcal{L}_2'::(\mathrm{p},[(l',0),(l_1',i')])}_{\mathcal{D}_1'::\mathcal{D}_2'::(\mathrm{p},[d])} ((\mathrm{p}, \gamma,\ \sigma_3',\ \Delta_2',\ \mathrm{acc},\ \mathrm{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (J) $d = wa$.

Given (I) and (J), by SMC^2 rule Public Array Write Public Value Public Index we have $(e_1, e_2) \nvdash \gamma$, $0 \leq i' \leq \alpha' - 1$, (K) $((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ e_1) \parallel C) \Downarrow^{\mathcal{L}_1'}_{\mathcal{D}_1'} ((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1',\ \mathrm{acc},\ i') \parallel C_1')$, (L) $((\mathrm{p}, \gamma,\ \sigma_1',\ \Delta_1,\ \mathrm{acc},\ e_2) \parallel C_1') \Downarrow^{\mathcal{L}_2'}_{\mathcal{D}_2'} ((\mathrm{p}, \gamma,\ \sigma_2',\ \Delta_2',\ \mathrm{acc},\ n') \parallel C_2')$, (M) $\gamma(x) = (l',\ \mathrm{public\ const}\ bty'*)$, (N) $\sigma_2'(l') = (\omega', \mathrm{public\ const}\ bty'*, 1, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{public\ const}\ bty'*, \mathrm{public}, 1))$, (O) $\mathrm{DecodePtr}(\mathrm{public\ const}\ bty'*, 1, \omega') = [1,\ [(l_1', 0)],\ [1],$

1], (P) $\sigma_2'(l_1') = (\omega_1', \text{public } bty', \alpha', \text{PermL}(\text{Freeable}, \text{public } bty', \text{public}, \alpha'))$, and (Q) $\text{UpdateArr}(\sigma_2', (l_1', i'), n', \text{public } bty') = \sigma_3'$.

Given (B) and (K), by the inductive hypothesis we have (R) $\sigma_1 = \sigma_1'$, (S) $\Delta_1 = \Delta_1'$, (T) $i = i'$, (U) $\mathcal{D}_1 = \mathcal{D}_1'$, (V) $\mathcal{L}_1 = \mathcal{L}_1'$, and (W) $C_1 = C_1'$.

Given (C), (L), (R), (S), and (W), by the inductive hypothesis we have (X) $\sigma_2 = \sigma_2'$, (Y) $\Delta_2 = \Delta_2'$, (Z) $n = n'$, (A1) $\mathcal{D}_2 = \mathcal{D}_2'$, (B1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (C1) $C_2 = C_2'$.

Given (D) and (M), by Definition 5.3 we have (D1) $l = l'$ and (E1) $bty = bty'$.

Given (E), (N), (X), and (D1), by Definition 5.4 we have (F1) $\omega = \omega'$.

Given (F), (O), (E1), and (F1), by Lemma 5.26 we have (G1) $l_1 = l_1'$.

Given (G), (P), (X), and (G1), by Definition 5.4 we have (H1) $\omega_1 = \omega_1'$ and (I1) $\alpha = \alpha'$.

Given (H), (Q), (X), (G1), (T), (Z), and (E1), by Lemma 5.35 we have (J1) $\sigma_3 = \sigma_3'$.

Given (E) and (F) by Lemma 5.62 we have accessed location (K1) $(\text{p}, [(l, 0)])$. Given (H), by Lemma 5.67 we have accessed location (L1) $(\text{p}, [(l_1, i)])$. Given (K1) and (L1), by Lemmas 5.44 and 5.45 we have (M1) $(\text{p}, [(l, 0), (l_1, i)])$.

Given (N) and (O) by Lemma 5.62 we have accessed location (N1) $(\text{p}, [(l', 0)])$. Given (Q), by Lemma 5.67 we have accessed location (O1) $(\text{p}, [(l_1', i')])$. Given (N1) and (O1), by Lemmas 5.44 and 5.45 we have (P1) $(\text{p}, [(l', 0), (l_1', i')])$.

Given (V), (B1), (M1), (P1), (D1), (G1), and (T), by Lemma 5.47 we have (Q1) $\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_1, i)]) = \mathcal{L}_1' :: \mathcal{L}_2' :: (\text{p}, [(l', 0), (l_1', i')])$.

Given (U), (A1), and $(\text{p}, [wa])$, by Lemma 5.38 we have (R1) $\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [wa]) = \mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [wa])$.

Given (Y), (J1), (C1), (Q1), and (R1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [wa2])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_1, i)])} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_2)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1 :: \mathcal{D}_2 :: (\text{p}, [wa2])}^{\mathcal{L}_1 :: \mathcal{L}_2 :: (\text{p}, [(l, 0), (l_1, i)])} ((\text{p}, \gamma, \sigma_3, \Delta_3, \text{acc}, \text{skip}) \parallel C_2)$ by SMC^2 rule Private Array Write Private Value Public Index we have $(e_1) \nvdash \gamma$, $(e_2) \vdash \gamma$, $0 \leq i \leq \alpha - 1$, (B) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, i) \parallel C_1)$, (C) $((\text{p}, \gamma, \sigma_1, \Delta_1, \text{acc}, e_2) \parallel C_1) \Downarrow_{\mathcal{D}_2}^{\mathcal{L}_2} ((\text{p}, \gamma, \sigma_2, \Delta_2, \text{acc}, n) \parallel C_2)$, (D) $\gamma(x) = (l, \text{private const } bty*)$, (E) $\sigma_2(l) = (\omega, \text{private const } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private const } bty*, \text{private}, 1))$, (F) $\text{DecodePtr}(\text{private const } bty*, 1, \omega) = [1, [(l_1, 0)], [1], 1]$, (G) $\sigma_2(l_1) = (\omega_1, \text{private } bty, \alpha, \text{PermL}(\text{Freeable}, \text{private } bty, \text{private}, \alpha))$, (H) $\text{DynamicUpdate}(\Delta_2, \sigma_2, [(l_1, i)], \text{acc}, \text{private } bty) = \Delta_3$, and (I) $\text{UpdateArr}(\sigma_2, (l_1, i), n, \text{private } bty) = \sigma_3$.

Given (J) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x[e_1] = e_2) \parallel C) \Downarrow_{\mathcal{D}_1' :: \mathcal{D}_2' :: (\text{p}, [d])}^{\mathcal{L}_1' :: \mathcal{L}_2' :: (\text{p}, [(l', 0), (l_1', i')])} ((\text{p}, \gamma, \sigma_3', \Delta_3', \text{acc}, \text{skip}) \parallel C_2')$ and (A), by Lemma 4.87 we have (K) $d = wa2$.

Given (J) and (K), by SMC^2 rule Private Array Write Private Value Public Index we have $(e_1) \nvdash \gamma$, $(e_2) \vdash \gamma$, $0 \leq i' \leq \alpha' - 1$, (L) $((\text{p}, \gamma, \sigma, \Delta, \text{acc}, e_1) \parallel C) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, i') \parallel C_1')$, (M) $((\text{p}, \gamma, \sigma_1', \Delta_1', \text{acc}, e_2) \parallel C_1')$

$\Downarrow_{\mathcal{D}_2'}^{\mathcal{L}_2'}$ ((p, γ, σ_2', Δ_2', acc, n') $\|$ C_2'), (N) $\gamma(x) = (l'$, private const $bty'*$), (O) $\sigma_2'(l') = (\omega'$, private const $bty'*$, 1, PermL_Ptr(Freeable, private const $bty'*$, private, 1)), (P) DecodePtr(private const $bty'*, 1, \omega'$) = [1, $[(l_1', 0)]$, [1], 1], (Q) $\sigma_2'(l_1') = (\omega_1'$, private bty', α', PermL(Freeable, private bty', private, α')), (R) DynamicUpdate(Δ_2', σ_2', $[(l_1', i')]$, acc, private bty') = Δ_3', and (S) UpdateArr(σ_2', (l_1', i'), n', private bty') = σ_3'.

Given (B) and (L), by the inductive hypothesis we have (T) $\sigma_1 = \sigma_1'$, (U) $\Delta_1 = \Delta_1'$, (V) $i = i'$, (W) $\mathcal{D}_1 = \mathcal{D}_1'$, (X) $\mathcal{L}_1 = \mathcal{L}_1'$, and (Y) $C_1 = C_1'$.

Given (C), (M), (T), (U), and (Y), by the inductive hypothesis we have (Z) $\sigma_2 = \sigma_2'$, (A1) $\Delta_2 = \Delta_2'$, (B1) $n = n'$, (C1) $\mathcal{D}_2 = \mathcal{D}_2'$, (D1) $\mathcal{L}_2 = \mathcal{L}_2'$, and (E1) $C_2 = C_2'$.

Given (D) and (N), by Definition 5.3 we have (F1) $l = l'$ and (G1) $bty = bty'$.

Given (E), (O), (Z), and (F1), by Definition 5.4 we have (H1) $\omega = \omega'$.

Given (F), (P), (G1), and (H1), by Lemma 5.26 we have (I1) $l_1 = l_1'$.

Given (G), (Q), (Z), and (I1), by Definition 5.4 we have (J1) $\omega_1 = \omega_1'$ and (K1) $\alpha = \alpha'$.

Given (H), (R), (A1), (Z), (I1), (V), and (G1), by Lemma 5.25 we have (L1) $\Delta_3 = \Delta_3'$.

Given (I), (S), (Z), (I1), (V), (B1), and (G1), by Lemma 5.35 we have (M1) $\sigma_3 = \sigma_3'$.

Given (E) and (F) by Lemma 5.62 we have accessed location (N1) (p, $[(l, 0)]$). Given (I), by Lemma 5.67 we have accessed location (O1) (p, $[(l_1, i)]$). Given (N1) and (O1), by Lemmas 5.44 and 5.45 we have (P1) (p, $[(l, 0), (l_1, i)]$).

Given (O) and (P) by Lemma 5.62 we have accessed location (Q1) (p, $[(l', 0)]$). Given (S), by Lemma 5.67 we have accessed location (R1) (p, $[(l_1', i')]$). Given (Q1) and (R1), by Lemmas 5.44 and 5.45 we have (S1) (p, $[(l', 0), (l_1', i')]$).

Given (X), (D1), (P1), (S1), (F1), (I1), and (V), by Lemma 5.47 we have (T1) $\mathcal{L}_1 :: \mathcal{L}_2 ::$ (p, $[(l, 0), (l_1, i)]$) $= \mathcal{L}_1' :: \mathcal{L}_2' ::$ (p, $[(l', 0), (l_1', i')]$).

Given (Y), (E1), and (K), by Lemma 5.38 we have (U1) $\mathcal{D}_1 :: \mathcal{D}_2 ::$ (p, $[wa2]$) $= \mathcal{D}_1' :: \mathcal{D}_2' ::$ (p, $[wa2]$).

Given (E1), (L1), (M1), (T1), and (U1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, $x[e_1] = e_2$) $\|$ C) $\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[wa1])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_1,i)])}$ ((p, γ, σ_3, Δ_2, acc, skip) $\|$ C_2)

This case is similar to Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, $x[e_1] = e_2$) $\|$ C) $\Downarrow_{\mathcal{D}_1::\mathcal{D}_2::(\text{p},[wa2])}^{\mathcal{L}_1::\mathcal{L}_2::(\text{p},[(l,0),(l_1,i)])}$ ((p, γ, σ_3, Δ_2, acc, skip) $\|$ C_2). We use Axiom 5.1 to prove that encrypt(n) = encrypt(n').

Case $\Pi \triangleright$ ((p, γ, σ, Δ, acc, $ty\ x$) $\|$ C) $\Downarrow_{(\text{p},[dp])}^{(\text{p},[(l,0)])}$ ((p, γ_1, σ_1, Δ, acc, skip) $\|$ C)

Given (A) $\Pi \triangleright$ ((p, γ, σ, Δ, acc, $ty\ x$) $\|$ C) $\Downarrow_{(\text{p},[dp])}^{(\text{p},[(l,0)])}$ ((p, γ_1, σ_1, Δ, acc, skip) $\|$ C) by SMC^2 rule Public Pointer Declaration we have (ty = public $bty*$) $\vee$ (($ty = bty*$) $\wedge$ ((bty = char) $\vee$ (bty = void))), acc = 0, (B)

$l = \phi()$, (C) $\text{GetIndirection}(*) = i$, (D) $\omega = \text{EncodePtr}(\text{public } bty*, [1, [(l_{default}, 0)], [1], i])$, (E) $\gamma_1 = \gamma[x \to (l, \text{ public } bty*)]$, and (F) $\sigma_1 = \sigma[l \to (\omega, \text{ public } bty*, 1, \text{ PermL\_Ptr}(\text{Freeable}, \text{public } bty*, \text{public}, 1))]$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma_1', \sigma_1', \Delta, \text{acc}, \text{skip}) \parallel C)$ and (A), by Lemma 4.87 we have (H) $d = dp$.

Given (G) and (H), by SMC^2 rule Public Pointer Declaration we have $(ty = \text{public } bty*) \vee ((ty = bty*) \wedge ((bty = \text{char}) \vee (bty = \text{void})))$, $\text{acc} = 0$, (I) $l' = \phi()$, (J) $\text{GetIndirection}(*) = i'$, (K) $\omega' = \text{EncodePtr}(\text{public } bty*, [1, [(l_{default}, 0)], [1], i'])$, (L) $\gamma_1' = \gamma[x \to (l', \text{ public } bty*)]$, and (M) $\sigma_1' = \sigma[l' \to (\omega', \text{ public } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{public } bty*, \text{public}, 1))]$.

Given (B) and (I), by Axiom 5.4 we have (N) $l = l'$.

Given (C) and (J), by Lemma 5.13 we have (O) $i = i'$.

Given (D), (K), and (O), by Lemma 5.32 we have (P) $\omega = \omega'$.

Given (E), (L), and (N), by Definition 5.3 we have (Q) $\gamma_1 = \gamma_1'$.

Given (F), (M), (N), and (P), by Definition 5.4 we have (R) $\sigma_1 = \sigma_1'$.

Given (A), (G), and (H), we have (S) $(\text{p}, [dp]) = (\text{p}, [dp])$.

Given (F), by Lemma 5.51 we have accessed (T) $(\text{p}, [(l, 0)])$. Given (M), by Lemma 5.51 we have accessed (U) $(\text{p}, [(l', 0)])$. Given (T), (U), and (N), we have (V) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (Q), (R), (S), and (V), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow_{(\text{p},[dp1])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, ty\ x) \parallel C) \Downarrow_{(\text{p},[dp])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma_1, \sigma_1, \Delta, \text{acc}, \text{skip}) \parallel C)$

Case $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x) \parallel C) \Downarrow_{(\text{p},[rp])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_1, \mu_1)) \parallel C)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, x) \parallel C) \Downarrow_{(\text{p},[rp])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma, \sigma, \Delta, \text{acc}, (l_1, \mu_1)) \parallel C)$ by SMC^2 rule Pointer Read

Single Location we have (B) $\gamma(x) = (l,\ a\ bty*)$, (C) $\sigma(l) = (\omega,\ a\ bty*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a\ bty*, a, 1))$, and (D) $\text{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ i]$.

Given (E) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[d])}^{(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (l'_1, \mu'_1)) \parallel C)$ and (A), by Lemma 4.87 we have (F) $d = rp$.

Given (E) and (F), by SMC^2 rule Pointer Read Single Location we have (G) $\gamma(x) = (l',\ a'\ bty'*)$, (H) $\sigma(l') = (\omega',\ a'\ bty'*,\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, a'\ bty'*, a', 1))$, and (I) $\text{DecodePtr}(a'\ bty'*,\ 1,\ \omega') = [1,\ [(l'_1, \mu'_1)],\ [1],\ i']$.

Given (B) and (G), by Definition 5.3 we have (J) $l = l'$, (K) $a =$, and (L) $bty = bty'$.

Given (C), (H), and (J), by Definition 5.4 we have (M) $\omega = \omega'$.

Given (D), (I), (K), (L), and (M), by Lemma 5.26 we have $[1,\ [(l_1, \mu_1)],\ [1],\ i] = [1,\ [(l'_1, \mu'_1)],\ [1],\ i']$ and therefore (N) $(l_1, \mu_1) = (l'_1, \mu'_1)$.

Given (C) and (D), by Lemma 5.62 we have accessed location (O) $(\text{p}, [(l, 0)])$.

Given (H) and (I), by Lemma 5.62 we have accessed location (P) $(\text{p}, [(l', 0)])$.

Given (O), (P), and (J), we have (Q) $(\text{p}, [(l, 0)]) = (\text{p}, [(l', 0)])$.

Given (F), (N), and (Q), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[rp1])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (l_1, \mu_1)) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x) \parallel C) \Downarrow_{(\text{p},[rp])}^{(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ (l_1, \mu_1)) \parallel C)$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wp1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wp1])}^{\mathcal{L}_1::(\text{p},[(l,0)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Write we have $(e) \nvDash \gamma$, (B) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma,\ \sigma_1,\ \Delta_1,\ \text{acc},\ (l_e, \mu_e)) \parallel C_1)$, (C) $\gamma(x) = (l,\ \text{private}\ bty*)$, (D) $\sigma_1(l) = (\omega,\ \text{private}\ bty*,\ \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private}\ bty*, \text{private}, \alpha))$, (E) $\text{DecodePtr}(\text{private}\ bty*,\ \alpha,\ \omega) = [\alpha,\ L,\ J,\ i]$, and (F) $\text{UpdatePtr}(\sigma_1,\ (l, 0),\ [1,\ [(l_e, \mu_e)],\ [1],\ i], \text{private}\ bty*) = (\sigma_2,\ 1)$.

Given (G) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ x = e) \parallel C) \Downarrow_{\mathcal{D}'_1::(\text{p},[d])}^{\mathcal{L}'_1::(\text{p},[(l',0)])} ((\text{p}, \gamma,\ \sigma'_2,\ \Delta'_1,\ \text{acc},\ \text{skip}) \parallel C'_1)$ and (A), by Lemma 4.87 we have (H) $d = wp1$.

Given (G) and (H), by SMC^2 rule Private Pointer Write we have $(e) \nvDash \gamma$, (I) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}'_1}^{\mathcal{L}'_1} ((\text{p}, \gamma,\ \sigma'_1,\ \Delta'_1,\ \text{acc},\ (l'_e, \mu'_e)) \parallel C'_1)$, (J) $\gamma(x) = (l',\ \text{private}\ bty'*)$, (K) $\sigma'_1(l') = (\omega',\ \text{private}\ bty'*,\ \alpha',$

PermL_Ptr(Freeable, private $bty'*$, private, α')), (L) DecodePtr(private $bty'*$, α', ω') = $[\alpha', L', J', i']$, and (M) UpdatePtr(σ_1', $(l', 0)$, $[1, [(l_e', \mu_e')], [1], i']$, private $bty'*$) = $(\sigma_2', 1)$.

Given (B) and (I), by the inductive hypothesis we have (N) $\sigma_1 = \sigma_1'$, (O) $\Delta_1 = \Delta_1'$, (P) $(l_e, \mu_e) = (l_e', \mu_e')$, (Q) $\mathcal{D}_1 = \mathcal{D}_1'$, (R) $\mathcal{L}_1 = \mathcal{L}_1'$, and (S) $C_1 = C_1'$.

Given (C) and (J), by Definition 5.3 we have (T) $l = l'$ and (U) $bty = bty'$.

Given (D), (K), (N), and (T), by Definition 5.4 we have (V) $\omega = \omega'$ and (W) $\alpha = \alpha'$.

Given (E), (L), (U), (W), and (V), by Lemma 5.26 we have $L = L'$, $J = J'$, and (X) $i = i'$.

Given (F), (M), (N), (T), (P), (X), and (V), by Lemma 5.36 we have (Y) $\sigma_2 = \sigma_2'$.

Given (Q) and (H), by Lemma 5.38 we have (Z) $\mathcal{D}_1 :: (\mathrm{p}, [wp1]) = \mathcal{D}_1' :: (\mathrm{p}, [wp1])$.

Given (D) and (E), by Lemma 5.62 we have accessed location (A1) $(\mathrm{p}, [(l, 0)])$. Given (F), by Lemma 5.68 we have accessed location (B1) $(\mathrm{p}, [(l, 0)])$.

Given (K) and (L), by Lemma 5.62 we have accessed location (C1) $(\mathrm{p}, [(l', 0)])$. Given (M), by Lemma 5.68 we have accessed location (D1) $(\mathrm{p}, [(l', 0)])$

Given (R), (A1), (B1), (C1), (D1), and (T), by Lemma 5.47 we have (E1) $\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)]) = \mathcal{L}_1' :: (\mathrm{p}, [(l', 0)])$.

Given (Y), (O), (S), (Z), and (E1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wp])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wp1])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wp2])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wp1])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_1, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wdp3])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1 :: (\mathrm{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)])}_{\mathcal{D}_1 :: (\mathrm{p}, [wdp3])} ((\mathrm{p}, \gamma, \sigma_2, \Delta_2, \mathrm{acc}, \mathrm{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Dereference Write Single Location Private Value we have $(e) \vdash \gamma$, $(bty = \mathrm{int}) \vee (bty = \mathrm{float})$, (B) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow^{\mathcal{L}_1}_{\mathcal{D}_1} ((\mathrm{p}, \gamma, \sigma_1, \Delta_1, \mathrm{acc}, n) \parallel C_1)$, (C) $\gamma(x) = (l, \text{ private } bty*)$, (D) $\sigma_1(l) = (\omega, \text{ private } bty*, 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty*, \text{private}, 1))$, (E) DecodePtr(private $bty*$, 1, ω) = $[1, [(l_1, \mu_1)], [1], 1]$, (F) DynamicUpdate($\Delta_1, \sigma_1, [(l_1, \mu_1)]$, acc, private bty) = (Δ_2, L_1), and (G) UpdateOffset(σ_1, (l_1, μ_1), n, private bty) = $(\sigma_2, 1)$.

Given (H) $\Sigma \triangleright ((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, *x = e) \parallel C) \Downarrow^{\mathcal{L}_1' :: (\mathrm{p}, [(l', 0)] :: L_1' :: [(l_1', \mu_1')])}_{\mathcal{D}_1' :: (\mathrm{p}, [d])} ((\mathrm{p}, \gamma, \sigma_2', \Delta_2', \mathrm{acc}, \mathrm{skip}) \parallel C_1')$ and (A), by Lemma 4.87 we have (I) $d = wdp3$.

Given (H) and (I), by SMC^2 rule Private Pointer Dereference Write Single Location Private Value we have $(e) \vdash \gamma$, $(bty' = \mathrm{int}) \vee (bty' = \mathrm{float})$, (J) $((\mathrm{p}, \gamma, \sigma, \Delta, \mathrm{acc}, e) \parallel C) \Downarrow^{\mathcal{L}_1'}_{\mathcal{D}_1'} ((\mathrm{p}, \gamma, \sigma_1', \Delta_1', \mathrm{acc}, n') \parallel C_1')$, (K)

$\gamma(x) = (l', \text{ private } bty'*)$, (L) $\sigma_1'(l') = (\omega', \text{ private } bty'*, \ 1, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty'*, \text{private}, 1))$, (M) DecodePtr(private $bty'*$, 1, ω') $= [1, [(l_1', \mu_1')], [1], 1]$, (N) DynamicUpdate$(\Delta_1', \sigma_1', [(l_1', \mu_1')]$, acc, private $bty') = (\Delta_2', L_1')$, and (O) UpdateOffset$(\sigma_1', (l_1', \mu_1'), n'$, private $bty') = (\sigma_2', 1)$.

Given (B) and (J), by the inductive hypothesis we have (P) $\sigma_1 = \sigma_1'$, (Q) $\Delta_1 = \Delta_1'$, (R) $n = n'$, (S) $\mathcal{D}_1 = \mathcal{D}_1'$, (T) $\mathcal{L}_1 = \mathcal{L}_1'$, and (U) $C_1 = C_1'$.

Given (C) and (K), by Definition 5.3 we have (V) $l = l'$ and (W) $bty = bty'$.

Given (D), (L), (P), and (V), by Definition 5.4 we have (X) $\omega = \omega'$.

Given (E), (M), (W), and (X), by Lemma 5.26 we have (Y) $(l_1, \mu_1) = (l_1', \mu_1')$.

Given (F), (N), (Q), (P), (Y), and (W), by Lemma 5.25 we have (Z) $\Delta_2 = \Delta_2'$ and (A1) $L_1 = L_1'$.

Given (G), (O), (P), (Y), (R), and (W), by Lemma 5.37 we have (B1) $\sigma_2 = \sigma_2'$.

Given (S) and (p, $[wdp3]$), by Lemma 5.38 we have (C1) $\mathcal{D}_1 :: (\text{p}, [wdp3]) = \mathcal{D}_1' :: (\text{p}, [wdp3])$.

Given (D) and (E), by Lemma 5.62 we have accessed location (D1) $(\text{p}, [(l, 0)])$. Given (F), by Lemma 5.61 we have accessed location (E1) (p, L_1). Given (G), by Lemma 5.69 we have accessed location (F1) $(\text{p}, [(l_1, \mu_1)])$. Given (D1), (E1), and (F1), by Lemmas 5.44 and 5.45 we have (G1) $(\text{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)])$.

Given (L) and (M), by Lemma 5.62 we have accessed location (H1) $(\text{p}, [(l', 0)])$. Given (N), by Lemma 5.61 we have accessed location (I1) (p, L_1'). Given (O), by Lemma 5.69 we have accessed location (J1) $(\text{p}, [(l_1', \mu_1')])$. Given (H1), (I1), and (J1), by Lemmas 5.44 and 5.45 we have (K1) $(\text{p}, [(l', 0)] :: L_1' :: [(l_1', \mu_1')])$

Given (T), (G1), (K1), (A1), (V), and (Y), by Lemma 5.47 we have (L1) $\mathcal{L}_1 :: (\text{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)) = \mathcal{L}_1' :: (\text{p}, [(l', 0)] :: L_1' :: [(l_1', \mu_1')] :: L_1')$.

Given (A1), (Z), (U), (B1), and (L1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp4])}^{\mathcal{L}_1 :: (\text{p}, [(l,0)] :: L_1 :: [(l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp3])}^{\mathcal{L}_1 :: (\text{p}, [(l,0)] :: L_1 :: [(l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ \text{skip}) \parallel C_1)$, with the addition of using Axiom 5.1 to prove that encrypt(n) = encrypt(n').

Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp])}^{\mathcal{L}_1 :: (\text{p}, [(l,0), (l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_1, \ \text{acc}, \ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp3])}^{\mathcal{L}_1 :: (\text{p}, [(l,0)] :: L_1 :: [(l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ \text{skip}) \parallel C_1)$, removing the reasoning about DynamicUpdate and its resulting locations, as it is not present in this rule.

Case $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp2])}^{\mathcal{L}_1 :: (\text{p}, [(l,0)] :: L_1 :: [(l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ \text{skip}) \parallel C_1)$

Given (A) $\Pi \triangleright ((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ *x = e) \parallel C) \Downarrow_{\mathcal{D}_1 :: (\text{p}, [wdp2])}^{\mathcal{L}_1 :: (\text{p}, [(l,0)] :: L_1 :: [(l_1, \mu_1)])} ((\text{p}, \gamma, \ \sigma_2, \ \Delta_2, \ \text{acc}, \ \text{skip}) \parallel C_1)$ by SMC^2 rule Private Pointer Dereference Write Multiple Locations to Single Location Higher Level Indirection we have $(e) \vdash \gamma, (bty = \text{int}) \vee (bty = \text{float})$, (B) $((\text{p}, \gamma, \ \sigma, \ \Delta, \ \text{acc}, \ e) \parallel C) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((\text{p}, \gamma, \ \sigma_1, \ \Delta_1, \ \text{acc}, [\alpha, L_e, J_e, i-1]) \parallel C_1)$,

(C) $\gamma(x) = (l,\ \text{private } bty{*})$, (D) $\sigma_1(l) = (\omega,\ \text{private } bty{*},\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty{*}, \text{private}, 1))$, (E) $\text{DecodePtr}(\text{private } bty{*},\ 1,\ \omega) = [1,\ [(l_1, \mu_1)],\ [1],\ 1]$, (F) $\text{DynamicUpdate}(\Delta_1, \sigma_1, [(l_1, \mu_1)],\ \text{acc}, \text{private } bty{*}) = (\Delta_2, L_1)$, and (G) $\text{UpdatePtr}(\sigma_1,\ (l_1, \mu_1),\ [\alpha, L_e, J_e, i-1], \text{private } bty{*}) = (\sigma_2, 1)$.

Given (H) $\Sigma \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {*x} = e) \parallel C) \Downarrow_{\mathcal{D}'_1::(\text{p},[d])}^{\mathcal{L}'_1::(\text{p},[(l',0)]::L'_1::[(l'_1,\mu'_1)])} ((\text{p}, \gamma,\ \sigma'_2,\ \Delta'_2,\ \text{acc},\ \text{skip}) \parallel C'_1)$ and (A), by Lemma 4.87 we have (I) $d = wdp2$.

Given (H) and (I), by SMC^2 rule Private Pointer Dereference Write Multiple Locations to Single Location Higher Level Indirection we have $(e) \vdash \gamma, (bty' = \text{int}) \vee (bty' = \text{float})$, (J) $((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ e) \parallel C) \Downarrow_{\mathcal{D}'_1}^{\mathcal{L}'_1} ((\text{p}, \gamma,\ \sigma'_1,\ \Delta'_1,\ \text{acc},\ [\alpha', L'_e, J'_e, i'-1]) \parallel C'_1)$, (K) $\gamma(x) = (l',\ \text{private } bty'{*})$, (L) $\sigma'_1(l') = (\omega',\ \text{private } bty'{*},\ 1,\ \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty'{*}, \text{private}, 1))$, (M) $\text{DecodePtr}(\text{private } bty'{*},\ 1,\ \omega') = [1,\ [(l'_1, \mu'_1)],\ [1],\ 1]$, (N) $\text{DynamicUpdate}(\Delta'_1, \sigma'_1, [(l'_1, \mu'_1)],\ \text{acc}, \text{private } bty'{*}) = (\Delta'_2, L'_1)$, and (O) $\text{UpdatePtr}(\sigma'_1,\ (l'_1, \mu'_1),\ [\alpha', L'_e, J'_e,\ i'-1], \text{private } bty'{*}) = (\sigma'_2, 1)$.

Given (B) and (J), by the inductive hypothesis we have (P) $\sigma_1 = \sigma'_1$, (Q) $\Delta_1 = \Delta'_1$, (R) $[\alpha, L_e, J_e, i-1] = [\alpha', L'_e, J'_e, i'-1]$, (S) $\mathcal{D}_1 = \mathcal{D}'_1$, (T) $\mathcal{L}_1 = \mathcal{L}'_1$, and (U) $C_1 = C'_1$.

Given (C) and (K), by Definition 5.3 we have (V) $l = l'$ and (W) $bty = bty'$.

Given (D), (L), (P), and (V), by Definition 5.4 we have (X) $\omega = \omega'$.

Given (E), (M), (W), and (X), by Lemma 5.26 we have (Y) $(l_1, \mu_1) = (l'_1, \mu'_1)$.

Given (F), (N), (Q), (P), (Y), and (W), by Lemma 5.25 we have (Z) $\Delta_2 = \Delta'_2$ and (A1) $L_1 = L'_1$.

Given (G), (O), (P), (Y), (R), and (W), by Lemma 5.37 we have (B1) $\sigma_2 = \sigma'_2$.

Given (S) and $(\text{p}, [wdp2])$, by Lemma 5.38 we have (C1) $\mathcal{D}_1 :: (\text{p}, [wdp2]) = \mathcal{D}'_1 :: (\text{p}, [wdp2])$.

Given (D) and (E), by Lemma 5.62 we have accessed location (D1) $(\text{p}, [(l, 0)])$. Given (F), by Lemma 5.61 we have accessed location (E1) (p, L_1). Given (G), by Lemma 5.68 we have accessed location (F1) $(\text{p}, [(l_1, \mu_1)])$. Given (D1), (E1), and (F1), by Lemmas 5.44 and 5.45 we have (G1) $(\text{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)])$.

Given (L) and (M), by Lemma 5.62 we have accessed location (H1) $(\text{p}, [(l', 0)])$. Given (N), by Lemma 5.61 we have accessed location (I1) (p, L'_1). Given (O), by Lemma 5.68 we have accessed location (J1) $(\text{p}, [(l'_1, \mu'_1)])$. Given (H1), (I1), and (J1), by Lemmas 5.44 and 5.45 we have (K1) $(\text{p}, [(l', 0)] :: L'_1 :: [(l'_1, \mu'_1)])$

Given (T), (G1), (K1), (A1), (V), and (Y), by Lemma 5.47 we have (L1) $\mathcal{L}_1 :: (\text{p}, [(l, 0)] :: L_1 :: [(l_1, \mu_1)) = \mathcal{L}'_1 :: (\text{p}, [(l', 0)] :: L'_1 :: [(l'_1, \mu'_1)] :: L'_1)$.

Given (A1), (Z), (U), (B1), and (L1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {*x} = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wdp1])}^{\mathcal{L}_1::(\text{p},[(l,0),(l_1,\mu_1)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_1,\ \text{acc},\ \text{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\text{p}, \gamma,\ \sigma,\ \Delta,\ \text{acc},\ {*x} = e) \parallel C) \Downarrow_{\mathcal{D}_1::(\text{p},[wdp2])}^{\mathcal{L}_1::(\text{p},[(l,0)]::L_1::[(l_1,\mu_1)])} ((\text{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \text{acc},$

skip) $\parallel C_1$), removing the reasoning about DynamicUpdate and its resulting locations, as it is not present in this rule.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\mathrm{p},[wdp5])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},\ \mathrm{skip}) \parallel C_1)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x = e) \parallel C) \Downarrow^{\mathcal{L}_1::(\mathrm{p},[(l,0)]::L_1::[(l_1,\mu_1)])}_{\mathcal{D}_1::(\mathrm{p},[wdp2])} ((\mathrm{p}, \gamma,\ \sigma_2,\ \Delta_2,\ \mathrm{acc},$ skip) $\parallel C_1$).

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,\mu_1)])}_{(\mathrm{p},[rdp])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$

Given (A) $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,\mu_1)])}_{(\mathrm{p},[rdp])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$ by SMC^2 rule Pointer Dereference Single Location, we have (B) $\gamma(x) = (l,\ a\ bty*)$, (C) $\sigma(l) = (\omega,\ a\ bty*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a\ bty*, a, 1))$, (D) $\mathrm{DecodePtr}(a\ bty*,\ 1,\ \omega) = [1,\ [(l_1,\mu_1)],\ [1],\ 1]$, and (E) $\mathrm{DerefPtr}(\sigma, a\ bty, (l_1,\mu_1)) = (n, 1)$.

Given (F) $\Sigma \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p},[(l',0),(l_1',\mu_1')])}_{(\mathrm{p},[d])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n') \parallel C)$ and (A), by Lemma 4.87 we have (G) $d = rdp$.

Given (F) and (G), by SMC^2 rule Pointer Dereference Single Location, we have (H) $\gamma(x) = (l',\ a'\ bty'*)$, (I) $\sigma(l') = (\omega',\ a'\ bty'*,\ 1,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, a'\ bty'*, a', 1))$, (J) $\mathrm{DecodePtr}(a'\ bty'*,\ 1,\ \omega') = [1,\ [(l_1',\mu_1')],\ [1],\ 1]$, and (K) $\mathrm{DerefPtr}(\sigma, a'\ bty', (l_1',\mu_1')) = (n', 1)$.

Given (B) and (H), by Definition 5.3 we have (L) $l = l'$ and (M) $a\ bty = a'\ bty'$.

Given (C), (I), and (L), by Definition 5.4 we have (N) $\omega = \omega'$ and (O) $a = a'$.

Given (D), (J), (M), and (N), by Lemma 5.26 we have (P) $(l_1,\mu_1) = (l_1',\mu_1')$.

Given (E), (K), (M), and (P), by Lemma 5.14 we have (Q) $n = n'$.

Given (C) and (D), by Lemma 5.62 we have accessed location (R) $(\mathrm{p}, [(l,0)])$. Given (E), by Lemma 5.70 we have accessed location (S) $(\mathrm{p}, [(l_1,\mu_1)])$. Given (R) and (S), by Lemmas 5.44 and 5.45 we have (T) $(\mathrm{p}, [(l,0),(l_1,\mu_1)])$.

Given (I) and (J), by Lemma 5.62 we have accessed location (U) $(\mathrm{p}, [(l',0)])$. Given (K), by Lemma 5.70 we have accessed location (V) $(\mathrm{p}, [(l_1',\mu_1')])$. Given (U) and (V), by Lemmas 5.44 and 5.45 we have (W) $(\mathrm{p}, [(l,0),(l_1,\mu_1)])$.

Given (T), (W), (L), and (P), we have (X) $(\mathrm{p}, [(l,0),(l_1,\mu_1)]) = (\mathrm{p}, [(l',0),(l_1',\mu_1')])$.

Given (Q) and (X), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,\mu_1)])}_{(\mathrm{p},[rdp1])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l_2,\mu_2)) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p},[(l,0),(l_1,\mu_1)])}_{(\mathrm{p},[rdp])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ n) \parallel C)$. The difference is in the use of Lemma 5.15 in place of Lemma 5.14 to reason about the use of DerefPtrHLI and that the pointer data structure being returned is equivalent. We use Lemma 5.71 in place of Lemma 5.70 to reason about the locations accessed within DerefPtrHLI.

Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p}, [(l, 0), (l_1, \mu_1)])}_{(\mathrm{p}, [\mathit{rdp2}])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ [\alpha,\ L,\ J,\ i-1]) \parallel C)$

This case is similar to Case $\Pi \triangleright ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ *x) \parallel C) \Downarrow^{(\mathrm{p}, [(l, 0), (l_1, \mu_1)])}_{(\mathrm{p}, [\mathit{rdp1}])} ((\mathrm{p}, \gamma,\ \sigma,\ \Delta,\ \mathrm{acc},\ (l_2, \mu_2)) \parallel C)$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow^{(1, (l^1, 0)::L^1) \parallel \dots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})}_{(\mathrm{ALL}, [\mathit{mprdp}])} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1) \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow^{(1, (l^1, 0)::L^1) \parallel \dots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0)::L^{\mathrm{q}})}_{(\mathrm{ALL}, [\mathit{mprdp}])} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1) \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$ by SMC^2 rule Multiparty Private Pointer Dereference Single Level Indirection, we have (B) $\{(x) \vdash \gamma^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$, (C) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private}\ \mathit{bty}*)\}^{\mathrm{q}}_{\mathrm{p}=1}$, (D) $\{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}},\ \mathrm{private}\ \mathit{bty}*,\ \alpha,\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ \mathit{bty}*, \mathrm{private}, \alpha))\}^{\mathrm{q}}_{\mathrm{p}=1}$, (E) $\alpha > 1$, (F) $\{\mathrm{DecodePtr}(\mathrm{private}\ \mathit{bty}*,\ \alpha,\ \omega^{\mathrm{p}}) = [\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ 1]\}^{\mathrm{q}}_{\mathrm{p}=1}$, (G) $\{\mathrm{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \mathrm{private}\ \mathit{bty}, \sigma^{\mathrm{p}}) = ([n^{\mathrm{p}}_0, ...n^{\mathrm{p}}_{\alpha-1}], 1)\}^{\mathrm{q}}_{\mathrm{p}=1}$, and (H) $\mathrm{MPC}_{dv}([[n^1_0,\ ...,\ n^1_{\alpha-1}],\ ..., [n^{\mathrm{q}}_0, ..., n^{\mathrm{q}}_{\alpha-1}]], [J^1, ..., J^{\mathrm{q}}]) = (n^1, ..., n^{\mathrm{q}})$.

Given (I) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow^{(1, (l'^1, 0)::L'^1) \parallel \dots \parallel (\mathrm{q}, (l'^{\mathrm{q}}, 0)::L'^{\mathrm{q}})}_{(\mathrm{ALL}, [d])} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n'^1) \parallel ... \parallel\ (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n'^{\mathrm{q}}))$ and (A), by Lemma 4.87 we have (J) $d = \mathit{mprdp}$.

Given (I) and (J), by SMC^2 rule Multiparty Private Pointer Dereference Single Level Indirection, we have (K) $\{(x) \vdash \gamma^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$, (L) $\{\gamma^{\mathrm{p}}(x) = (l'^{\mathrm{p}}, \mathrm{private}\ \mathit{bty}'*)\}^{\mathrm{q}}_{\mathrm{p}=1}$, (M) $\{\sigma^{\mathrm{p}}(l'^{\mathrm{p}}) = (\omega'^{\mathrm{p}},\ \mathrm{private}\ \mathit{bty}'*,\ \alpha',\ \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ \mathit{bty}'*, \mathrm{private}, \alpha'))\}^{\mathrm{q}}_{\mathrm{p}=1}$, (N) $\alpha' > 1$, (O) $\{\mathrm{DecodePtr}(\mathrm{private}\ \mathit{bty}'*,\ \alpha',\ \omega'^{\mathrm{p}}) = [\alpha', L'^{\mathrm{p}}, J'^{\mathrm{p}}, 1]\}^{\mathrm{q}}_{\mathrm{p}=1}$, (P) $\{\mathrm{Retrieve\_vals}(\alpha', L'^{\mathrm{p}}, \mathrm{private}\ \mathit{bty}', \sigma^{\mathrm{p}}) = ([n'^{\mathrm{p}}_0, ...n'^{\mathrm{p}}_{\alpha'-1}], 1)\}^{\mathrm{q}}_{\mathrm{p}=1}$, and (Q) $\mathrm{MPC}_{dv}([[n'^1_0, ..., n'^1_{\alpha'-1}], ..., [n'^{\mathrm{q}}_0, ..., n'^{\mathrm{q}}_{\alpha'-1}]], [J'^1, ..., J'^{\mathrm{q}}]) = (n'^1, ..., n'^{\mathrm{q}})$.

Given (C) and (L), by Definition 5.3 we have (R) $\{l^{\mathrm{p}} = l'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$, and (S) $\mathit{bty} = \mathit{bty}'$.

Given (D), (M), and (R), by Definition 5.4 we have (T) $\{\omega^{\mathrm{p}} = \omega'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (U) $\alpha = \alpha'$.

Given (F), (O), (S), (U), and (T), by Lemma 5.26 we have (V) $\{L^{\mathrm{p}} = L'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (W) $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (G), (P), (U), (V), and (S), by Lemma 5.39 we have (X) $\{[n^{\mathrm{p}}_0, ...n^{\mathrm{p}}_{\alpha-1}] = [n'^{\mathrm{p}}_0, ...n'^{\mathrm{p}}_{\alpha'-1}]\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (H), (Q), (X), and (W), by Axiom 5.11 we have (Y) $\{n^{\mathrm{p}} = n'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (D) and (F), by Lemma 5.62 we have accessed location (Z) $\{(\mathrm{p}, [(l^{\mathrm{p}}, 0)])\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (G), by Lemma 5.72 we have accessed locations (A1) $\{(\mathrm{p}, L^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (Z) and (A1), by Lemmas 5.44 and 5.45 we have (B1) $\{(\mathrm{p}, (l^{\mathrm{p}}, 0) :: L^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (M) and (O), by Lemma 5.62 we have accessed location (C1) $\{(\mathrm{p}, [(l'^{\mathrm{p}}, 0)])\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (P), by Lemma 5.72

we have accessed locations (D1) $\{(\mathrm{p}, L'^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (C1) and (D1), by Lemmas 5.44 and 5.45 we have (E1) $\{(\mathrm{p}, (l'^{\mathrm{p}}, 0) :: L'^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B1), (E1), (R), and (V), we have (F1) $(1, (l^1, 0) :: L^1) \parallel ... \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0) :: L^{\mathrm{q}}) = (1, (l'^1, 0) :: L'^1) \parallel ... \parallel (\mathrm{q}, (l'^{\mathrm{q}}, 0) :: L'^{\mathrm{q}})$.

Given (Y) and (F1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp1])}^{(1,(l^1,0)::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, [\alpha_\alpha, L_\alpha^1, J_\alpha^1, i-1]) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, [\alpha_\alpha, L_\alpha^{\mathrm{q}}, J_\alpha^{\mathrm{q}}, i-1]))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x)) \Downarrow_{(\mathrm{ALL}, [mprdp])}^{(1,(l^1,0)::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L^{\mathrm{q}})} ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$. We use Axiom 5.12 to reason about the behavior of MPC_{dp}.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp3])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1::(\mathrm{ALL}, [mpwdp3])}^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel ... \parallel (\mathrm{q},(l^{\mathrm{q}},0)::L_1^{\mathrm{q}}::L^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta_2^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta_2^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ by SMC^2 rule Multiparty Private Pointer Dereference Write Private Value, we have (B) $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1}^{\mathcal{L}_1} ((1, \gamma^1, \sigma_1^1, \Delta_1^1, \mathrm{acc}, n^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1^{\mathrm{q}}, \Delta_1^{\mathrm{q}}, \mathrm{acc}, n^{\mathrm{q}}))$, (D) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}}, \mathrm{private}\ bty*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (E) $\{\sigma_1^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \mathrm{private}\ bty*, \alpha, \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty*, \mathrm{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\alpha > 1$, (G) $\{\mathrm{DecodePtr}(\mathrm{private}\ bty*, \alpha, \omega^{\mathrm{p}}) = [\alpha, L^{\mathrm{p}}, J^{\mathrm{p}}, 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $\{\mathrm{DynamicUpdate}(\Delta_1^{\mathrm{p}}, \sigma_1^{\mathrm{p}}, L^{\mathrm{p}}, \mathrm{acc}, \mathrm{private}\ bty) = (\Delta_2^{\mathrm{p}}, L_1^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$,
(I) $\{\mathrm{Retrieve\_vals}(\alpha, L^{\mathrm{p}}, \mathrm{private}\ bty, \sigma_1^{\mathrm{p}}) = ([n_0^{\mathrm{p}}, ... n_{\alpha-1}^{\mathrm{p}}], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\mathrm{MPC}_{wdv}([[n_0^1, ..., n_{\alpha-1}^1], ..., [n_0^{\mathrm{q}}, ..., n_{\alpha-1}^{\mathrm{q}}]], [n^1, ..., n^{\mathrm{q}}], [J^1, ..., J^{\mathrm{q}}]) = ([n_0'^1, ..., n_{\alpha-1}'^1], ..., [n_0'^{\mathrm{q}}, ..., n_{\alpha-1}'^{\mathrm{q}}])$, and (K) $\{\mathrm{UpdateDerefVals}(\alpha, L^{\mathrm{p}}, [n_0'^{\mathrm{p}}, ..., n_{\alpha-1}'^{\mathrm{p}}], \mathrm{private}\ bty, \sigma_1^{\mathrm{p}}) = \sigma_2^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (L) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow_{\mathcal{D}_1'::(\mathrm{ALL}, [d])}^{\mathcal{L}_1'::(1,(l'^1,0)::L_1'^1::L'^1) \parallel ... \parallel (\mathrm{q},(l'^{\mathrm{q}},0)::L_1'^{\mathrm{q}}::L'^{\mathrm{q}})} ((1, \gamma^1, \sigma_2'^1, \Delta_2'^1, \mathrm{acc}, \mathrm{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2'^{\mathrm{q}}, \Delta_2'^{\mathrm{q}}, \mathrm{acc}, \mathrm{skip}))$ and (A), by Lemma 4.87 we have (M) $d = mpwdp3$.

Given (L) and (M), by SMC^2 rule Multiparty Private Pointer Dereference Write Private Value, we have (N) $\{(e) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (O) $((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, e) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, e)) \Downarrow_{\mathcal{D}_1'}^{\mathcal{L}_1'} ((1, \gamma^1, \sigma_1'^1, \Delta_1'^1, \mathrm{acc}, n'^1) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_1'^{\mathrm{q}}, \Delta_1'^{\mathrm{q}}, \mathrm{acc}, n'^{\mathrm{q}}))$, (P) $\{\gamma^{\mathrm{p}}(x) = (l'^{\mathrm{p}}, \mathrm{private}\ bty'*)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Q) $\{\sigma_1'^{\mathrm{p}}(l'^{\mathrm{p}}) = (\omega'^{\mathrm{p}}, \mathrm{private}\ bty'*, \alpha', \mathrm{PermL\_Ptr}(\mathrm{Freeable}, \mathrm{private}\ bty'*, \mathrm{private}, \alpha'))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (R) $\alpha' > 1$, (S) $\{\mathrm{DecodePtr}(\mathrm{private}\ bty'*, \alpha', \omega'^{\mathrm{p}}) = [\alpha', L'^{\mathrm{p}}, J'^{\mathrm{p}}, 1]\}_{\mathrm{p}=1}^{\mathrm{q}}$, (T) $\{\mathrm{DynamicUpdate}(\Delta_1'^{\mathrm{p}}, \sigma_1'^{\mathrm{p}}, L'^{\mathrm{p}}, \mathrm{acc}, \mathrm{private}\ bty') = (\Delta_2'^{\mathrm{p}}, L_1'^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$,
(U) $\{\mathrm{Retrieve\_vals}(\alpha', L'^{\mathrm{p}}, \mathrm{private}\ bty', \sigma_1'^{\mathrm{p}}) = ([n_0''^{\mathrm{p}}, ... n_{\alpha'-1}''^{\mathrm{p}}], 1)\}_{\mathrm{p}=1}^{\mathrm{q}}$, (V) $\mathrm{MPC}_{wdv}([[n_0''^1, ..., n_{\alpha'-1}''^1], ..., [n_0''^{\mathrm{q}},$

$\ldots, n''^{\mathrm{q}}_{\alpha'-1}]], [n'^1, \ldots, n'^{\mathrm{q}}], [J'^1, \ldots, J'^{\mathrm{q}}]) = ([n'''^1_0, \ldots, n'''^1_{\alpha'-1}], \ldots, [n'''^{\mathrm{q}}_0, \ldots, n'''^{\mathrm{q}}_{\alpha'-1}])$, and
(W) $\{\mathrm{UpdateDerefVals}(\alpha', L'^{\mathrm{p}}, [n'''^{\mathrm{p}}_0, \ldots, n'''^{\mathrm{p}}_{\alpha'-1}], \text{private } bty', \sigma'^{\mathrm{p}}_1) = \sigma'^{\mathrm{p}}_2\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (C) and (O), by the inductive hypothesis we have (X) $\{\sigma^{\mathrm{p}}_1 = \sigma'^{\mathrm{p}}_1\}^{\mathrm{q}}_{\mathrm{p}=1}$, (Y) $\{\Delta^{\mathrm{p}}_1 = \Delta'^{\mathrm{p}}_1\}^{\mathrm{q}}_{\mathrm{p}=1}$, (Z) $\{n^{\mathrm{p}} = n'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$, (A1) $\mathcal{D}_1 = \mathcal{D}'_1$, and (B1) $\mathcal{L}_1 = \mathcal{L}'_1$.

Given (D) and (P), by Definition 5.3 we have (C1) $\{l^{\mathrm{p}} = l'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (D1) $bty = bty'$.

Given (E), (Q), (X), and (C1), by Definition 5.4 we have (E1) $\{\omega^{\mathrm{p}} = \omega'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (F1) $\alpha = \alpha'$.

Given (G), (S), (W), and (X), by Lemma 5.26 we have (G1) $\{L^{\mathrm{p}} = L'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (H1) $\{J^{\mathrm{p}} = J'^{\mathrm{p}}\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (F), (N), (Q), (P), (Y), and (W), by Lemma 5.25 we have (I1) $\{\Delta^{\mathrm{p}}_2 = \Delta'^{\mathrm{p}}_2\}^{\mathrm{q}}_{\mathrm{p}=1}$ and (J1) $\{L^{\mathrm{p}}_1 = L'^{\mathrm{p}}_1\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (I), (U), (F1), (G1), (D1), and (X), by Lemma 5.39 we have (K1) $\{[n^{\mathrm{p}}_0, \ldots n^{\mathrm{p}}_{\alpha-1}] = [n''^{\mathrm{p}}_0, \ldots n''^{\mathrm{p}}_{\alpha'-1}]\}^{\mathrm{q}}_{\mathrm{p}=1}$

Given (J), (V), (K1), (Z), and (H1), by Axiom 5.14 we have (L1) $\{[n'^{\mathrm{p}}_0, \ldots, n'^{\mathrm{p}}_{\alpha-1}] = [n'''^{\mathrm{p}}_0, \ldots, n'''^{\mathrm{p}}_{\alpha'-1}]\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (K), (W), (F1), (G1), (L1), (D1), and (X), by Lemma 5.43 we have (M1) $\{\sigma^{\mathrm{p}}_2 = \sigma'^{\mathrm{p}}_2\}^{\mathrm{q}}_{\mathrm{p}=1}$.

Given (A1) and (ALL, $[mpwdp3]$), by Lemma 5.38 we have (N1) $\mathcal{D}_1 :: (\mathrm{ALL}, [mpwdp3]) = \mathcal{D}'_1 :: (\mathrm{ALL}, [mpwdp3])$.

Given (E) and (G), by Lemma 5.62 we have accessed location (O1) $\{(\mathrm{p}, [(l^{\mathrm{p}}, 0)])\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (N), by Lemma 5.61 we have accessed locations (P1) $\{(\mathrm{p}, L^{\mathrm{p}}_1)\}^{\mathrm{q}}_{\mathrm{p}=1}$ Given (I) and (K), by Lemma 5.72 and 5.76 we have accessed locations (Q1) $\{(\mathrm{p}, L^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$ Given (C), (O1), (P1), and (Q1), by Lemmas 5.44 and 5.45 we have (R1) $\mathcal{L}_1 :: (1, (l^1, 0) :: L^1_1 :: L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0) :: L^{\mathrm{q}}_1 :: L^{\mathrm{q}})$.

Given (Q) and (S), by Lemma 5.62 we have accessed location (S1) $\{(\mathrm{p}, [(l'^{\mathrm{p}}, 0)])\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (T), by Lemma 5.61 we have accessed locations (T1) $\{(\mathrm{p}, L'^{\mathrm{p}}_1)\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (U) and (W), by Lemma 5.72 and 5.76 we have accessed locations (U1) $\{(\mathrm{p}, L'^{\mathrm{p}})\}^{\mathrm{q}}_{\mathrm{p}=1}$. Given (O), (S1), (T1), and (U1), by Lemmas 5.44 and 5.45 we have (V1) $\mathcal{L}'_1 :: (1, (l'^1, 0) :: L'^1_1 :: L'^1) \parallel \ldots \parallel (\mathrm{q}, (l'^{\mathrm{q}}, 0) :: L'^{\mathrm{q}}_1 :: L'^{\mathrm{q}})$.

Given (R1), (V1), (B1), (C1), (J1), and (G1), by Lemma 5.47 we have (W1) $\mathcal{L}_1 :: (1, (l^1, 0) :: L^1_1 :: L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0) :: L^{\mathrm{q}}_1 :: L^{\mathrm{q}}) = \mathcal{L}'_1 :: (1, (l'^1, 0) :: L'^1_1 :: L'^1) \parallel \ldots \parallel (\mathrm{q}, (l'^{\mathrm{q}}, 0) :: L'^{\mathrm{q}}_1 :: L'^{\mathrm{q}})$

Given (M1), (I1), (W1), and (N1), by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e)) \Downarrow^{\mathcal{L}_1 :: (1, (l^1, 0) :: L^1_1 :: L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0) :: L^{\mathrm{q}}_1 :: L^{\mathrm{q}})}_{\mathcal{D}_1 :: (\mathrm{ALL}, [mpwdp])}$ $((1, \gamma^1, \sigma^1_2, \Delta^1_2, \mathrm{acc}, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_2, \Delta^{\mathrm{q}}_2, \mathrm{acc}, \mathrm{skip}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \mathrm{acc}, *x = e) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \mathrm{acc}, *x = e))$ $\Downarrow^{\mathcal{L}_1 :: (1, (l^1, 0) :: L^1_1 :: L^1) \parallel \ldots \parallel (\mathrm{q}, (l^{\mathrm{q}}, 0) :: L^{\mathrm{q}}_1 :: L^{\mathrm{q}})}_{\mathcal{D}_1 :: (\mathrm{ALL}, [mpwdp3])}$ $((1, \gamma^1, \sigma^1_2, \Delta^1_2, \mathrm{acc}, \mathrm{skip}) \parallel \ldots \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}_2, \Delta^{\mathrm{q}}_2, \mathrm{acc}, \mathrm{skip}))$. We use Axiom 5.1 to reason about the use of encrypt.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e)) \Downarrow^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}_{\mathcal{D}_1::(\text{ALL},[mpwdp2])}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e))$ $\Downarrow^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}_{\mathcal{D}_1::(\text{ALL},[mpwdp3])}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$. We use Axiom 5.15 to reason about the use of MPC_{wdp}.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e)) \Downarrow^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}_{\mathcal{D}_1::(\text{ALL},[mpwdp1])}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$

This case is similar to Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, *x = e) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, *x = e))$ $\Downarrow^{\mathcal{L}_1::(1,(l^1,0)::L_1^1::L^1) \parallel \ldots \parallel (\text{q},(l^\text{q},0)::L_1^\text{q}::L^\text{q})}_{\mathcal{D}_1::(\text{ALL},[mpwdp3])}$ $((1, \gamma^1, \sigma_2^1, \Delta_2^1, \text{acc}, \text{skip}) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_2^\text{q}, \Delta_2^\text{q}, \text{acc}, \text{skip}))$. We use Axiom 5.15 to reason about the use of MPC_{wdp}.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, {++}\ x) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, {++}\ x)) \Downarrow^{(1,[(l^1,0)]) \parallel \ldots \parallel (\text{q},[(l^\text{q},0)])}_{(\text{ALL},[mppin])} ((1, \gamma^1, \sigma_1^1, \Delta^1,$ $\text{acc}, n_2^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_1^\text{q}, \Delta^\text{q}, \text{acc}, n_2^\text{q}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, {++}\ x) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, {++}\ x)) \Downarrow^{(1,[(l^1,0)]) \parallel \ldots \parallel (\text{q},[(l^\text{q},0)])}_{(\text{ALL},[mppin])} ((1, \gamma^1, \sigma_1^1,$ $\Delta^1, \text{acc}, n_2^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_1^\text{q}, \Delta^\text{q}, \text{acc}, n_2^\text{q}))$ by SMC^2 rule Multiparty Pre-Increment Private Float Variable, we have (B) $\{\gamma^\text{p}(x) = (l^\text{p}, \text{private float})\}_{\text{p}=1}^\text{q}$, (C) $\{\sigma^\text{p}(l^\text{p}) = (\omega^\text{p}, \text{private float}, 1, \text{PermL}(\text{Freeable}, \text{private float}, \text{private}, 1))\}_{\text{p}=1}^\text{q}$, (D) $\{(x) \vdash \gamma^\text{p}\}_{\text{p}=1}^\text{q}$, (E) $\{\text{DecodeVal}(\text{private float}, \omega^\text{p}) = n_1^\text{p}\}_{\text{p}=1}^\text{q}$, (F) $\text{MPC}_u({++}, n_1^1, ..., n_1^\text{q}) = (n_2^1, ..., n_2^\text{q})$, and (G) $\{\text{UpdateVal}(\sigma^\text{p}, l^\text{p}, n_2^\text{p}, \text{private float}) = \sigma_1^\text{p}\}_{\text{p}=1}^\text{q}$.

Given (H) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, {++}\ x) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma^\text{q}, \Delta^\text{q}, \text{acc}, {++}\ x)) \Downarrow^{(1,[(l'^1,0)]) \parallel \ldots \parallel (\text{q},[(l'^\text{q},0)])}_{(\text{ALL},[d])} ((1, \gamma^1, \sigma_1'^1,$ $\Delta^1, \text{acc}, n_2'^1) \parallel ... \parallel (\text{q}, \gamma^\text{q}, \sigma_1'^\text{q}, \Delta^\text{q}, \text{acc}, n_2'^\text{q}))$ and (A), by Lemma 4.87 we have (I) $d = mppin$.

Given (H) and (I), by SMC^2 rule Multiparty Pre-Increment Private Float Variable, we have (J) $\{\gamma^\text{p}(x) = (l'^\text{p}, \text{private float})\}_{\text{p}=1}^\text{q}$, (K) $\{\sigma^\text{p}(l'^\text{p}) = (\omega'^\text{p}, \text{private float}, 1, \text{PermL}(\text{Freeable}, \text{private float}, \text{private}, 1))\}_{\text{p}=1}^\text{q}$, (L)

$\{(x) \vdash \gamma^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (M) $\{\text{DecodeVal}(\text{private float}, \omega'^{\mathrm{p}}) = n_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, (N) $\text{MPC}_u(++, n_1'^1, ..., n_1'^{\mathrm{q}}) = (n_2'^1, ..., n_2'^{\mathrm{q}})$, and (O) $\{\text{UpdateVal}(\sigma^{\mathrm{p}}, l'^{\mathrm{p}}, n_2'^{\mathrm{p}}, \text{private float}) = \sigma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (B) and (J), by Definition 5.3 we have (P) $\{l^{\mathrm{p}} = l'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (C), (K), and (P), by Definition 5.4 we have (Q) $\{\omega^{\mathrm{p}} = \omega'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (E), (M), and (Q), by Lemma 5.29 we have (R) $\{n_1^{\mathrm{p}} = n_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (F), (N), and (R), by Axiom 5.9 we have (S) $\{n_2^{\mathrm{p}} = n_2'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (G), (O), (P), and (S), by Lemma 5.34 we have (T) $\{\sigma_1^{\mathrm{p}} = \sigma_1'^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (A), (H), and (I), we have (U) $(\text{ALL}, [mppin]) = (\text{ALL}, [mppin])$.

Given (C), (E), and (G), by Lemma 5.64 and Lemma 5.66 we have accessed location (V) $\{(\mathrm{p}, [(l, 0)])\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (V), by Lemmas 5.44 and 5.46 we have (W) $(1, [(l^1, 0)]) \parallel ... \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)])$.

Given (K), (M), and (O), by Lemma 5.64 and Lemma 5.66 we have accessed location (X) $\{(\mathrm{p}, [(l', 0)])\}_{\mathrm{p}=1}^{\mathrm{q}}$. Given (X), by Lemmas 5.44 and 5.46 we have (Y) $(1, [(l'^1, 0)]) \parallel ... \parallel (\mathrm{q}, [(l'^{\mathrm{q}}, 0)])$.

Given (W), (Y), and (P), we have (Z) $(1, [(l^1, 0)]) \parallel ... \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)]) = (1, [(l'^1, 0)]) \parallel ... \parallel (\mathrm{q}, [(l'^{\mathrm{q}}, 0)])$.

Given (T), (S), (U), and (Z) by Definition 5.2 we have $\Pi \simeq_L \Sigma$.

Case $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{pfree}(x)))$ $\Downarrow_{(\text{ALL}, [mpfre])}^{(1, [(l^1, 0)]::L^1::L_1^1) \parallel ... \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)]::L^{\mathrm{q}}::L_1^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta^1, \text{acc}, \text{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{skip}))$

Given (A) $\Pi \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{pfree}(x)))$ $\Downarrow_{(\text{ALL}, [mpfre])}^{(1, [(l^1, 0)]::L^1::L_1^1) \parallel ... \parallel (\mathrm{q}, [(l^{\mathrm{q}}, 0)]::L^{\mathrm{q}}::L_1^{\mathrm{q}})} ((1, \gamma^1, \sigma_2^1, \Delta^1, \text{acc}, \text{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{skip}))$ by SMC^2 rule Private Free Multiple Locations, we have (B) $\{\gamma^{\mathrm{p}}(x) = (l^{\mathrm{p}},\ \text{private } bty{*})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (C) $\text{acc} = 0$, (D) $(bty = \text{int}) \vee (bty = \text{float})$, (E) $\{\sigma^{\mathrm{p}}(l^{\mathrm{p}}) = (\omega^{\mathrm{p}}, \text{private } bty{*}, \alpha, \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty{*}, \text{private}, \alpha))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (F) $\{\alpha > 1\}_{\mathrm{p}=1}^{\mathrm{q}}$, (G) $\{[\alpha,\ L^{\mathrm{p}},\ J^{\mathrm{p}},\ i] = \text{DecodePtr}(\text{private } bty{*},\ \alpha,\ \omega^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (H) $\text{if}(i > 1)\{ty = \text{private } bty{*}\} \text{ else } \{ty = \text{private } bty\}$, (I) $\{\text{CheckFreeable}(\gamma^{\mathrm{p}}, L^{\mathrm{p}}, J^{\mathrm{p}}, \sigma^{\mathrm{p}}) = 1\}_{\mathrm{p}=1}^{\mathrm{q}}$, (J) $\{\forall (l_m^{\mathrm{p}}, 0) \in L^{\mathrm{p}}.\ \sigma^{\mathrm{p}}(l_m^{\mathrm{p}}) = (\omega_m^{\mathrm{p}}, ty, \alpha_m, \text{PermL}(\text{Freeable}, ty, \text{private}, \alpha_m))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (K) $\text{MPC}_{free}([[\omega_0^1, ..., \omega_{\alpha-1}^1], ..., [\omega_0^{\mathrm{q}}, ..., \omega_{\alpha-1}^{\mathrm{q}}]], [J^1, ... J^{\mathrm{q}}]) = ([[\omega_0'^1, ..., \omega_{\alpha-1}'^1], ..., [\omega_0'^{\mathrm{q}}, ..., \omega_{\alpha-1}'^{\mathrm{q}}]], [J'^1, ..., J'^{\mathrm{q}}])$, (L) $\{\text{UpdateBytesFree}(\sigma^{\mathrm{p}}, L^{\mathrm{p}}, [\omega_0'^{\mathrm{p}}, ..., \omega_{\alpha-1}'^{\mathrm{p}}]) = \sigma_1^{\mathrm{p}}\}_{\mathrm{p}=1}^{\mathrm{q}}$, and (M) $\{(\sigma_2^{\mathrm{p}}, L_1^{\mathrm{p}}) = \text{UpdatePointerLocations}(\sigma_1^{\mathrm{p}}, L^{\mathrm{p}}[1:\alpha-1], J^{\mathrm{p}}[1:\alpha-1], L^{\mathrm{p}}[0], J^{\mathrm{p}}[0])\}_{\mathrm{p}=1}^{\mathrm{q}}$.

Given (N) $\Sigma \triangleright ((1, \gamma^1, \sigma^1, \Delta^1, \text{acc}, \text{pfree}(x)) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{pfree}(x)))$ $\Downarrow_{(\text{ALL}, [d])}^{(1, [(l'^1, 0)]::L'^1::L_1'^1) \parallel ... \parallel (\mathrm{q}, [(l'^{\mathrm{q}}, 0)]::L'^{\mathrm{q}}::L_1'^{\mathrm{q}})} ((1, \gamma^1, \sigma_2'^1, \Delta^1, \text{acc}, \text{skip}) \parallel ... \parallel (\mathrm{q}, \gamma^{\mathrm{q}}, \sigma_2'^{\mathrm{q}}, \Delta^{\mathrm{q}}, \text{acc}, \text{skip}))$ and (A), by Lemma 4.87 we have (O) $d = mpfre$.

Given (N) and (O), by SMC^2 rule Private Free Multiple Locations, we have (P) $\{\gamma^{\mathrm{p}}(x) = (l'^{\mathrm{p}},\ \text{private } bty'{*})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (Q) $\text{acc} = 0$, (R) $(bty' = \text{int}) \vee (bty' = \text{float})$, (S) $\{\sigma^{\mathrm{p}}(l'^{\mathrm{p}}) = (\omega'^{\mathrm{p}}, \text{private } bty'{*}, \alpha', \text{PermL\_Ptr}(\text{Freeable}, \text{private } bty'{*}, \text{private}, \alpha'))\}_{\mathrm{p}=1}^{\mathrm{q}}$, (T) $\{\alpha' > 1\}_{\mathrm{p}=1}^{\mathrm{q}}$, (U) $\{[\alpha',\ L'^{\mathrm{p}},\ J''^{\mathrm{p}},\ i'] = \text{DecodePtr}(\text{private } bty'{*},\ \alpha',\ \omega'^{\mathrm{p}})\}_{\mathrm{p}=1}^{\mathrm{q}}$, (V) $\text{if}(i' > 1)\{ty' = \text{private } bty'{*}\} \text{ else } \{ty' = \text{private } bty'\}$, (W) $\{\text{CheckFreeable}(\gamma^{\mathrm{p}}, L'^{\mathrm{p}}, J''^{\mathrm{p}}, \sigma^{\mathrm{p}}) = 1\}_{\mathrm{p}=1}^{\mathrm{q}}$,

(X) $\{\forall(l'^{p}_{m'}, 0) \in L'^{p}.\ \sigma^{p}(l'^{p}_{m'}) = (\omega''^{p}_{m'}, ty', \alpha'_{m'}, \text{PermL}(\text{Freeable}, ty', \text{private}, \alpha'_{m'}))\}^{q}_{p=1}$, (Y) $\text{MPC}_{free}([[\omega''^{1}_{0}, ..., \omega''^{1}_{\alpha'-1}], ..., [\omega''^{q}_{0}, ..., \omega''^{q}_{\alpha'-1}]], [J''^{1}, ...J''^{q}]) = ([[\omega'''^{1}_{0}, ..., \omega'''^{1}_{\alpha'-1}], ..., [\omega'''^{q}_{0}, ..., \omega'''^{q}_{\alpha'-1}]], [J'''^{1}, ..., J'''^{q}])$, (Z) $\{\text{UpdateBytesFree}(\sigma^{p}, L'^{p}, [\omega'''^{p}_{0}, ..., \omega'''^{p}_{\alpha'-1}]) = \sigma'^{p}_{1}\}^{q}_{p=1}$, and (A1) $\{(\sigma'^{p}_{2}, L'^{p}_{1}) = \text{UpdatePointerLocations}(\sigma'^{p}_{1}, L'^{p}[1 : \alpha' - 1], J''^{p}[1 : \alpha' - 1], L'^{p}[0], J''^{p}[0])\}^{q}_{p=1}$.

Given (B) and (P), by Definition 5.3 we have (B1) $\{l^{p} = l'^{p}\}^{q}_{p=1}$ and (C1) $bty = bty'$.

Given (E), (S), and (B1), by Definition 5.4 we have (D1) $\{\omega^{p} = \omega'^{p}\}^{q}_{p=1}$ and (E1) $\alpha = \alpha'$.

Given (G), (U), (C1), (E1), and (D1), by Lemma 5.26 we have (F1) $\{L^{p} = L'^{p}\}^{q}_{p=1}$, (G1) $\{J^{p} = J''^{p}\}^{q}_{p=1}$, and (H1) $i = i'$.

Given (H), (V), (H1), and (C1), we have (I1) $ty = ty'$.

Given (I), (W), (F1), and (G1), by Lemma 5.40 we have (J1) $1 = 1$.

Given (J), (X), and (F1), we have (K1) $\{l^{p}_{m} = l'^{p}_{m'}\}^{q}_{p=1}$ such that (L1) $m = m'$. Given (J), (X), (F1), (K1), (E1), and (L1), by Definition 5.4 we have (M1) $\{\forall m = m' \in \{0...\alpha - 1\},\ \omega^{p}_{m} = \omega''^{p}_{m'}\}^{q}_{p=1}$ and (N1) $\forall m = m' \in \{0...\alpha - 1\},\ \alpha_{m} = \alpha'_{m'}$.

Given (K), (Y), (E1), (M1), and (G1), by Axiom 4.14 we have (O1) $\{\forall m = m' \in \{0...\alpha - 1\},\ \omega'^{p}_{m} = \omega'''^{p}_{m'}\}^{q}_{p=1}$ and (P1) $\{J'^{p} = J'''^{p}\}^{q}_{p=1}$.

Given (L), (Z), (F1), (O1), and (E1), by Lemma 5.41 we have (Q1) $\{\sigma^{p}_{1} = \sigma'^{p}_{1}\}^{q}_{p=1}$.

Given (M), (A1), (Q1), (F1), (E1), and (G1), by Lemma 5.42 we have (R1) $\{\sigma^{p}_{2} = \sigma'^{p}_{2}\}^{q}_{p=1}$ and (S1) $\{L^{p}_{1} = L'^{p}_{1}\}^{q}_{p=1}$.

Given (A), (N), and (O), we have (T1) $(\text{ALL}, [mpfre]) = (\text{ALL}, [mpfre])$.

Given (E) and (G), by Lemma 5.62 we have accessed location (U1) $\{(\text{p}, [(l^{p}, 0)])\}^{q}_{p=1}$. Given (I), (J), (L), by Lemma 5.73 and Lemma 5.74 we have accessed locations (V1) $\{(\text{p}, L^{p})\}^{q}_{p=1}$. Given (M), by Lemma 5.75 we have accessed locations (W1) $\{(\text{p}, L^{p}_{1})\}^{q}_{p=1}$. Given (U1), (V1), and (W1), by Lemmas 5.44 and 5.46 we have (X1) $(1, [(l^{1}, 0)] :: L^{1} :: L^{1}_{1})\ \|\ ...\ \|\ (\text{q}, [(l^{q}, 0)] :: L^{q} :: L^{q}_{1})$.

Given (S) and (U), by Lemma 5.62 we have accessed location (Y1) $\{(\text{p}, [(l'^{p}, 0)])\}^{q}_{p=1}$. Given (W), (X), (Z), by Lemma 5.73 and Lemma 5.74 we have accessed locations (Z1) $\{(\text{p}, L'^{p})\}^{q}_{p=1}$. Given (A1), by Lemma 5.75 we have accessed locations (A2) $\{(\text{p}, L'^{p}_{1})\}^{q}_{p=1}$. Given (Y1), (Z1), and (A2), by Lemmas 5.44 and 5.46 we have (B2) $(1, [(l'^{1}, 0)] :: L'^{1} :: L'^{1}_{1})\ \|\ ...\ \|\ (\text{q}, [(l'^{q}, 0)] :: L'^{q} :: L'^{q}_{1})$.

Given (X1), (B2), (B1), (F1), and (S1), we have (C2) $(1, [(l^{1}, 0)] :: L^{1} :: L^{1}_{1})\ \|\ ...\ \|\ (\text{q}, [(l^{q}, 0)] :: L^{q} :: L^{q}_{1}) = (1, [(l'^{1}, 0)] :: L'^{1} :: L'^{1}_{1})\ \|\ ...\ \|\ (\text{q}, [(l'^{q}, 0)] :: L'^{q} :: L'^{q}_{1})$.

Given (R1), (T1), and (C2) by Definition 5.2 we have $\Pi \simeq_{L} \Sigma$.

□

6 EXAMPLE PROGRAMS

In this section, we show the PICCO programs that can be used to run each of the examples given in the main paper. The program with a simple example of a private-conditioned branch (Figure 6(a) and 9(a) in the main paper) is shown in Figure 45. The program for challenges of pointer manipulations inside private-conditioned branched (Figure 7(a) and 9(c) in the main paper) is shown in Figure 46. The program used to illustrate why single-statement resolution is more costly when modifying variables multiple times in both branches (Figure 8(a) in the main paper) is shown in Figure 47. More lengthy versions of this program designed to stress the differences caused by this are shown in the benchmarks section in Figure 55, 56, 57, and 58. The program giving a simple example of pointer use within a private-conditioned branch (Figure 9(b) in the main paper) is shown in Figure 48.

```
public int main() {
  private int a=3, b=7, c=0;
  if (a < b) { c = a; }
  else { c = b; }
  return 0;
}
```

Fig. 45. Simple example of a private-conditioned branch.

```
public int main() {
  private int a=3, b=7, c=5, *p=&a;
  if (a < b) { *p = c; }
  else { p = &b; }
  return 0;
}
```

Fig. 46. Challenges of pointer manipulations within private-conditioned branched.

```
public int main() {
  private int c, a=1, b=2;
  if(a < b) {
    c = a;
    a = a + b;
    c = c * b;
    a = c + a;
  }
  else {
    c = b;
    a = a - b;
    c = c * a;
    a = c - a;
  }
}
```

Fig. 47. Illustrating why single-statement resolution is more costly when modifying variables multiple times in both branches.

```
public int main() {
  private int a=3, b=7, *p;
  if (a < b) { p = &a; }
  else { p = &b; }
  return 0;
}
```

Fig. 48. Simple example of pointer use within a private-conditioned branch.

```
public int main() {
  public int i=1,j=2;
  private int a[j],b=7,c=3,d=4;
  a[0]=0; a[1]=0;
  if (c<d) { a[i]=c; }
  else { a[j]=d; }
  return 0;
}
```

Fig. 49. Challenges of writing at a public index in a private array within a private-conditioned branch.

The program showing the challenges of writing to a private array at a public index within a private-conditioned branch (Figure 9(d) in the main paper) is shown in Figure 49. This highlights how simple variable tracking cannot be trusted to be correct for arrays when a public index is used due to the potential for having an out-of-bounds access. When this program is run, it is not guaranteed to run without error, as we do not guarantee that out-of-bounds accesses will be well-aligned or correct within the implementation. The program for pay-gap (shown in Figure 1 in the main paper) is showing in Figure 51, as it is also used as a benchmarking program. The program for the modified version of pay-gap (Figure 11 in the main paper) is shown in Figure 50.

```
public int main() {
   public int numParticipants = 100, i, j, maxInputSize = 100;
   public int inputSize[numParticipants], inputNum;
   private int salary[numParticipants][maxInputSize];
   private int<1> gender[numParticipants][maxInputSize];
   private int avgMaleSalary = 0, avgFemaleSalary = 0;
   private int maleCount = 0, femaleCount = 0;
   public int historicFemaleSalaryAvg, historicMaleSalaryAvg;
   public int avgFemaleSalPub, femaleCountPub;
   public int avgMaleSalPub, maleCountPub;

   smcinput(inputSize, 1, numParticipants);
   smcinput(gender, 1, numParticipants, maxInputSize);
   smcinput(salary, 1, numParticipants, maxInputSize);
   smcinput(historicFemaleSalaryAvg, 1);
   smcinput(historicMaleSalaryAvg, 1);

   for (i = 0; i < numParticipants; i++){
      for (j = 0; j < inputSize[i]; j++){
         if (gender[i][j] == 0) {
            avgFemaleSalary += salary[i][j];
            femaleCount++;
         }
         else {
            avgMaleSalary += salary[i][j];
            maleCount++;
         }
      }
   }

   avgFemaleSalPub=smcopen(avgFemaleSalary);
   femaleCountPub=smcopen(femaleCount);
   avgMaleSalPub=smcopen(avgMaleSalary);
   maleCountPub=smcopen(maleCount);

   avgFemaleSalPub=(avgFemaleSalPub/femaleCountPub)/2+historicFemaleSalaryAvg/2;
   avgMaleSalPub=(avgMaleSalPub/maleCountPub)/2+historicMaleSalaryAvg/2;

   smcoutput(avgFemaleSalPub, 1);
   smcoutput(avgMaleSalPub, 1);
   return 1;
}
```

Fig. 50. Example program: extended version of pay-gap

7 BENCHMARKS

Benchmarking program pay-gap is shown in Figure 51. Here, we use the PICCO syntax for executing `smcinput` and `smcoutput` this program, as opposed to the simplified syntax used in the Figure 1 in the main paper. For simplicity, we aggregated the input data into a single file, rather than reading from 100 different files.

```
public int main() {
   public int numParticipants = 100, i, j, maxInputSize = 100;
   public int inputSize[numParticipants], inputNum;
   private int salary[numParticipants][maxInputSize];
   private int<1> gender[numParticipants][maxInputSize];
   private int avgMaleSalary = 0, avgFemaleSalary = 0;
   private int maleCount = 0, femaleCount = 0;
   public int historicFemaleSalaryAvg, historicMaleSalaryAvg;

   smcinput(inputSize, 1, numParticipants);
   smcinput(gender, 1, numParticipants, maxInputSize);
   smcinput(salary, 1, numParticipants, maxInputSize);
   smcinput(historicFemaleSalaryAvg, 1);
   smcinput(historicMaleSalaryAvg, 1);

   for (i = 0; i < numParticipants; i++){
      for (j = 0; j < inputSize[i]; j++){
         if (gender[i][j] == 0) {
            avgFemaleSalary += salary[i][j];
            femaleCount++;
         }
         else {
            avgMaleSalary += salary[i][j];
            maleCount++;
         }
      }
   }

   avgFemaleSalary=(avgFemaleSalary/femaleCount)/2 + historicFemaleSalaryAvg/2;
   avgMaleSalary=(avgMaleSalary/maleCount)/2 + historicFemaleSalaryAvg/2;

   smcoutput(avgFemaleSalary, 1);
   smcoutput(avgMaleSalary, 1);
   return 1;
}
```

Fig. 51. Benchmarking program: pay-gap.c

Benchmarking program LR-parser is split into two parts due to the length of the program, and shown in Figures 52 and 53. When reading the program, be aware that several lines contain multiple statements to be able to show this program within two figures, and the program contains comments (enclosed in `/* ... */`) to help understand the program.

```
public int K = 100; /* max number of variables in the expression */
  /* this defines integer representation of symbols: */
  /* + = K; * = K+1; ( = K+2; ) = K+3; EOF = K+4  */
public int M = 10; /* the number of variables in the expression */
public int S = 29; /* the length of the expression */
struct token{ private int val; public int type; struct token* next; };
  /* type == 0 --- id; type == 1 --- F; type == 2 --- T; type == 3 --- S  */
  /* type == 4 --- +; type == 5 --- *; type == 6 --- (; type == 7 --- ) */
struct token *pop(struct token** header) {
  struct token* t = *header; struct token* tmp = *header;
  *header = tmp->next; return t; }
public void push(struct token** header, struct token* t) {
  t->next = *header;  *header = t;  }
public void id_routine(struct token** header, int val) {
  struct token* t; t = pmalloc(1, struct token);
  t->type = 0; t->val = val; push(header, t); }
public void check_for_removable_lbra(struct token** header) {
  struct token* t; t = pop(header);
  if(t->type != 6) push(header, t); }
public void prod_sub_routine(struct token** header,struct token* x1, public int
 flag){
  struct token* x3;  x3 = pop(header);
  x1->type = 2; /* T */ x1->val =  x3->val * x1->val;
  if(*header != 0) {
    struct token* x4; x4 = pop(header);
    if(x4->type == 4) /* + */ {
      struct token* x5; x5 = pop(header);
      x1->val = x1->val + x5->val; x1->type = 3; }
    else push(header, x4); }
  if(flag == 1) check_for_removable_lbra(header);
  push(header, x1); }
public void plus_sub_routine(struct token** header,struct token* x1, public int
 flag){
  struct token* x3; x3 = pop(header);
  x1->type = 3; x1->val = x1->val + x3->val;
  if(flag == 1) check_for_removable_lbra(header);
  push(header, x1); }
public void plus_routine(struct token** header) {
  struct token* plus; plus = pmalloc(1, struct token); plus->type = 4;
  struct token* x1; x1 = pop(header);
  if(*header != 0) {
    if(x1->type == 0) /* id */ x1->type == 1; /* F */
    struct token* x2; x2 = pop(header);
    if(x2->type == 5) /* * */ prod_sub_routine(header, x1, 0);
    else if(x2->type == 4) /* +  */ plus_sub_routine(header, x1, 0);
    else if(x2->type == 6) {  /* (  */
      x1->type = 3; /* S */ push(header, x2); push(header, x1);  } }
  else { x1->type = 3; push(header, x1); }
  push(header, plus); }
```

Fig. 52. Benchmarking program: LR-parser.c (Part 1/2)

```
public void prod_routine(struct token** header) {
  struct token* prod; prod = pmalloc(1, struct token);
  prod->type = 5; struct token* x1; x1 = pop(header);
  if(*header != 0){
    if(x1->type == 0) /* id */ x1->type == 1; /* F */
    struct token* x2; x2 = pop(header);
    if(x2->type == 5){
      struct token* x3; x3 = pop(header);
      x1->type = 2; x1->val = x1->val * x3->val; push(header, x1); } /* * */
    else if(x2->type == 4 || x2->type == 6) {  /* + or ( */
      x1->type = 2; /* T  */ push(header, x2); push(header, x1); } }
  else { x1->type = 2; push(header, x1); }
  push(header, prod); }
public void lbra_routine(struct token** header) {
  struct token* t;  t = pmalloc(1, struct token);
  t->type = 6;  push(header, t); }
public void rbra_routine(struct token** header){
  struct token* x1; x1 = pop(header);
  if(*header != 0) {
  if(x1->type == 0) /* id  */ x1->type == 1; /* F */
  struct token* x2; x2 = pop(header);
    if(x2->type == 5) /* * */ prod_sub_routine(header, x1, 1);
    else if(x2->type == 4) /* + */ plus_sub_routine(header, x1, 1);
    else if(x2->type == 6) { x1->type = 1; push(header, x1); } } }
public void eof_routine(struct token** header){
  struct token* x1; x1 = pop(header); int result = 0;
  if(*header != 0) {
    if(x1->type == 0) /* id */ x1->type == 1; /* F */
    struct token* x2; x2 = pop(header);
    if(x2->type == 5) /* * */ prod_sub_routine(header, x1, 0);
    else if(x2->type == 4) /* + */ plus_sub_routine(header, x1, 0);
    x1 = pop(header);  result = x1->val; smcoutput(result, 1); }
  else{ result = x1->val; smcoutput(result, 1); /*output the result */ } }
public int main() {
  private int ids[M]; public int expr[S];
  struct token *header = 0; //header of the stack
  public int index = 0; public int symbol = 0;
  smcinput(expr, 1, S); smcinput(ids, 1, M);
  while(index < S) {
    symbol = expr[index];
    if(symbol < K) /* id */ id_routine(&header, ids[symbol]);
    else if(symbol == K) /* + */ plus_routine(&header);
    else if(symbol == K+1) /* * */ prod_routine(&header);
    else if(symbol == K+2) /* ( */ lbra_routine(&header);
    else if(symbol == K+3) /* ) */ rbra_routine(&header);
    else if(symbol == K+4) /* EOF */ eof_routine(&header);
    index = index+1; }
  return 1;
}
```

Fig. 53. Benchmarking program: LR-parser.c (Part 2/2)

```
public int K=1000; /* length of input set */
public int main() {
  public int i; private int year1[K], year2[K]; private int final[K];
  smcinput(year1, 1, K); smcinput(year2, 1, K);
  for(i = 0; i < K; i++) { year2[i] = (year2[i] - year1[i]) * 1000; }
  for(i = 0; i < K; i++) { final[i] = year2[i] / year1[i]; }
  smcoutput(final, 1, K);
  return 0;
}
```

Fig. 54. Benchmarking program: h_analysis.c

```
public int main() {
   public int S = 100; private int A[S]; private int B[S];
   private int c; public int i, j;
   smcinput(A, 1, S); smcinput(B, 1, S);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = A[i]; } else{ c = B[i]; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] > B[i]){ c = A[i]; } else{ c = B[i]; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = B[i] - A[i]; } else{ c = A[i] - B[i]; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] > B[i]){ c = A[i] - B[i]; } else{ c = B[i] - A[i]; } }
   smcoutput(c, 1);
   for (i = 0; i < 1000; i++) {
      j = i%100;
      if (A[j] < B[j]){ c = A[j]; } else{ c = B[j]; } }
   smcoutput(c, 1);
   return 0;
}
```

Fig. 55. Benchmarking program: private-branching.c

```
public int main() {
   public int S=100; private int A[S];  private int B[S];
   private int c; public int i, j;
   smcinput(A, 1, S); smcinput(B, 1, S);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = A[i]; c = c + B[i]; c = c * 2; }
      else{ c = B[i]; c = c + B[i]; c = c + 2; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = A[i]; c = c + B[i]; c = c * 2; }
      else{ c = B[i]; c = c + B[i]; c = c + 2; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = A[i]; c = c + B[i]; c = c * 2; }
      else{ c = B[i]; c = c + B[i]; c = c + 2; } }
   smcoutput(c, 1);
   for (i = 0; i < S; i++) {
      if (A[i] < B[i]){ c = A[i]; c = c + B[i]; c = c * 2; }
      else{ c = B[i]; c = c + B[i]; c = c + 2; } }
   smcoutput(c, 1);
   for (i = 0; i < 1000; i++) {
      j = i%100;
      if (A[j] < B[j]){ c = A[j]; c = c + B[j]; c = c * 2; }
      else{ c = B[j]; c = c + B[j]; c = c + 2; } }
   smcoutput(c, 1);
   return 0;
}
```

Fig. 56. Benchmarking program: private-branching-mult.c

```
public int main() {
    public int S=100; private int A[S]; private int B[S];
    private int c, d; public int i, j;
    smcinput(A, 1, S); smcinput(B, 1, S);
    for (i = 0; i < S; i++) {
        if (A[i] < B[i]){
            c = A[i]; d = c; c = c + B[i]; d = d * c; c = c * 2; d = d + c; }
        else{
            c = B[i]; d = c; c = c + B[i]; d = d * c; c = c + 2; d = d + c; } }
    smcoutput(c, 1); smcoutput(d, 1);
    for (i = 0; i < S; i++) {
        if (A[i] < B[i]){
            c = A[i]; d = c; c = c + B[i]; d = d * c; c = c * 2; d = d + c; }
        else{
            c = B[i]; d = c; c = c + B[i]; d = d * c; c = c + 2; d = d + c; } }
    smcoutput(c, 1); smcoutput(d, 1);
    for (i = 0; i < S; i++) {
        if (A[i] < B[i]){
            c = A[i]; d = c; c = c + B[i]; d = d * c; c = c * 2; d = d + c; }
        else{
            c = B[i]; d = c; c = c + B[i]; d = d * c; c = c + 2; d = d + c; } }
    smcoutput(c, 1); smcoutput(d, 1);
    for (i = 0; i < S; i++) {
        if (A[i] < B[i]){
            c = A[i]; d = c; c = c + B[i]; d = d * c; c = c * 2; d = d + c; }
        else{
            c = B[i]; d = c; c = c + B[i]; d = d * c; c = c + 2; d = d + c; } }
    smcoutput(c, 1); smcoutput(d, 1);
    for (i = 0; i < 1000; i++) {
        j = i%100;
        if (A[j] < B[j]){
            c = A[j]; d = c; c = c + B[j]; d = d * c; c = c * 2; d = d + c; }
        else{
            c = B[j]; d = c; c = c + B[j]; d = d * c; c = c + 2; d = d + c; } }
    smcoutput(c, 1); smcoutput(d, 1);
    return 0;
}
```

Fig. 57. Benchmarking program: private-branching-add.c

```
public int main() {
    public int S=100; private int A[S]; private int B[S];
    private int c=0, d=0, e=0; public int i, j;
    smcinput(A, 1, S); smcinput(B, 1, S);
    for (i = 0; i < 100000; i++) {
        j = i%100;
        if (A[j] < B[j]){
            c = c + A[j]; e = e + c; d = d + c; e = e - 2;
            c = c + B[j]; e = e + d; d = d + c; e = e - c;
            c = c + 2; e = e - 2; d = d + c; e = e + 10;
            e = e - 100; e = e + d - c; }
        else{
            c = c + B[j]; e = e + c; d = d + c; e = e + d;
            c = c + B[j]; e = e - 50; d = d + c; e = e + e;
            c = c + 2; e = e - c - d; d = d + c; e = e + 10;
            e = e - 100; e = e + d - c; }
        if(e > 100000){ e = e - 100000; }
        if(i%50 == 0){ c = 0; d = 0; e = 0; } }
    smcoutput(c, 1); smcoutput(d, 1); smcoutput(e, 1);
    return 0;
}
```

Fig. 58. Benchmarking program: private-branching-reuse.c

Program Name	PICCO Average	PICCO Standard Deviation	SMC^2 Average	SMC^2 Standard Deviation
LR-parser	0.00226	0.00047	0.00222	0.00044
pay-gap	4.05632	0.08251	4.08879	0.13961
h_analysis	12.60051	0.14929	12.81269	0.22293
private-branching	1.67252	0.16551	1.57602	0.08988
private-branching-mult	1.89925	0.17203	1.61712	0.14109
private-branching-add	2.30731	0.09335	1.77433	0.1394
private-branching-reuse	307.72411	2.17444	207.99393	1.29311

Table 1. Average runtimes and standard deviation for local computation.

Program Name	PICCO Average	PICCO Standard Deviation	SMC^2 Average	SMC^2 Standard Deviation
LR-parser	0.00242	0.00029	0.00242	0.00019
pay-gap	13.86972	0.38221	14.30434	0.25417
h_analysis	33.17922	0.26027	33.16173	0.31844
private-branching	3.44979	0.06592	3.222	0.06049
private-branching-mult	4.56412	0.06466	3.23905	0.06333
private-branching-add	6.45718	0.02443	3.99943	0.14629
private-branching-reuse	923.30999	18.20752	470.81101	9.9084

Table 2. Average runtimes and standard deviation for distributed computation.

7.1 Runtime Averages and Standard Deviation

To calculate the averages and standard deviation, we first average the runtimes of each of the 3 parties in a single run (i.e., (Party3 + Party2 + Party1)/3). We then use the average timing for each run to obtain the total average and standard deviation for the runtime of each program. To calculate percent speedup with PICCO as the baseline, we used the formula: (PICCO avg - SMC^2 avg)/PICCO avg * 100. To calculate the standard deviation error bars, we used the formula: ((PICCO avg - (SMC^2 avg - SMC^2 st dev))/PICCO avg*100) - ((PICCO avg - SMC^2 avg)/PICCO avg * 100)

7.2 Local Runtimes

Table 3. h_analysis - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	12.6165	12.6182	12.6205	26	12.5379	12.539	12.5413
2	12.6407	12.6411	12.643	27	12.5548	12.5553	12.5582
3	12.669	12.6697	12.6712	28	12.5402	12.5409	12.5431
4	12.6986	12.6992	12.702	29	12.5705	12.5712	12.5729
5	12.9585	12.9592	12.9607	30	12.4891	12.4898	12.4921
6	12.5118	12.5127	12.515	31	12.511	12.5116	12.5142
7	12.5107	12.5113	12.5136	32	12.4431	12.4438	12.4454
8	12.5615	12.5621	12.5644	33	12.5129	12.5132	12.5156
9	12.6041	12.6049	12.6066	34	12.5426	12.5432	12.5446
10	12.5351	12.5357	12.5371	35	12.567	12.5674	12.569
11	12.715	12.7158	12.7175	36	12.5775	12.5781	12.5805
12	12.5866	12.5868	12.5885	37	12.5252	12.5259	12.5283
13	12.6791	12.6796	12.6811	38	12.6392	12.64	12.6414
14	12.5861	12.5865	12.5885	39	13.2066	13.2092	13.2113
15	12.5791	12.5807	12.5815	40	12.5044	12.505	12.5066
16	12.5046	12.5051	12.5069	41	12.4948	12.4952	12.4977
17	12.539	12.5396	12.5409	42	12.5765	12.5777	12.5789
18	12.5444	12.5448	12.5462	43	12.5128	12.5135	12.5156
19	12.5525	12.5532	12.5555	44	12.535	12.5358	12.538
20	12.5229	12.5235	12.525	45	13.1942	13.1948	13.197
21	12.575	12.5758	12.5778	46	12.5688	12.5692	12.5705
22	12.512	12.5126	12.5144	47	12.7549	12.7573	12.7592
23	12.5044	12.5053	12.5067	48	12.6145	12.6152	12.6166
24	12.5996	12.6005	12.603	49	12.5559	12.5566	12.5583
25	12.5209	12.5218	12.5241	50	12.6109	12.6116	12.6138

Table 4. h_analysis - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	12.8186	12.8203	12.8245	26	12.4757	12.4765	12.4777
2	12.8695	12.8724	12.8727	27	12.7187	12.721	12.7243
3	12.9447	12.9467	12.9501	28	12.5477	12.5487	12.5506
4	12.7745	12.7764	12.7794	29	13.0753	13.0778	13.0812
5	12.9614	12.9634	12.9664	30	12.9022	12.9048	12.9051
6	12.6944	12.6975	12.6977	31	12.551	12.5516	12.5535
7	12.6011	12.6032	12.6032	32	13.1789	13.1817	13.1821
8	12.6476	12.6498	12.6527	33	12.8397	12.8421	12.8449
9	12.89	12.8924	12.8958	34	13.2833	13.285	13.2879
10	12.9264	12.9273	12.9319	35	12.9896	12.993	12.9935
11	13.1416	13.1442	13.1444	36	13.0452	13.0475	13.0502
12	12.7686	12.7721	12.7708	37	12.7573	12.7599	12.7628
13	12.8705	12.873	12.8766	38	12.7752	12.7778	12.7781
14	12.8357	12.8382	12.8384	39	12.7908	12.7921	12.7955
15	12.8516	12.8545	12.8553	40	12.5684	12.5689	12.57
16	12.7758	12.7791	12.7777	41	13.063	13.0651	13.0677
17	12.7853	12.7878	12.7905	42	12.8728	12.8752	12.8781
18	13.0108	13.0138	13.0141	43	12.6476	12.6506	12.6513
19	13.4011	13.4038	13.407	44	12.5682	12.5687	12.5713
20	13.259	13.2618	13.2648	45	12.5392	12.5403	12.5426
21	12.9835	12.9867	12.9871	46	12.4598	12.4604	12.4627
22	12.5944	12.5972	12.5979	47	12.5042	12.5045	12.5066
23	12.7999	12.8027	12.8056	48	12.5861	12.5869	12.5893
24	12.798	12.8004	12.8031	49	12.5304	12.5316	12.534
25	12.6918	12.6953	12.6942	50	12.5667	12.5671	12.5685

Table 5. LR-parser - local PICCO

Run No.	Party 3	Party 2	Party 1	Run No. (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	$8.54 \cdot 10^{-4}$	$2.316 \cdot 10^{-3}$	$5.012 \cdot 10^{-3}$	26	$1.572 \cdot 10^{-3}$	$2.111 \cdot 10^{-3}$	$4.37 \cdot 10^{-3}$
2	$7.99 \cdot 10^{-4}$	$1.695 \cdot 10^{-3}$	$2.984 \cdot 10^{-3}$	27	$9.49 \cdot 10^{-4}$	$1.74 \cdot 10^{-3}$	$6.2 \cdot 10^{-3}$
3	$8.34 \cdot 10^{-4}$	$1.633 \cdot 10^{-3}$	$3.939 \cdot 10^{-3}$	28	$9.02 \cdot 10^{-4}$	$2.551 \cdot 10^{-3}$	$3.305 \cdot 10^{-3}$
4	$7.75 \cdot 10^{-4}$	$1.176 \cdot 10^{-3}$	$2.742 \cdot 10^{-3}$	29	$9.01 \cdot 10^{-4}$	$2.073 \cdot 10^{-3}$	$4.935 \cdot 10^{-3}$
5	$8.27 \cdot 10^{-4}$	$1.489 \cdot 10^{-3}$	$3.861 \cdot 10^{-3}$	30	$8.86 \cdot 10^{-4}$	$2.61 \cdot 10^{-3}$	$5.219 \cdot 10^{-3}$
6	$8.16 \cdot 10^{-4}$	$1.664 \cdot 10^{-3}$	$3.401 \cdot 10^{-3}$	31	$8.99 \cdot 10^{-4}$	$1.662 \cdot 10^{-3}$	$5.993 \cdot 10^{-3}$
7	$7.92 \cdot 10^{-4}$	$1.086 \cdot 10^{-3}$	$3.703 \cdot 10^{-3}$	32	$9.48 \cdot 10^{-4}$	$2.482 \cdot 10^{-3}$	$3.348 \cdot 10^{-3}$
8	$7.27 \cdot 10^{-4}$	$1.064 \cdot 10^{-3}$	$3.501 \cdot 10^{-3}$	33	$9.61 \cdot 10^{-4}$	$2.432 \cdot 10^{-3}$	$3.02 \cdot 10^{-3}$
9	$7.7 \cdot 10^{-4}$	$1.632 \cdot 10^{-3}$	$2.592 \cdot 10^{-3}$	34	$7.91 \cdot 10^{-4}$	$1.508 \cdot 10^{-3}$	$3.285 \cdot 10^{-3}$
10	$7.88 \cdot 10^{-4}$	$1.344 \cdot 10^{-3}$	$3.129 \cdot 10^{-3}$	35	$9.16 \cdot 10^{-4}$	$2.268 \cdot 10^{-3}$	$5.13 \cdot 10^{-3}$
11	$8.18 \cdot 10^{-4}$	$1.392 \cdot 10^{-3}$	$3.188 \cdot 10^{-3}$	36	$1.021 \cdot 10^{-3}$	$3.897 \cdot 10^{-3}$	$2.418 \cdot 10^{-3}$
12	$8.31 \cdot 10^{-4}$	$1.641 \cdot 10^{-3}$	$3.369 \cdot 10^{-3}$	37	$9.02 \cdot 10^{-4}$	$2.55 \cdot 10^{-3}$	$4.612 \cdot 10^{-3}$
13	$8.9 \cdot 10^{-4}$	$2.542 \cdot 10^{-3}$	$5.069 \cdot 10^{-3}$	38	$8.87 \cdot 10^{-4}$	$2.491 \cdot 10^{-3}$	$5.528 \cdot 10^{-3}$
14	$8.3 \cdot 10^{-4}$	$1.722 \cdot 10^{-3}$	$3.196 \cdot 10^{-3}$	39	$8.87 \cdot 10^{-4}$	$2.556 \cdot 10^{-3}$	$5.101 \cdot 10^{-3}$
15	$8.31 \cdot 10^{-4}$	$1.463 \cdot 10^{-3}$	$3.075 \cdot 10^{-3}$	40	$9.79 \cdot 10^{-4}$	$2.485 \cdot 10^{-3}$	$3.595 \cdot 10^{-3}$
16	$9.18 \cdot 10^{-4}$	$2.5 \cdot 10^{-3}$	$2.92 \cdot 10^{-3}$	41	$1.032 \cdot 10^{-3}$	$2.776 \cdot 10^{-3}$	$5.811 \cdot 10^{-3}$
17	$8.71 \cdot 10^{-4}$	$2.389 \cdot 10^{-3}$	$5.403 \cdot 10^{-3}$	42	$9.46 \cdot 10^{-4}$	$3.891 \cdot 10^{-3}$	$2.379 \cdot 10^{-3}$
18	$8.19 \cdot 10^{-4}$	$1.29 \cdot 10^{-3}$	$3.023 \cdot 10^{-3}$	43	$8 \cdot 10^{-4}$	$1.312 \cdot 10^{-3}$	$2.932 \cdot 10^{-3}$
19	$8.21 \cdot 10^{-4}$	$1.494 \cdot 10^{-3}$	$3.746 \cdot 10^{-3}$	44	$9.2 \cdot 10^{-4}$	$2.252 \cdot 10^{-3}$	$4.529 \cdot 10^{-3}$
20	$9.12 \cdot 10^{-4}$	$2.459 \cdot 10^{-3}$	$3.413 \cdot 10^{-3}$	45	$9.12 \cdot 10^{-4}$	$2.482 \cdot 10^{-3}$	$5.177 \cdot 10^{-3}$
21	$8.18 \cdot 10^{-4}$	$1.555 \cdot 10^{-3}$	$3.85 \cdot 10^{-3}$	46	$8.94 \cdot 10^{-4}$	$2.409 \cdot 10^{-3}$	$3.015 \cdot 10^{-3}$
22	$7.64 \cdot 10^{-4}$	$1.487 \cdot 10^{-3}$	$3.134 \cdot 10^{-3}$	47	$8.93 \cdot 10^{-4}$	$2.597 \cdot 10^{-3}$	$3.109 \cdot 10^{-3}$
23	$8.05 \cdot 10^{-4}$	$1.256 \cdot 10^{-3}$	$2.739 \cdot 10^{-3}$	48	$9.63 \cdot 10^{-4}$	$2.347 \cdot 10^{-3}$	$2.817 \cdot 10^{-3}$
24	$7.89 \cdot 10^{-4}$	$1.39 \cdot 10^{-3}$	$2.839 \cdot 10^{-3}$	49	$1.014 \cdot 10^{-3}$	$2.666 \cdot 10^{-3}$	$6.002 \cdot 10^{-3}$
25	$8.06 \cdot 10^{-4}$	$1.434 \cdot 10^{-3}$	$2.744 \cdot 10^{-3}$	50	$9.22 \cdot 10^{-4}$	$3.894 \cdot 10^{-3}$	$2.536 \cdot 10^{-3}$

Table 6. LR-parser - local SMC^2

Run No.	Party 3	Party 2	Party 1	Run No. (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	$8.71 \cdot 10^{-4}$	$1.145 \cdot 10^{-3}$	$3.156 \cdot 10^{-3}$	26	$8.31 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$3.705 \cdot 10^{-3}$
2	$9.39 \cdot 10^{-4}$	$2.686 \cdot 10^{-3}$	$3.339 \cdot 10^{-3}$	27	$7.83 \cdot 10^{-4}$	$1.69 \cdot 10^{-3}$	$2.561 \cdot 10^{-3}$
3	$8.77 \cdot 10^{-4}$	$2.452 \cdot 10^{-3}$	$5.342 \cdot 10^{-3}$	28	$7.53 \cdot 10^{-4}$	$1.155 \cdot 10^{-3}$	$3.258 \cdot 10^{-3}$
4	$8.84 \cdot 10^{-4}$	$2.273 \cdot 10^{-3}$	$2.63 \cdot 10^{-3}$	29	$8.12 \cdot 10^{-4}$	$1.665 \cdot 10^{-3}$	$3.065 \cdot 10^{-3}$
5	$9.01 \cdot 10^{-4}$	$2.452 \cdot 10^{-3}$	$5.315 \cdot 10^{-3}$	30	$7.8 \cdot 10^{-4}$	$1.332 \cdot 10^{-3}$	$3.652 \cdot 10^{-3}$
6	$1.003 \cdot 10^{-3}$	$3.679 \cdot 10^{-3}$	$2.311 \cdot 10^{-3}$	31	$7.86 \cdot 10^{-4}$	$1.475 \cdot 10^{-3}$	$3.993 \cdot 10^{-3}$
7	$9.09 \cdot 10^{-4}$	$2.376 \cdot 10^{-3}$	$5.328 \cdot 10^{-3}$	32	$7.71 \cdot 10^{-4}$	$1.214 \cdot 10^{-3}$	$2.527 \cdot 10^{-3}$
8	$9.48 \cdot 10^{-4}$	$2.487 \cdot 10^{-3}$	$2.941 \cdot 10^{-3}$	33	$8.32 \cdot 10^{-4}$	$1.611 \cdot 10^{-3}$	$3.107 \cdot 10^{-3}$
9	$8.99 \cdot 10^{-4}$	$2.349 \cdot 10^{-3}$	$3.401 \cdot 10^{-3}$	34	$8.19 \cdot 10^{-4}$	$1.365 \cdot 10^{-3}$	$3.329 \cdot 10^{-3}$
10	$1.225 \cdot 10^{-3}$	$4.193 \cdot 10^{-3}$	$2.377 \cdot 10^{-3}$	35	$8.44 \cdot 10^{-4}$	$1.51 \cdot 10^{-3}$	$4.705 \cdot 10^{-3}$
11	$8.88 \cdot 10^{-4}$	$2.421 \cdot 10^{-3}$	$4.982 \cdot 10^{-3}$	36	$8.78 \cdot 10^{-4}$	$1.842 \cdot 10^{-3}$	$3.954 \cdot 10^{-3}$
12	$9.07 \cdot 10^{-4}$	$1.575 \cdot 10^{-3}$	$5.926 \cdot 10^{-3}$	37	$8.51 \cdot 10^{-4}$	$1.525 \cdot 10^{-3}$	$3.25 \cdot 10^{-3}$
13	$9.02 \cdot 10^{-4}$	$2.34 \cdot 10^{-3}$	$5.113 \cdot 10^{-3}$	38	$8.99 \cdot 10^{-4}$	$2.663 \cdot 10^{-3}$	$4.528 \cdot 10^{-3}$
14	$9.49 \cdot 10^{-4}$	$2.139 \cdot 10^{-3}$	$5.352 \cdot 10^{-3}$	39	$7.82 \cdot 10^{-4}$	$8.51 \cdot 10^{-4}$	$3.009 \cdot 10^{-3}$
15	$9.29 \cdot 10^{-4}$	$2.621 \cdot 10^{-3}$	$3.371 \cdot 10^{-3}$	40	$7.99 \cdot 10^{-4}$	$1.503 \cdot 10^{-3}$	$3.539 \cdot 10^{-3}$
16	$8.9 \cdot 10^{-4}$	$2.363 \cdot 10^{-3}$	$6 \cdot 10^{-3}$	41	$8.42 \cdot 10^{-4}$	$1.464 \cdot 10^{-3}$	$3.614 \cdot 10^{-3}$
17	$8.82 \cdot 10^{-4}$	$2.386 \cdot 10^{-3}$	$5.253 \cdot 10^{-3}$	42	$8.21 \cdot 10^{-4}$	$1.842 \cdot 10^{-3}$	$3.921 \cdot 10^{-3}$
18	$9.13 \cdot 10^{-4}$	$3.613 \cdot 10^{-3}$	$2.008 \cdot 10^{-3}$	43	$7.43 \cdot 10^{-4}$	$1.336 \cdot 10^{-3}$	$3.164 \cdot 10^{-3}$
19	$9.78 \cdot 10^{-4}$	$2.622 \cdot 10^{-3}$	$5.543 \cdot 10^{-3}$	44	$8.52 \cdot 10^{-4}$	$1.627 \cdot 10^{-3}$	$2.923 \cdot 10^{-3}$
20	$8.43 \cdot 10^{-4}$	$2.472 \cdot 10^{-3}$	$3.559 \cdot 10^{-3}$	45	$8.29 \cdot 10^{-4}$	$1.585 \cdot 10^{-3}$	$3.94 \cdot 10^{-3}$
21	$8.53 \cdot 10^{-4}$	$1.528 \cdot 10^{-3}$	$5.406 \cdot 10^{-3}$	46	$7.95 \cdot 10^{-4}$	$1.371 \cdot 10^{-3}$	$2.803 \cdot 10^{-3}$
22	$9.08 \cdot 10^{-4}$	$2.544 \cdot 10^{-3}$	$5.216 \cdot 10^{-3}$	47	$8.23 \cdot 10^{-4}$	$1.489 \cdot 10^{-3}$	$3.662 \cdot 10^{-3}$
23	$9.14 \cdot 10^{-4}$	$3.811 \cdot 10^{-3}$	$2.401 \cdot 10^{-3}$	48	$8.2 \cdot 10^{-4}$	$1.478 \cdot 10^{-3}$	$2.972 \cdot 10^{-3}$
24	$8.45 \cdot 10^{-4}$	$1.225 \cdot 10^{-3}$	$3.31 \cdot 10^{-3}$	49	$7.57 \cdot 10^{-4}$	$1.175 \cdot 10^{-3}$	$3.16 \cdot 10^{-3}$
25	$9.78 \cdot 10^{-4}$	$2.569 \cdot 10^{-3}$	$2.828 \cdot 10^{-3}$	50	$7.5 \cdot 10^{-4}$	$1.28 \cdot 10^{-3}$	$3.948 \cdot 10^{-3}$

Table 7. pay-gap - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	4.16726	4.16811	4.16981	26	4.2061	4.20698	4.20958
2	4.03592	4.03685	4.03826	27	4.12068	4.12291	4.12347
3	4.00866	4.00958	4.01208	28	4.1028	4.10058	4.10424
4	4.10085	4.10164	4.10376	29	4.0136	4.01449	4.01568
5	4.04085	4.04344	4.04604	30	3.96217	3.96325	3.96492
6	3.9802	3.98091	3.98327	31	4.03994	4.04055	4.04318
7	4.01444	4.01522	4.01747	32	4.00828	4.0098	4.01079
8	4.29332	4.29734	4.29578	33	3.97393	3.97476	3.97659
9	4.05769	4.05859	4.06192	34	4.02941	4.0303	4.03258
10	4.02625	4.02711	4.02931	35	3.98304	3.98376	3.98556
11	3.99417	3.9953	4.00207	36	4	4.00049	4.00202
12	4.04705	4.04803	4.04939	37	3.98944	3.9902	3.99271
13	4.12716	4.12822	4.13046	38	4.17173	4.17255	4.17467
14	3.98831	3.98895	3.99048	39	4.13375	4.13451	4.13671
15	4.0011	4.00219	4.00437	40	3.98662	3.98736	3.98963
16	4.03571	4.03719	4.03813	41	4.02506	4.02599	4.02843
17	4.25825	4.25555	4.26014	42	3.98928	3.99069	3.99465
18	4.01612	4.01703	4.0182	43	4.19137	4.19228	4.19389
19	4.21264	4.21373	4.21821	44	4.04239	4.04339	4.04532
20	4.02595	4.027	4.02937	45	3.98701	3.98745	3.98949
21	4.17542	4.17704	4.18179	46	3.98489	3.98588	3.99115
22	4.0156	4.01614	4.01937	47	3.99005	3.99224	3.99289
23	3.98058	3.98066	3.98402	48	4.04459	4.04541	4.04779
24	4.00213	4.00046	4.00343	49	4.16325	4.16382	4.16617
25	4.00142	4.00205	4.00435	50	4.00166	4.00349	4.00453

Table 8. pay-gap - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	4.02504	4.02626	4.02898	26	3.97448	3.9751	3.97744
2	4.00674	4.00838	4.01023	27	4.49137	4.4936	4.49689
3	4.00143	4.00328	4.00424	28	4.12475	4.12648	4.12932
4	4.20375	4.20519	4.20611	29	3.98664	3.9884	3.98931
5	4.01353	4.01505	4.01574	30	3.98306	3.98355	3.98665
6	4.17787	4.17887	4.18324	31	3.99172	3.99308	3.99364
7	3.98043	3.98128	3.98279	32	4.27796	4.27945	4.27989
8	4.18946	4.19008	4.19118	33	4.14147	4.14232	4.1444
9	4.11967	4.12053	4.12294	34	4.20099	4.20122	4.20419
10	4.02879	4.02999	4.03219	35	4.02764	4.02862	4.03097
11	4.47848	4.48203	4.48278	36	4.01179	4.01307	4.01597
12	4.01288	4.01387	4.01608	37	3.98353	3.98402	3.98563
13	3.97971	3.98042	3.98265	38	4.02764	4.02832	4.03072
14	4.19594	4.19803	4.1989	39	3.98398	3.98509	3.98716
15	4.03561	4.03655	4.03919	40	3.99998	4.00196	4.00284
16	3.98955	3.9898	3.99255	41	4.10665	4.10772	4.11041
17	4.00117	4.00187	4.00328	42	4.18827	4.18885	4.19138
18	3.98528	3.98599	3.98754	43	3.97756	3.97819	3.98022
19	4.25372	4.25549	4.25623	44	3.98376	3.98494	3.98699
20	3.97173	3.97263	3.9748	45	4.11079	4.11221	4.11315
21	3.99446	3.99529	3.99759	46	4.03477	4.03561	4.03757
22	4.02002	4.02076	4.02276	47	4.22173	4.22256	4.22479
23	3.98496	3.98674	3.98779	48	4.15755	4.15995	4.16039
24	4.0055	4.00646	4.00781	49	4.16673	4.16786	4.16921
25	4.55908	4.56078	4.56227	50	3.99967	4.0005	4.00172

Table 9. private-branching - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	1.60458	1.60673	1.60834	26	1.61402	1.61473	1.61719
2	1.6533	1.65421	1.6568	27	1.60158	1.60215	1.60342
3	1.60857	1.60919	1.6108	28	1.60954	1.61027	1.61187
4	1.5971	1.59767	1.59945	29	1.60095	1.60162	1.60396
5	1.61263	1.6133	1.61579	30	1.83951	1.84268	1.84503
6	1.62257	1.62348	1.6255	31	1.56627	1.56726	1.56862
7	1.61156	1.6122	1.61395	32	1.607	1.60719	1.61004
8	1.59742	1.59801	1.60061	33	1.6172	1.61777	1.61957
9	1.60492	1.60566	1.60717	34	1.84574	1.84823	1.85089
10	1.60842	1.60911	1.61165	35	1.58155	1.5823	1.58442
11	1.59461	1.59533	1.59769	36	1.5946	1.59521	1.5974
12	2.50973	2.51158	2.5149	37	1.59218	1.59294	1.59525
13	1.584	1.58452	1.58694	38	1.59741	1.59851	1.59918
14	1.60348	1.60428	1.60652	39	1.60366	1.60427	1.60685
15	1.76747	1.7682	1.76988	40	1.92327	1.92631	1.92846
16	1.96118	1.96397	1.96693	41	1.59204	1.59228	1.59454
17	1.5961	1.59674	1.59823	42	1.94828	1.95088	1.95365
18	1.5945	1.59513	1.59753	43	1.66567	1.66617	1.66852
19	1.64438	1.64514	1.65158	44	1.61146	1.61214	1.61398
20	1.66879	1.66941	1.67101	45	1.63584	1.63656	1.63896
21	1.61062	1.61118	1.61233	46	1.59177	1.59239	1.59357
22	2.04844	2.05102	2.05405	47	1.6045	1.60493	1.6065
23	1.61313	1.61384	1.61556	48	1.79401	1.79672	1.79906
24	1.59406	1.59463	1.59695	49	1.60696	1.60738	1.60991
25	1.59437	1.59497	1.59651	50	1.60545	1.60631	1.60769

Table 10. private-branching - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	1.54662	1.54731	1.54858	26	1.55277	1.55355	1.55581
2	1.54512	1.54577	1.54682	27	1.55109	1.55155	1.55298
3	1.55057	1.5512	1.55271	28	1.53933	1.53991	1.54191
4	1.52629	1.52701	1.52838	29	1.54686	1.54776	1.55025
5	1.53156	1.53222	1.5341	30	1.76529	1.76749	1.77031
6	1.55063	1.5514	1.55305	31	1.52424	1.52501	1.52687
7	1.55285	1.5535	1.55526	32	1.53305	1.53401	1.5359
8	1.55436	1.55503	1.5573	33	1.53941	1.53998	1.54147
9	1.8771	1.8785	1.88182	34	1.5467	1.54761	1.54911
10	1.51997	1.52075	1.5232	35	1.53822	1.53879	1.54017
11	1.59211	1.59217	1.59526	36	1.53811	1.53882	1.54105
12	1.5399	1.54064	1.54209	37	1.5332	1.53388	1.53615
13	1.5393	1.53956	1.54156	38	1.54169	1.54203	1.5446
14	1.60499	1.6059	1.60759	39	1.53514	1.53569	1.53741
15	1.55078	1.55152	1.55306	40	1.52526	1.52616	1.52859
16	1.53605	1.53667	1.53897	41	1.55529	1.55616	1.5583
17	1.54284	1.54336	1.54498	42	1.52545	1.52637	1.52794
18	1.84004	1.84267	1.84534	43	1.54622	1.54679	1.54858
19	1.53181	1.53255	1.53451	44	1.55675	1.55736	1.55886
20	1.53991	1.54024	1.54194	45	1.54968	1.55074	1.55217
21	1.54858	1.54927	1.55104	46	1.88024	1.88338	1.88447
22	1.72932	1.73221	1.73412	47	1.53542	1.53611	1.53888
23	1.53651	1.537	1.53847	48	1.72987	1.73224	1.7347
24	1.53265	1.53333	1.53584	49	1.53644	1.5371	1.53974
25	1.55363	1.55451	1.55675	50	1.53938	1.53983	1.5423

Table 11. private-branching-mult - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	1.84452	1.84614	1.84859	26	1.99169	1.99413	1.99673
2	1.83161	1.83239	1.83416	27	1.78829	1.78899	1.79113
3	1.82864	1.82927	1.83088	28	1.82606	1.82748	1.82819
4	1.83207	1.8329	1.83461	29	1.81855	1.81883	1.82111
5	1.82922	1.83012	1.83177	30	1.82258	1.82325	1.82468
6	2.17623	2.17756	2.18199	31	1.82914	1.82949	1.83157
7	1.81076	1.81217	1.81256	32	2.52867	2.53119	2.53375
8	2.1886	2.19121	2.19377	33	1.81997	1.82122	1.82293
9	1.82277	1.82319	1.82446	34	1.82738	1.82803	1.83074
10	1.8289	1.82947	1.83144	35	1.8117	1.81202	1.81471
11	1.84167	1.84242	1.8436	36	1.92061	1.92316	1.92566
12	1.8225	1.82295	1.82549	37	1.80583	1.80639	1.80898
13	1.83336	1.83418	1.83658	38	2.13714	2.14014	2.14237
14	1.83057	1.83099	1.83256	39	1.81543	1.81637	1.81789
15	1.83686	1.83733	1.83899	40	1.82036	1.82112	1.82356
16	1.85853	1.85929	1.86167	41	1.82171	1.82248	1.82504
17	1.81368	1.81429	1.81695	42	2.16868	2.16983	2.17338
18	2.07964	2.08242	2.08509	43	1.80742	1.80765	1.81031
19	1.81677	1.81699	1.81945	44	2.27346	2.27592	2.27898
20	2.48197	2.48377	2.48751	45	1.81042	1.8112	1.81293
21	1.82953	1.83019	1.83176	46	1.81737	1.81804	1.82065
22	1.83899	1.83977	1.84206	47	1.8322	1.83295	1.83462
23	1.81361	1.81446	1.81593	48	1.80715	1.80719	1.80937
24	1.81658	1.81801	1.81881	49	1.82041	1.82082	1.8234
25	1.81737	1.81791	1.8204	50	1.84488	1.84581	1.84804

Table 12. private-branching-mult - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	1.54888	1.55031	1.55182	26	1.52296	1.52361	1.5261
2	1.53811	1.53893	1.54046	27	1.5476	1.54816	1.54963
3	1.53257	1.53305	1.53573	28	1.55751	1.55837	1.56068
4	1.54548	1.54625	1.54842	29	1.54656	1.54736	1.54994
5	1.54048	1.54082	1.54309	30	1.69967	1.70265	1.7036
6	1.53612	1.53677	1.53841	31	1.53887	1.53953	1.54117
7	1.52219	1.52293	1.52515	32	1.67955	1.68211	1.6845
8	1.90041	1.90295	1.90517	33	1.55447	1.55541	1.55708
9	1.54033	1.54103	1.54261	34	1.53531	1.53588	1.53827
10	1.59069	1.59132	1.59357	35	1.71574	1.71853	1.7193
11	1.77185	1.77268	1.77516	36	1.55839	1.5608	1.56344
12	2.08521	2.08763	2.09016	37	1.52407	1.52591	1.52682
13	1.53254	1.53304	1.53565	38	1.52493	1.52541	1.52702
14	1.64102	1.64349	1.6463	39	1.61137	1.61196	1.6138
15	1.52872	1.52959	1.53131	40	1.72207	1.72284	1.72529
16	1.54642	1.5472	1.54897	41	1.56052	1.56134	1.56352
17	1.54477	1.54589	1.54836	42	1.81293	1.81597	1.81702
18	1.80777	1.81025	1.81294	43	1.97124	1.97382	1.97671
19	1.52321	1.52405	1.52658	44	1.78799	1.79075	1.79322
20	1.52073	1.52143	1.52325	45	1.52358	1.52413	1.52567
21	1.53302	1.53371	1.53533	46	1.5596	1.56031	1.56216
22	1.53418	1.53491	1.53636	47	1.54804	1.54877	1.55009
23	1.53334	1.53404	1.53558	48	2.02455	2.02757	2.02838
24	1.53236	1.53317	1.53466	49	1.53037	1.53106	1.53258
25	1.53389	1.53415	1.53668	50	1.65964	1.66256	1.66333

Table 13. private-branching-add - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	2.26137	2.26344	2.26575	26	2.2357	2.23646	2.23913
2	2.25126	2.25196	2.25415	27	2.34608	2.34681	2.34867
3	2.26869	2.26954	2.27113	28	2.2449	2.24554	2.24794
4	2.37157	2.37416	2.3764	29	2.25011	2.2508	2.25272
5	2.25388	2.25427	2.257	30	2.25984	2.26068	2.26289
6	2.26388	2.26453	2.26589	31	2.34647	2.34694	2.34923
7	2.2479	2.24844	2.25111	32	2.32354	2.32579	2.32836
8	2.44936	2.45201	2.45472	33	2.26515	2.26598	2.26776
9	2.26814	2.26872	2.27139	34	2.25612	2.25678	2.25834
10	2.28338	2.28404	2.28663	35	2.24158	2.2422	2.2445
11	2.29533	2.29539	2.29759	36	2.61099	2.61364	2.61595
12	2.25524	2.2563	2.259	37	2.28469	2.28557	2.28706
13	2.27186	2.27259	2.27411	38	2.38805	2.39068	2.39333
14	2.25764	2.25817	2.26003	39	2.22541	2.22597	2.22799
15	2.24657	2.24718	2.24961	40	2.51652	2.51919	2.52146
16	2.52695	2.52984	2.53208	41	2.25918	2.2602	2.26283
17	2.28396	2.28441	2.28616	42	2.25602	2.25717	2.25856
18	2.26472	2.26531	2.26676	43	2.25852	2.25908	2.26142
19	2.28065	2.28115	2.28294	44	2.26348	2.26391	2.26521
20	2.41586	2.41889	2.41946	45	2.25589	2.25649	2.25918
21	2.23359	2.23398	2.23586	46	2.34781	2.35	2.35257
22	2.46324	2.46554	2.46761	47	2.25329	2.25419	2.25654
23	2.2536	2.25417	2.25589	48	2.2537	2.25424	2.2566
24	2.25771	2.25838	2.26094	49	2.25657	2.25756	2.25943
25	2.28734	2.28779	2.28942	50	2.57972	2.58237	2.58505

Table 14. private-branching-add - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	1.71235	1.71382	1.71662	26	1.87633	1.87756	1.8802
2	1.72306	1.72404	1.72604	27	1.71121	1.71202	1.71425
3	1.71693	1.71758	1.72048	28	2.28842	2.29075	2.29354
4	1.84594	1.84914	1.85015	29	1.98787	1.99013	1.99183
5	1.69609	1.69681	1.69823	30	1.71027	1.71093	1.7134
6	1.71023	1.71102	1.71333	31	1.69438	1.69494	1.69622
7	1.69963	1.70073	1.70256	32	1.99784	2.00022	2.00298
8	1.81254	1.81504	1.81743	33	1.69397	1.69459	1.6958
9	1.69577	1.6965	1.69884	34	1.70427	1.70499	1.70739
10	1.9957	1.99863	1.99932	35	1.70912	1.70961	1.71124
11	1.71234	1.71335	1.71481	36	1.73849	1.73924	1.7406
12	1.70743	1.70784	1.71001	37	1.71039	1.71123	1.71276
13	1.70228	1.70313	1.70442	38	1.88031	1.88292	1.88553
14	1.70913	1.70986	1.71147	39	1.69504	1.69572	1.69726
15	1.70737	1.70801	1.70997	40	1.6934	1.69383	1.69548
16	2.02438	2.02725	2.02912	41	1.7109	1.71165	1.71303
17	1.69039	1.69109	1.69275	42	1.69498	1.69654	1.69744
18	1.69969	1.70007	1.70276	43	1.71914	1.7199	1.72149
19	1.71531	1.71623	1.71845	44	1.73577	1.73649	1.73809
20	2.27585	2.27866	2.28155	45	1.79852	1.79906	1.80076
21	1.71745	1.71798	1.71927	46	1.71324	1.71384	1.71557
22	1.93455	1.93817	1.93695	47	1.70883	1.70936	1.71105
23	1.70552	1.70621	1.70856	48	1.71332	1.71389	1.71615
24	1.70485	1.7057	1.70729	49	1.7139	1.71435	1.71633
25	1.70303	1.70357	1.7053	50	1.73054	1.7309	1.73261

Table 15. private-branching-reuse - local PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	306.417	306.417	306.42	26	308.3	308.301	308.303
2	306.77	306.77	306.773	27	307.684	307.685	307.688
3	307.605	307.605	307.608	28	311.649	311.652	311.655
4	304.854	304.854	304.856	29	304.103	304.104	304.106
5	305.65	305.651	305.653	30	307.001	307.001	307.003
6	313.356	313.356	313.359	31	307.276	307.277	307.279
7	307.517	307.517	307.519	32	306.671	306.671	306.674
8	306.909	306.91	306.912	33	306.545	306.546	306.548
9	308.475	308.477	308.478	34	305.21	305.211	305.213
10	307.584	307.585	307.588	35	306.935	306.936	306.939
11	308.927	308.928	308.93	36	306.058	306.058	306.06
12	305.307	305.308	305.31	37	305.447	305.447	305.449
13	307.22	307.221	307.224	38	309.23	309.23	309.233
14	307.052	307.053	307.055	39	305.702	305.703	305.705
15	306.521	306.522	306.524	40	309.179	309.18	309.182
16	313.535	313.536	313.536	41	308.917	308.917	308.919
17	309.687	309.687	309.69	42	306.335	306.335	306.337
18	309.937	309.937	309.939	43	306.904	306.906	306.906
19	312.982	312.984	312.985	44	305.18	305.181	305.184
20	309.028	309.027	309.029	45	308.632	308.633	308.634
21	307.059	307.059	307.061	46	305.768	305.769	305.771
22	306.067	306.068	306.069	47	310.487	310.488	310.49
23	309.403	309.404	309.406	48	306.675	306.675	306.678
24	308.003	308.004	308.006	49	307.25	307.251	307.254
25	311.32	311.32	311.323	50	305.823	305.824	305.826

Table 16. private-branching-reuse - local SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	207.435	207.436	207.438	26	208.61	208.611	208.614
2	208.437	208.438	208.442	27	207.952	207.953	207.955
3	207.562	207.562	207.564	28	206.238	206.239	206.24
4	208.017	208.018	208.02	29	207.875	207.876	207.878
5	207.633	207.635	207.636	30	208.124	208.125	208.127
6	208.643	208.645	208.646	31	206.764	206.765	206.767
7	206.995	206.995	206.997	32	213.974	213.976	213.977
8	208.24	208.241	208.243	33	207.685	207.686	207.688
9	207.078	207.078	207.081	34	207.504	207.505	207.506
10	208.841	208.842	208.843	35	207.304	207.305	207.306
11	206.712	206.713	206.714	36	206.047	206.047	206.05
12	209.619	209.619	209.621	37	206.517	206.518	206.519
13	207.887	207.887	207.89	38	208.409	208.41	208.411
14	208.898	208.898	208.901	39	209.372	209.372	209.375
15	208.852	208.853	208.855	40	207.55	207.55	207.553
16	209.493	209.494	209.496	41	209.026	209.027	209.029
17	207.297	207.298	207.299	42	207.617	207.617	207.619
18	206.614	206.615	206.617	43	208.531	208.532	208.533
19	209.137	209.138	209.14	44	206.267	206.267	206.27
20	206.691	206.691	206.693	45	207.775	207.775	207.778
21	207.515	207.516	207.517	46	206.919	206.922	206.923
22	210.254	210.255	210.257	47	208.3	208.301	208.303
23	208.206	208.207	208.209	48	209.102	209.103	209.105
24	206.548	206.548	206.551	49	208.143	208.145	208.148
25	207.497	207.498	207.5	50	207.93	207.931	207.932

Table 17. h_analysis - distributed PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	32.9724	32.9741	32.974	26	33.5028	33.5037	33.5039
2	33.003	33.0038	33.0038	27	33.3172	33.3185	33.3193
3	33.1318	33.1323	33.1327	28	32.8595	32.8605	32.8608
4	33.0414	33.0425	33.0429	29	33.2181	33.2191	33.2192
5	33.1214	33.1224	33.1228	30	32.8554	32.8566	32.8564
6	33.0819	33.0831	33.0846	31	33.5673	33.5682	33.5691
7	33.2351	33.2362	33.2367	32	32.668	32.6691	32.6698
8	32.6875	32.6882	32.6885	33	33.3845	33.3856	33.3862
9	33.0159	33.0171	33.0174	34	33.4602	33.4609	33.461
10	33.3818	33.3826	33.3836	35	33.3772	33.3784	33.3791
11	33.4762	33.4771	33.4776	36	32.809	32.8104	32.811
12	33.0908	33.0904	33.0908	37	33.5681	33.5695	33.5702
13	32.9965	32.9971	32.9977	38	32.9796	32.9805	32.9809
14	33.4771	33.4784	33.4783	39	32.9324	32.9333	32.9336
15	33.2027	33.2043	33.2043	40	33.0995	33.1009	33.1015
16	33.1458	33.1465	33.1469	41	33.4864	33.4872	33.4876
17	33.0859	33.087	33.0874	42	33.0941	33.095	33.0954
18	33.2989	33.2999	33.2999	43	33.4355	33.4369	33.4379
19	33.3585	33.3594	33.3601	44	32.764	32.7646	32.7649
20	33.4771	33.4781	33.4782	45	32.9461	32.9474	32.9473
21	32.5989	32.5988	32.5992	46	33.1884	33.1895	33.19
22	33.2016	33.2024	33.2031	47	33.3549	33.3554	33.3559
23	33.4653	33.4665	33.4667	48	33.1047	33.1055	33.1057
24	33.4956	33.497	33.4975	49	33.4986	33.4993	33.5002
25	33.5036	33.5048	33.5059	50	32.9027	32.9039	32.9047

7.3 Distributed Runtimes

Table 18. h_analysis - distributed SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	33.386	33.3859	33.3858	26	33.4063	33.4057	33.406
2	33.4402	33.4409	33.441	27	32.8752	32.876	32.876
3	33.0142	33.0152	33.0157	28	33.3966	33.3981	33.398
4	33.304	33.3052	33.3056	29	33.5832	33.5847	33.5851
5	33.4974	33.4984	33.4991	30	33.3178	33.3189	33.3192
6	32.7267	32.7272	32.7281	31	33.377	33.3777	33.3783
7	33.2915	33.2938	33.293	32	33.2906	33.2918	33.2921
8	33.2545	33.256	33.2564	33	33.0039	33.0044	33.0059
9	33.3282	33.3296	33.3298	34	33.4027	33.4044	33.4042
10	32.4965	32.4973	32.5	35	33.5612	33.5625	33.5626
11	33.2348	33.2357	33.2357	36	33.3462	33.3479	33.3485
12	32.4525	32.4532	32.4534	37	33.2213	33.2239	33.2245
13	32.5411	32.5419	32.5428	38	32.7314	32.7322	32.7333
14	33.2799	33.2807	33.2825	39	33.525	33.5262	33.527
15	33.2421	33.2439	33.2444	40	33.0647	33.066	33.0659
16	33.2257	33.2272	33.2276	41	33.4969	33.4981	33.4987
17	33.6459	33.6471	33.6472	42	33.6321	33.633	33.6344
18	32.4917	32.4932	32.4936	43	32.9909	32.9921	32.9929
19	32.839	32.8403	32.8407	44	33.2335	33.2346	33.2348
20	33.284	33.2849	33.2864	45	33.086	33.0868	33.0877
21	33.3813	33.3825	33.3828	46	32.9811	32.9821	32.9822
22	33.2197	33.2208	33.222	47	33.0257	33.0263	33.0269
23	32.839	32.8383	32.8382	48	33.1959	33.1973	33.1965
24	33.3357	33.3365	33.3374	49	32.792	32.7926	32.7939
25	33.2647	33.2659	33.2659	50	32.4894	32.4902	32.4912

Table 19. LR-parser - distributed PICCO

Run No.	Party 3	Party 2	Party 1	Run No. (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	$1.549 \cdot 10^{-3}$	$3.097 \cdot 10^{-3}$	$3.742 \cdot 10^{-3}$	26	$1.228 \cdot 10^{-3}$	$2.165 \cdot 10^{-3}$	$3.021 \cdot 10^{-3}$
2	$1.574 \cdot 10^{-3}$	$2.668 \cdot 10^{-3}$	$2.459 \cdot 10^{-3}$	27	$1.608 \cdot 10^{-3}$	$2.521 \cdot 10^{-3}$	$3.243 \cdot 10^{-3}$
3	$1.594 \cdot 10^{-3}$	$2.662 \cdot 10^{-3}$	$2.632 \cdot 10^{-3}$	28	$1.488 \cdot 10^{-3}$	$2.998 \cdot 10^{-3}$	$3.607 \cdot 10^{-3}$
4	$1.609 \cdot 10^{-3}$	$2.899 \cdot 10^{-3}$	$2.222 \cdot 10^{-3}$	29	$1.666 \cdot 10^{-3}$	$2.615 \cdot 10^{-3}$	$3.443 \cdot 10^{-3}$
5	$1.584 \cdot 10^{-3}$	$3.201 \cdot 10^{-3}$	$3.635 \cdot 10^{-3}$	30	$1.196 \cdot 10^{-3}$	$2.203 \cdot 10^{-3}$	$3.062 \cdot 10^{-3}$
6	$1.497 \cdot 10^{-3}$	$2.546 \cdot 10^{-3}$	$2.317 \cdot 10^{-3}$	31	$1.588 \cdot 10^{-3}$	$2.649 \cdot 10^{-3}$	$2.558 \cdot 10^{-3}$
7	$1.539 \cdot 10^{-3}$	$2.773 \cdot 10^{-3}$	$3.946 \cdot 10^{-3}$	32	$1.245 \cdot 10^{-3}$	$2.054 \cdot 10^{-3}$	$3.131 \cdot 10^{-3}$
8	$1.739 \cdot 10^{-3}$	$2.73 \cdot 10^{-3}$	$3.145 \cdot 10^{-3}$	33	$1.723 \cdot 10^{-3}$	$2.394 \cdot 10^{-3}$	$2.611 \cdot 10^{-3}$
9	$1.508 \cdot 10^{-3}$	$2.747 \cdot 10^{-3}$	$2.376 \cdot 10^{-3}$	34	$1.257 \cdot 10^{-3}$	$2.056 \cdot 10^{-3}$	$2.666 \cdot 10^{-3}$
10	$1.529 \cdot 10^{-3}$	$2.863 \cdot 10^{-3}$	$2.891 \cdot 10^{-3}$	35	$1.655 \cdot 10^{-3}$	$2.45 \cdot 10^{-3}$	$2.621 \cdot 10^{-3}$
11	$1.523 \cdot 10^{-3}$	$2.528 \cdot 10^{-3}$	$3.607 \cdot 10^{-3}$	36	$1.627 \cdot 10^{-3}$	$2.73 \cdot 10^{-3}$	$3.777 \cdot 10^{-3}$
12	$1.507 \cdot 10^{-3}$	$2.286 \cdot 10^{-3}$	$2.583 \cdot 10^{-3}$	37	$1.598 \cdot 10^{-3}$	$2.868 \cdot 10^{-3}$	$3.194 \cdot 10^{-3}$
13	$1.482 \cdot 10^{-3}$	$2.596 \cdot 10^{-3}$	$2.993 \cdot 10^{-3}$	38	$1.164 \cdot 10^{-3}$	$1.121 \cdot 10^{-3}$	$1.113 \cdot 10^{-3}$
14	$1.619 \cdot 10^{-3}$	$2.295 \cdot 10^{-3}$	$3.279 \cdot 10^{-3}$	39	$1.556 \cdot 10^{-3}$	$2.399 \cdot 10^{-3}$	$2.723 \cdot 10^{-3}$
15	$1.46 \cdot 10^{-3}$	$2.817 \cdot 10^{-3}$	$3.066 \cdot 10^{-3}$	40	$1.538 \cdot 10^{-3}$	$2.593 \cdot 10^{-3}$	$2.758 \cdot 10^{-3}$
16	$1.591 \cdot 10^{-3}$	$2.88 \cdot 10^{-3}$	$2.949 \cdot 10^{-3}$	41	$1.624 \cdot 10^{-3}$	$2.681 \cdot 10^{-3}$	$2.761 \cdot 10^{-3}$
17	$1.586 \cdot 10^{-3}$	$2.81 \cdot 10^{-3}$	$2.955 \cdot 10^{-3}$	42	$1.536 \cdot 10^{-3}$	$2.343 \cdot 10^{-3}$	$3.304 \cdot 10^{-3}$
18	$1.583 \cdot 10^{-3}$	$2.686 \cdot 10^{-3}$	$3.654 \cdot 10^{-3}$	43	$1.618 \cdot 10^{-3}$	$2.886 \cdot 10^{-3}$	$2.842 \cdot 10^{-3}$
19	$1.584 \cdot 10^{-3}$	$2.731 \cdot 10^{-3}$	$3.259 \cdot 10^{-3}$	44	$1.587 \cdot 10^{-3}$	$2.269 \cdot 10^{-3}$	$2.522 \cdot 10^{-3}$
20	$1.541 \cdot 10^{-3}$	$2.752 \cdot 10^{-3}$	$3.583 \cdot 10^{-3}$	45	$1.687 \cdot 10^{-3}$	$3.383 \cdot 10^{-3}$	$3.675 \cdot 10^{-3}$
21	$1.719 \cdot 10^{-3}$	$2.697 \cdot 10^{-3}$	$3.675 \cdot 10^{-3}$	46	$1.66 \cdot 10^{-3}$	$3.083 \cdot 10^{-3}$	$3.629 \cdot 10^{-3}$
22	$1.607 \cdot 10^{-3}$	$2.661 \cdot 10^{-3}$	$2.966 \cdot 10^{-3}$	47	$1.664 \cdot 10^{-3}$	$3.295 \cdot 10^{-3}$	$3.371 \cdot 10^{-3}$
23	$1.597 \cdot 10^{-3}$	$2.556 \cdot 10^{-3}$	$2.753 \cdot 10^{-3}$	48	$1.537 \cdot 10^{-3}$	$2.805 \cdot 10^{-3}$	$3.419 \cdot 10^{-3}$
24	$1.561 \cdot 10^{-3}$	$2.754 \cdot 10^{-3}$	$3.218 \cdot 10^{-3}$	49	$1.697 \cdot 10^{-3}$	$2.794 \cdot 10^{-3}$	$3.796 \cdot 10^{-3}$
25	$1.51 \cdot 10^{-3}$	$2.876 \cdot 10^{-3}$	$2.876 \cdot 10^{-3}$	50	$1.559 \cdot 10^{-3}$	$2.7 \cdot 10^{-3}$	$3.388 \cdot 10^{-3}$

Table 20. LR-parser - distributed SMC^2

Run No.	Party 3	Party 2	Party 1	Run No. (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	$1.637 \cdot 10^{-3}$	$2.625 \cdot 10^{-3}$	$3.257 \cdot 10^{-3}$	26	$1.691 \cdot 10^{-3}$	$2.589 \cdot 10^{-3}$	$2.947 \cdot 10^{-3}$
2	$1.567 \cdot 10^{-3}$	$2.633 \cdot 10^{-3}$	$3.69 \cdot 10^{-3}$	27	$1.616 \cdot 10^{-3}$	$2.265 \cdot 10^{-3}$	$3.365 \cdot 10^{-3}$
3	$1.491 \cdot 10^{-3}$	$2.16 \cdot 10^{-3}$	$2.374 \cdot 10^{-3}$	28	$1.436 \cdot 10^{-3}$	$2.379 \cdot 10^{-3}$	$2.555 \cdot 10^{-3}$
4	$1.565 \cdot 10^{-3}$	$2.612 \cdot 10^{-3}$	$2.544 \cdot 10^{-3}$	29	$1.662 \cdot 10^{-3}$	$3.476 \cdot 10^{-3}$	$3.694 \cdot 10^{-3}$
5	$1.618 \cdot 10^{-3}$	$2.542 \cdot 10^{-3}$	$3.21 \cdot 10^{-3}$	30	$1.488 \cdot 10^{-3}$	$2.437 \cdot 10^{-3}$	$3.049 \cdot 10^{-3}$
6	$1.477 \cdot 10^{-3}$	$2.852 \cdot 10^{-3}$	$2.89 \cdot 10^{-3}$	31	$1.515 \cdot 10^{-3}$	$2.998 \cdot 10^{-3}$	$3.454 \cdot 10^{-3}$
7	$1.62 \cdot 10^{-3}$	$2.481 \cdot 10^{-3}$	$2.512 \cdot 10^{-3}$	32	$1.537 \cdot 10^{-3}$	$2.58 \cdot 10^{-3}$	$3.648 \cdot 10^{-3}$
8	$1.577 \cdot 10^{-3}$	$2.879 \cdot 10^{-3}$	$3.007 \cdot 10^{-3}$	33	$1.526 \cdot 10^{-3}$	$2.6 \cdot 10^{-3}$	$3.531 \cdot 10^{-3}$
9	$1.462 \cdot 10^{-3}$	$2.566 \cdot 10^{-3}$	$2.138 \cdot 10^{-3}$	34	$1.56 \cdot 10^{-3}$	$2.723 \cdot 10^{-3}$	$3.289 \cdot 10^{-3}$
10	$1.533 \cdot 10^{-3}$	$2.772 \cdot 10^{-3}$	$3.091 \cdot 10^{-3}$	35	$1.546 \cdot 10^{-3}$	$2.668 \cdot 10^{-3}$	$2.603 \cdot 10^{-3}$
11	$1.524 \cdot 10^{-3}$	$2.36 \cdot 10^{-3}$	$2.444 \cdot 10^{-3}$	36	$1.528 \cdot 10^{-3}$	$2.727 \cdot 10^{-3}$	$3.226 \cdot 10^{-3}$
12	$1.553 \cdot 10^{-3}$	$2.868 \cdot 10^{-3}$	$2.984 \cdot 10^{-3}$	37	$1.608 \cdot 10^{-3}$	$2.723 \cdot 10^{-3}$	$2.855 \cdot 10^{-3}$
13	$1.717 \cdot 10^{-3}$	$2.877 \cdot 10^{-3}$	$2.968 \cdot 10^{-3}$	38	$1.564 \cdot 10^{-3}$	$2.645 \cdot 10^{-3}$	$2.646 \cdot 10^{-3}$
14	$1.549 \cdot 10^{-3}$	$2.622 \cdot 10^{-3}$	$2.733 \cdot 10^{-3}$	39	$1.642 \cdot 10^{-3}$	$2.64 \cdot 10^{-3}$	$3.251 \cdot 10^{-3}$
15	$1.62 \cdot 10^{-3}$	$2.809 \cdot 10^{-3}$	$3.237 \cdot 10^{-3}$	40	$1.502 \cdot 10^{-3}$	$2.829 \cdot 10^{-3}$	$2.936 \cdot 10^{-3}$
16	$1.602 \cdot 10^{-3}$	$2.23 \cdot 10^{-3}$	$3.178 \cdot 10^{-3}$	41	$1.564 \cdot 10^{-3}$	$2.362 \cdot 10^{-3}$	$3.332 \cdot 10^{-3}$
17	$1.554 \cdot 10^{-3}$	$2.687 \cdot 10^{-3}$	$2.697 \cdot 10^{-3}$	42	$1.551 \cdot 10^{-3}$	$2.969 \cdot 10^{-3}$	$2.876 \cdot 10^{-3}$
18	$1.538 \cdot 10^{-3}$	$2.568 \cdot 10^{-3}$	$3.231 \cdot 10^{-3}$	43	$1.516 \cdot 10^{-3}$	$2.174 \cdot 10^{-3}$	$3.008 \cdot 10^{-3}$
19	$1.559 \cdot 10^{-3}$	$2.369 \cdot 10^{-3}$	$2.405 \cdot 10^{-3}$	44	$1.577 \cdot 10^{-3}$	$2.618 \cdot 10^{-3}$	$3.239 \cdot 10^{-3}$
20	$1.59 \cdot 10^{-3}$	$2.871 \cdot 10^{-3}$	$3.28 \cdot 10^{-3}$	45	$1.656 \cdot 10^{-3}$	$2.653 \cdot 10^{-3}$	$3.531 \cdot 10^{-3}$
21	$1.581 \cdot 10^{-3}$	$2.597 \cdot 10^{-3}$	$2.555 \cdot 10^{-3}$	46	$1.573 \cdot 10^{-3}$	$2.582 \cdot 10^{-3}$	$3.645 \cdot 10^{-3}$
22	$1.525 \cdot 10^{-3}$	$2.47 \cdot 10^{-3}$	$2.604 \cdot 10^{-3}$	47	$1.593 \cdot 10^{-3}$	$2.875 \cdot 10^{-3}$	$3.258 \cdot 10^{-3}$
23	$1.622 \cdot 10^{-3}$	$2.545 \cdot 10^{-3}$	$2.641 \cdot 10^{-3}$	48	$1.47 \cdot 10^{-3}$	$2.418 \cdot 10^{-3}$	$3.786 \cdot 10^{-3}$
24	$1.464 \cdot 10^{-3}$	$2.367 \cdot 10^{-3}$	$2.57 \cdot 10^{-3}$	49	$1.541 \cdot 10^{-3}$	$2.741 \cdot 10^{-3}$	$3.348 \cdot 10^{-3}$
25	$1.584 \cdot 10^{-3}$	$3.247 \cdot 10^{-3}$	$3.428 \cdot 10^{-3}$	50	$1.543 \cdot 10^{-3}$	$2.614 \cdot 10^{-3}$	$3.615 \cdot 10^{-3}$

Table 21. private-branching - distributed PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	3.23942	3.24051	3.24039	26	3.49143	3.49223	3.49284
2	3.32584	3.32801	3.32975	27	3.40712	3.4081	3.40841
3	3.41976	3.42104	3.42105	28	3.45702	3.45831	3.45812
4	3.50588	3.5709	3.50791	29	3.47067	3.4716	3.47144
5	3.50504	3.50583	3.50598	30	3.45716	3.45846	3.45848
6	3.52268	3.52391	3.52445	31	3.47034	3.47139	3.4716
7	3.49831	3.49998	3.50061	32	3.32684	3.32811	3.32818
8	3.48104	3.48227	3.48231	33	3.44954	3.45095	3.45088
9	3.27109	3.27205	3.2731	34	3.51742	3.5183	3.51939
10	3.47149	3.47318	3.4733	35	3.45803	3.45918	3.45967
11	3.4677	3.46963	3.4693	36	3.45175	3.45274	3.45274
12	3.4115	3.41241	3.41247	37	3.43281	3.43406	3.43469
13	3.50748	3.50928	3.50926	38	3.46394	3.46513	3.46535
14	3.46141	3.46275	3.46261	39	3.53129	3.533	3.53338
15	3.48426	3.48535	3.48518	40	3.52613	3.52715	3.52741
16	3.41861	3.41867	3.41885	41	3.48972	3.49049	3.49174
17	3.52237	3.52339	3.52409	42	3.33954	3.34034	3.33998
18	3.4747	3.47588	3.47688	43	3.47977	3.48063	3.48168
19	3.41949	3.42051	3.42104	44	3.44981	3.45097	3.45144
20	3.42541	3.42669	3.42736	45	3.42624	3.42774	3.42787
21	3.41429	3.4158	3.41606	46	3.32197	3.32286	3.32288
22	3.51975	3.52094	3.52186	47	3.51539	3.51679	3.51722
23	3.40489	3.40664	3.40703	48	3.51441	3.5156	3.5164
24	3.4831	3.48456	3.48458	49	3.456	3.4572	3.45723
25	3.43877	3.44023	3.44059	50	3.42339	3.42473	3.42506

Table 22. private-branching - distributed SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	3.22575	3.22693	3.22709	26	3.23015	3.23186	3.23219
2	3.31135	3.31285	3.31392	27	3.1856	3.18664	3.18778
3	3.32549	3.32621	3.32626	28	3.22072	3.22191	3.22287
4	3.1409	3.14259	3.14264	29	3.12446	3.12533	3.12534
5	3.2253	3.22638	3.22636	30	3.28129	3.2827	3.28344
6	3.19258	3.19365	3.19423	31	3.21136	3.21253	3.21259
7	3.19228	3.19378	3.19397	32	3.30145	3.30228	3.3024
8	3.25889	3.25975	3.26027	33	3.24651	3.2475	3.24837
9	3.21114	3.21212	3.21235	34	3.25778	3.2588	3.25932
10	3.20388	3.20535	3.2059	35	3.1906	3.19143	3.19146
11	3.27287	3.27393	3.27411	36	3.2563	3.25724	3.25729
12	3.27767	3.27873	3.2786	37	3.23832	3.23925	3.23916
13	3.30338	3.30437	3.30485	38	3.20474	3.20554	3.20572
14	3.31036	3.31107	3.31126	39	3.18841	3.1877	3.18864
15	3.16673	3.1676	3.16863	40	3.33595	3.33666	3.33635
16	3.1691	3.16895	3.16904	41	3.16652	3.16739	3.16768
17	3.11065	3.11214	3.11259	42	3.17713	3.17823	3.17821
18	3.18197	3.18278	3.18289	43	3.318	3.31935	3.31969
19	3.24375	3.24505	3.24465	44	3.18529	3.18608	3.18621
20	3.22024	3.22139	3.22159	45	3.20002	3.20121	3.2019
21	3.24179	3.24286	3.24307	46	3.18166	3.18285	3.18282
22	3.33262	3.3337	3.33387	47	3.13168	3.13296	3.13328
23	3.20144	3.20242	3.20316	48	3.16449	3.16536	3.1655
24	3.18528	3.18648	3.18624	49	3.06241	3.06366	3.06418
25	3.23825	3.23958	3.24017	50	3.25584	3.25689	3.25775

Table 23. private-branching-mult - distributed PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	4.5954	4.59636	4.59609	26	4.55368	4.55424	4.55463
2	4.53936	4.54049	4.54061	27	4.57286	4.57407	4.57444
3	4.49363	4.4952	4.49554	28	4.60919	4.61025	4.61068
4	4.63247	4.63376	4.63421	29	4.57066	4.57152	4.57266
5	4.52342	4.52411	4.52453	30	4.64601	4.64708	4.64726
6	4.56561	4.56762	4.56772	31	4.54558	4.54722	4.54768
7	4.43305	4.4341	4.43538	32	4.59367	4.59467	4.59476
8	4.59511	4.59691	4.59689	33	4.53832	4.53918	4.53935
9	4.57623	4.57736	4.57834	34	4.46108	4.46304	4.46307
10	4.63623	4.63711	4.63736	35	4.51477	4.51649	4.51689
11	4.60049	4.60142	4.6022	36	4.59608	4.59698	4.59723
12	4.61652	4.61733	4.61768	37	4.48141	4.48268	4.48337
13	4.55563	4.55672	4.5565	38	4.55729	4.55807	4.55816
14	4.60324	4.6042	4.6045	39	4.57702	4.57796	4.57899
15	4.26586	4.26708	4.2672	40	4.53622	4.53781	4.53826
16	4.53426	4.53562	4.53601	41	4.65083	4.65278	4.65319
17	4.55882	4.55996	4.56028	42	4.50843	4.51003	4.51003
18	4.58667	4.58789	4.58838	43	4.61518	4.61689	4.61719
19	4.59861	4.59996	4.60039	44	4.58877	4.58969	4.5902
20	4.61689	4.6177	4.61797	45	4.61282	4.61464	4.61505
21	4.53529	4.53596	4.53716	46	4.54268	4.54413	4.54483
22	4.62299	4.62452	4.62489	47	4.51636	4.51743	4.5182
23	4.56215	4.56327	4.56317	48	4.61769	4.6186	4.61878
24	4.58665	4.58748	4.58826	49	4.63943	4.64074	4.64127
25	4.5051	4.50637	4.50617	50	4.57378	4.57488	4.57506

Table 24. private-branching-mult - distributed SMC^2

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	3.24871	3.2485	3.2485	26	3.25063	3.25161	3.25164
2	3.24513	3.24615	3.24697	27	3.24884	3.25005	3.25041
3	3.28705	3.28819	3.28874	28	3.26558	3.26676	3.26762
4	3.19546	3.1964	3.19654	29	3.14865	3.14964	3.14947
5	3.30502	3.30589	3.30737	30	3.26928	3.27028	3.27031
6	3.20291	3.20383	3.2039	31	3.18859	3.18982	3.1898
7	3.26045	3.26134	3.26235	32	3.32418	3.32516	3.3262
8	3.23805	3.23976	3.24205	33	3.16827	3.16913	3.16982
9	3.22116	3.22245	3.22218	34	3.14683	3.14786	3.1478
10	3.30338	3.30437	3.30457	35	3.14419	3.14557	3.14431
11	3.23186	3.23358	3.23359	36	3.21007	3.21171	3.21177
12	3.24383	3.24506	3.24505	37	3.28256	3.28375	3.28489
13	3.26867	3.26974	3.27076	38	3.16202	3.16304	3.16409
14	3.2274	3.2285	3.22928	39	3.26626	3.26717	3.26719
15	3.14338	3.14479	3.14475	40	3.25889	3.26021	3.26068
16	3.26569	3.26706	3.26746	41	3.20518	3.20634	3.2062
17	3.19437	3.19544	3.19594	42	3.31639	3.31774	3.31825
18	3.20916	3.21031	3.21105	43	3.20719	3.20798	3.20822
19	3.24551	3.24647	3.24649	44	3.28656	3.2877	3.28863
20	3.37681	3.37665	3.37682	45	3.23714	3.23846	3.23915
21	3.0765	3.07712	3.07828	46	3.39525	3.396	3.39599
22	3.3742	3.37544	3.37491	47	3.27075	3.27159	3.27197
23	3.2233	3.22408	3.2251	48	3.21416	3.21503	3.21609
24	3.29998	3.30118	3.30176	49	3.14558	3.14653	3.14661
25	3.18802	3.18935	3.18907	50	3.22141	3.22269	3.22296

Table 25. private-branching-add - distributed PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	6.50164	6.50246	6.50214	26	6.42595	6.42746	6.42788
2	6.46684	6.46768	6.46832	27	6.44219	6.44334	6.44332
3	6.48319	6.48395	6.48499	28	6.452	6.45313	6.45325
4	6.42252	6.42351	6.42375	29	6.45469	6.45614	6.45663
5	6.46481	6.46628	6.46677	30	6.42916	6.43043	6.43107
6	6.51813	6.51912	6.51919	31	6.43679	6.43779	6.43815
7	6.41184	6.41273	6.41274	32	6.48524	6.4866	6.48672
8	6.46118	6.46196	6.46297	33	6.46485	6.46631	6.46651
9	6.48374	6.48493	6.48537	34	6.47884	6.48058	6.48092
10	6.46173	6.46315	6.46359	35	6.37378	6.37457	6.37482
11	6.44803	6.44916	6.4491	36	6.45143	6.45277	6.45247
12	6.46533	6.46643	6.46651	37	6.43697	6.43808	6.43825
13	6.44199	6.44286	6.44395	38	6.46466	6.46566	6.4655
14	6.4664	6.46746	6.46753	39	6.45191	6.4533	6.45383
15	6.45817	6.45885	6.45899	40	6.45636	6.45735	6.45731
16	6.45497	6.4563	6.45621	41	6.44296	6.44427	6.44446
17	6.47309	6.47411	6.47428	42	6.42655	6.42765	6.42843
18	6.46976	6.47101	6.4716	43	6.46664	6.46785	6.46804
19	6.47669	6.4779	6.47804	44	6.46296	6.46393	6.46425
20	6.44402	6.44509	6.44497	45	6.44887	6.44998	6.45031
21	6.46258	6.46305	6.46305	46	6.47456	6.47553	6.47668
22	6.45184	6.45304	6.45379	47	6.46136	6.4623	6.46228
23	6.46708	6.46874	6.46888	48	6.43963	6.44074	6.44077
24	6.50002	6.50121	6.50155	49	6.47579	6.47713	6.47777
25	6.43453	6.43648	6.43651	50	6.42132	6.4228	6.42314

Table 26. private-branching-add - distributed SMC[2]

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	4.17418	4.17424	4.1745	26	4.01589	4.01801	4.01814
2	3.7459	3.74765	3.74947	27	4.02534	4.02631	4.02747
3	3.75282	3.75458	3.75771	28	4.13796	4.13845	4.13946
4	3.82356	3.82493	3.82896	29	3.93888	3.93997	3.9409
5	4.0707	4.07152	4.07156	30	4.00427	4.0055	4.00619
6	4.10426	4.10537	4.10543	31	4.16683	4.16785	4.16876
7	4.07842	4.07957	4.07937	32	4.05642	4.0582	4.05857
8	4.11001	4.11096	4.11143	33	4.1546	4.15534	4.15575
9	4.01142	4.01319	4.01349	34	4.15907	4.15994	4.16018
10	4.11668	4.11746	4.11801	35	4.01624	4.01715	4.01773
11	4.06778	4.06862	4.06961	36	4.00861	4.00983	4.01022
12	4.14263	4.14362	4.14347	37	3.97886	3.97984	3.98024
13	4.04563	4.04723	4.04756	38	4.11109	4.11238	4.1124
14	4.10021	4.10129	4.10145	39	3.73384	3.73468	3.73571
15	3.91254	3.91383	3.91392	40	3.7755	3.77636	3.77728
16	4.21228	4.21303	4.21319	41	3.99583	3.99726	3.99769
17	3.9684	3.96974	3.97054	42	4.0246	4.02564	4.02683
18	4.01983	4.02107	4.02158	43	4.03986	4.04157	4.04184
19	4.07483	4.07623	4.07665	44	4.14853	4.14932	4.14955
20	4.12336	4.12429	4.12477	45	4.09817	4.09904	4.09903
21	4.03838	4.03942	4.03948	46	3.75002	3.75129	3.75145
22	3.95919	3.96049	3.96082	47	3.72462	3.72539	3.72574
23	4.00141	4.00303	4.00309	48	3.66927	3.6702	3.67034
24	3.99121	3.99227	3.99317	49	3.69157	3.69257	3.69338
25	4.11988	4.12121	4.12112	50	3.7326	3.73357	3.73455

Table 27. private-branching-reuse - distributed PICCO

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	939.677	939.678	939.678	26	930.345	930.347	930.347
2	934.2	934.201	934.201	27	929.639	929.641	929.641
3	933.701	933.702	933.702	28	946.8	946.802	946.802
4	933.659	933.659	933.657	29	930.576	930.578	930.578
5	933.478	933.478	933.477	30	932.871	932.872	932.873
6	933.456	933.457	933.457	31	931.209	931.21	921.211
7	933.273	933.273	933.274	32	933.872	933.873	933.874
8	930.833	930.834	930.833	33	936.157	936.159	936.159
9	930.503	930.504	930.504	34	918.741	918.742	918.742
10	929.933	929.934	929.934	35	917.047	917.048	917.049
11	893.004	893.005	893.006	36	927.901	927.902	927.902
12	883.523	883.524	883.523	37	929.557	929.559	929.559
13	883.316	883.317	883.318	38	922.454	922.455	922.455
14	882.261	882.262	882.262	39	928.38	928.381	928.382
15	879.77	879.771	879.771	40	929.934	929.935	929.936
16	878.815	878.816	878.817	41	928.381	928.383	928.383
17	874.693	874.694	874.694	42	929.074	929.075	929.075
18	934.262	934.262	934.262	43	926.697	926.698	926.699
19	896.204	896.206	896.206	44	924.68	924.681	924.682
20	933.213	933.214	933.215	45	929.556	929.557	929.557
21	934.8	934.801	934.801	46	925.54	925.542	925.542
22	930.751	930.752	930.753	47	928.33	928.331	928.332
23	939.329	939.329	939.336	48	924.925	924.917	924.917
24	931.335	931.336	931.336	49	934.215	934.217	934.218
25	934.438	934.439	934.44	50	929.488	929.489	929.489

Table 28. private-branching-reuse - distributed SMC^2

Run	Party 3	Party 2	Party 1	Run (Cont.)	Party 3 (Cont.)	Party 2 (Cont.)	Party 1 (Cont.)
1	475.252	475.252	475.252	26	480.441	480.442	480.443
2	474.749	474.75	474.751	27	477.851	477.852	477.853
3	475.626	475.627	475.628	28	477.094	477.095	477.095
4	456.563	456.564	456.564	29	485.52	485.521	485.521
5	450.563	450.651	450.661	30	450.194	450.196	450.195
6	471.822	471.824	471.835	31	450.494	450.496	450.496
7	471.068	471.069	471.08	32	449.176	449.176	449.177
8	448.641	448.643	448.654	33	452.379	452.38	452.381
9	452.754	452.756	452.767	34	472.001	472.001	472.002
10	452.021	452.023	452.042	35	477.727	477.728	477.73
11	458.783	458.785	458.796	36	478.685	478.685	478.687
12	465.188	465.19	465.211	37	476.482	476.483	476.483
13	474.854	474.855	474.855	38	473.843	473.843	473.844
14	478.646	478.647	478.648	39	476.136	476.137	476.138
15	475.822	475.824	475.825	40	476.402	476.404	476.404
16	474.708	474.709	474.709	41	475.359	475.36	475.36
17	472.542	472.543	472.544	42	478.351	478.352	478.352
18	475.325	475.326	475.326	43	477.333	477.334	477.334
19	474.326	474.327	474.328	44	475.617	475.618	475.619
20	474.09	474.091	474.092	45	475.808	475.81	475.81
21	474.348	474.349	474.349	46	474.918	474.919	474.92
22	473.485	473.486	473.486	47	476.794	476.795	476.796
23	471.181	471.182	471.183	48	474.985	474.986	474.986
24	477.147	477.148	477.148	49	475.471	475.472	475.473
25	476.871	476.872	476.872	50	474.977	474.978	474.978

REFERENCES

[1] R. Gennaro, M. Rabin, and T. Rabin. 1998. Simplified VSS and Fast-Track Multiparty Computations with Applications to Threshold Cryptography. In *ACM Symposium on Principles of Distributed Computing (PODC)*. 101–111.
[2] A. Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.