## A. Security Verification

The proof of Theorem 1 is by induction on the length of `is`, which is assumed to be finite, and uses the techniques established in previous work [15, 16] in which by-contruction properties of operations on layered state monads (e.g., `K`) are used to prove the equality. The three principal properties used are *atomic noninterference*, computational innocence, and the *clobber* rule. We describe these properties informally as the technical details may be found in the aforementioned articles.

A layered state monad is a monad constructed from multiple applications of the state monad transformer. The monad `K`, for example, is the result of three applications of state monad transformer to the identity monad:

```
type K         = (StT SharedReg
                   (StT CPUState
                    (StT CPUState I)))
```

Atomic noninterference formalizes the notion that operations (i.e., atoms) lifted from distinct layers in a layered commute (i.e., do not interfere) with the monadic bind operator. Computational innocence shows how computations that are side-effect free (i.e., "innocent" computations) may be added to other computations preserving equality. For example, for the `get` operation defined by `StT`, $\texttt{get} >> \varphi = \varphi$ for any computation $\varphi$. Finally, the "clobber rule" shows that operations within the same state layer may be cancelled out—i.e., clobbered. For example in `K`, we defined $\texttt{mask}_\texttt{H}$ as:

```
maskH :: K ()
maskH = liftKH (update (const s0))
   where s0 = undefined
```

By the clobber rule, $\texttt{liftKH}\ \varphi >> \texttt{maskH} = \texttt{maskH} = \texttt{liftKH}\ \gamma >> \texttt{maskH}$ for any appropriately typed $\varphi$ and $\gamma$.

Additionally, the "monad laws" [22] are also applied extensively throughout the verification. These are:

$$
\begin{array}{lll}
\texttt{return v} >>= \texttt{f} & = \texttt{f v} & \text{— left unit}\\
\texttt{x} >>= \texttt{return} & = \texttt{x} & \text{— right unit}\\
(\texttt{x} >>= \lambda\texttt{v}.\ \texttt{y}) >>= \lambda\texttt{w}.\ \texttt{z} & = \texttt{x} >>= \lambda\texttt{v}.\ (\texttt{y} >>= \lambda\texttt{w}.\ \texttt{z}) & \text{— associativity}
\end{array}
$$

The proof of Theorem 1 follows the pattern, illustrated below. In the informal sketch below, we do some violence to the syntax in order to provide the reader with a roadmap to the proof of Theorem 1. The first step unrolls the operation of (`harness lo hi`) into a sequence of operations, $lh_i$, which combine actions from both `lo` and `hi` and their operations on the shared register layer. The idempotence of $\texttt{mask}_\texttt{H}$ is used to clone it and associativity is used to move $\texttt{mask}_\texttt{H}$ to the right of $lh_n$. The clobber rule is used to cancel `hi`'s actions, producing $l_n$ whose actions consist only of `lo`'s and `lo`'s writes to the shared register. $\texttt{mask}_\texttt{H}$ commutes with $\texttt{l}_\texttt{n}$ and this clobber-then-commute pattern is repeated until all of `hi`'s effects have been cancelled. Then, the cloned $\texttt{mask}_\texttt{H}$ may be "backed out" and removed by its idempotence. The result is equal to the r.h.s. of Theorem 1.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\textit{pull os}\ [i_1,\ldots,i_n]\ (\textit{harness lo hi}) >>= \lambda \textit{os}.\ \textit{mask}_H >> \texttt{return}\ \textit{os}}\\
\quad = (\texttt{lh}_1 ;\ldots; \texttt{lh}_\texttt{n}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— } \textit{mask}_H \text{ idempotent}\\
\quad = (\texttt{lh}_1 ;\ldots; \texttt{lh}_\texttt{n} ; \texttt{mask}_\texttt{H}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— assoc.}\\
\quad = (\texttt{lh}_1 ;\ldots; \texttt{l}_\texttt{n} ; \texttt{mask}_\texttt{H}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— clobber}\\
\quad = (\texttt{lh}_1 ;\ldots; \texttt{mask}_\texttt{H} ; \texttt{l}_\texttt{n}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— atomic nonint.}\\
\quad = (\texttt{lh}_1 ; \texttt{mask}_\texttt{H} ; \ldots; \texttt{l}_\texttt{n}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— atomic nonint.}\\
\quad = (\texttt{l}_1 ; \texttt{mask}_\texttt{H} ; \ldots; \texttt{l}_\texttt{n}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— clobber}\\
\quad = (\texttt{l}_1 ; \ldots; \texttt{l}_\texttt{n}) >>= \lambda\texttt{os}.\ \texttt{mask}_\texttt{H} >> \texttt{return os} & \text{— "reversing previous steps"}\\
\quad = \textit{pull os}\ [i_1,\ldots,i_n]\ (\textit{harness lo}\ (\textit{skip}\ o_0\ i_0)) >>= \lambda \textit{os}.\ \textit{mask}_H >> \texttt{return}\ \textit{os} &
\end{array}
$$

The remainder of this appendix consists of the following. Section A.1 discusses lemmas which simplify the proof of Theorem 1. These lemmas follow by routine, if somewhat laborious, application and simplification of the definitions of the harness. We include the proof of Lemma 4 which is the most complex of the lemmas. Section A.2 contains the proof of Theorem 1. Section B presents the proof of Lemma 4.

### A.1 Lemmas

This section presents four lemmas used to prove Theorem 1. Each of them involves unfolding definitions from the harness and simplifying using the monad laws, $\beta$-reduction, etc. The proof of Lemma 4 is presented in Section B.

Lemma 1 unwinds the definition of `pull` on an $n$ length input list into $n$ calls to `next`.

**Lemma 1** (pull)**.** *Given $\varphi$ and os of appropriate type. For every* $\texttt{n} \in \mathbb{N}$*,*

$$
\begin{array}{rlll}
\textit{pull os}\ [i_1,\ldots,i_n]\ \varphi = & \textit{next}\,\varphi & >>= & \lambda \textit{Right}(o_1,\kappa_1).\\
& \textit{next}(\kappa_1\ i_1) & >>= & \lambda \textit{Right}(o_2,\kappa_2).\\
& \vdots & & \\
& \textit{next}(\kappa_{n-1}\ i_{n-1}) & >>= & \lambda \textit{Right}(o_n,\kappa_n).\\
& \multicolumn{3}{l}{\texttt{return}(\textit{os} +\!\!+ [\textit{fst}\ o_1,\ldots,\textit{fst}\ o_n])}
\end{array}
$$

□

Lemma 2 formulates the interaction of `next` with `harness`.

**Lemma 2** (*next ∘ harness*)**.** *For any appropriately typed* `hi` *and* `lo`

$$
\begin{array}{l}
\mathit{next}\ (\mathit{harness}\ \mathit{lo}\ \mathit{hi}) \\
\quad = \ (\mathit{lift}\ .\ \mathit{lift}^{l}_{K})\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}, \kappa^{l}). \\
\qquad\ (\mathit{lift}\ .\ \mathit{lift}^{h}_{K})\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}, \kappa^{h}) \\
\qquad\ \mathbf{let} \\
\qquad\quad \mathtt{f}\ =\ \lambda(\mathtt{i}^{\mathtt{l}}, \mathtt{i}^{\mathtt{h}}).\ \mathit{checkHiPort}\ i^{h}\ o^{h} \quad \texttt{>>=} \quad \lambda \hat{i}^{h}. \\
\qquad\qquad\qquad\qquad\quad \mathit{checkLoPort}\ o^{l} \quad \texttt{>>} \\
\qquad\qquad\qquad\qquad\quad \mathit{harness}\ (\kappa^{l}\ i^{l})\ (\kappa^{h}\ \hat{i}^{h}) \\
\qquad\ \mathbf{in} \\
\qquad\ \mathtt{return}\ (\mathtt{Right}\ ((\mathtt{o}^{\mathtt{l}}, \mathtt{o}^{\mathtt{h}}), \mathtt{f}))
\end{array}
$$

□

Lemma 3 formulates the interaction between `next` and the `lift` for the `ReT` monad transformer. N.b., `next` behaves as a kind of inverse or project for that `lift`.

**Lemma 3** (next ∘ lift)**.** *The following holds.*

$$\mathtt{next}\ (\mathtt{lift}\ \mathtt{x} \texttt{ >>= } \mathtt{f}) = \mathtt{x} \texttt{ >>= } \mathtt{next}\ .\ \mathtt{f}$$

□

Lemma 4 captures the interaction of `pull` with `harness` in which a call to `pull` on `harness` is reduced to a (co)recursive call.

**Lemma 4** (pull ∘ harness)**.** *For appropriately typed os, hi and lo, and assuming WLOG that* $i_1 = (i^{l}_{1}, i^{h}_{1})$,

$$
\begin{array}{l}
\mathit{pull}\ \mathit{os}\ [i_1, \ldots, i_n]\ (\mathit{harness}\ \mathit{lo}\ \mathit{hi}) \\
\quad = \ \mathit{lift}^{l}_{K}\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}_{1}, \kappa^{l}). \\
\qquad\ \mathit{lift}^{h}_{K}\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}_{1}, \kappa^{h}). \\
\qquad\ \mathit{chkHPrt}\ i^{h}_{1}\ o^{h}_{1} \quad \texttt{>>=} \quad \lambda \hat{i}^{h}. \\
\qquad\ \mathit{chkLPrt}\ o^{l}_{1} \quad \texttt{>>} \\
\qquad\ \mathit{pull}\ (\mathit{os} +\!\!+ [o^{l}_{1}])\ [i_2, \ldots, i_n]\ (\kappa^{l}\ i^{l}_{1})\ (\kappa^{h}\ \hat{i}^{h})
\end{array}
$$

□

### A.2 Theorem 1

*Proof.* Proof of Theorem 1.

$$
\begin{array}{l}
\mathit{pull}\ \mathit{os}\ ((i^{l}_{1}, i^{h}_{1}) : \mathit{is})\ (\mathit{harness}\ \mathit{lo}\ \mathit{hi}) \texttt{ >>= } \lambda v.\ \mathit{mask}_{H} \texttt{ >> } \mathtt{return}\ v \\
\{\mathit{Lemma\ 4.}\} \\
\quad = \ \mathit{lift}^{l}_{K}\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}_{1}, \kappa^{l}). \\
\qquad\ \mathit{lift}^{h}_{K}\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}_{1}, \kappa^{h}). \\
\qquad\ \mathit{chkHPrt}\ i^{h}\ o^{h}_{1} \quad \texttt{>>=} \quad \lambda \hat{i}^{h}. \\
\qquad\ \mathit{chkLPrt}\ o^{l}_{1} \quad \texttt{>>} \\
\qquad\ \mathit{pull}\ (\mathit{os} +\!\!+ [o^{l}_{1}])\ [i_2, \ldots, i_n]\ (\kappa^{l}\ i^{l})\ (\kappa^{h}\ \hat{i}^{h}) \quad \texttt{>>=} \quad \lambda v.\ \mathit{mask}_{H} \texttt{ >> } \mathtt{return}\ v \\
\{\mathit{Induction\ hypothesis.}\} \\
\quad = \ \mathit{lift}^{l}_{K}\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}_{1}, \kappa^{l}). \\
\qquad\ \mathit{lift}^{h}_{K}\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}_{1}, \kappa^{h}). \\
\qquad\ \mathit{chkHPrt}\ i^{h}\ o^{h}_{1} \quad \texttt{>>=} \quad \lambda \hat{i}^{h}. \\
\qquad\ \mathit{chkLPrt}\ o^{l}_{1} \quad \texttt{>>} \\
\qquad\ \mathit{pull}\ (\mathit{os} +\!\!+ [o^{l}_{1}])\ [i_2, \ldots, i_n]\ (\kappa^{l}\ i^{l})\ (\mathit{skip}\ o_0\ \hat{i}^{h}) \quad \texttt{>>=} \quad \lambda v.\ \mathit{mask}_{H} \texttt{ >> } \mathtt{return}\ v \\
\{\mathit{Defn.\ skip.}\} \\
\quad = \ \mathit{lift}^{l}_{K}\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}_{1}, \kappa^{l}). \\
\qquad\ \mathit{lift}^{h}_{K}\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}_{1}, \kappa^{h}). \\
\qquad\ \mathit{chkHPrt}\ i^{h}\ o^{h}_{1} \quad \texttt{>>=} \quad \lambda \hat{i}^{h}. \\
\qquad\ \mathit{chkLPrt}\ o^{l}_{1} \quad \texttt{>>} \\
\qquad\ \mathit{pull}\ (\mathit{os} +\!\!+ [o^{l}_{1}])\ [i_2, \ldots, i_n]\ (\kappa^{l}\ i^{l})\ (\mathit{skip}\ o_0\ i_0) \quad \texttt{>>=} \quad \lambda v.\ \mathit{mask}_{H} \texttt{ >> } \mathtt{return}\ v \\
\\
\{\mathit{Defn.\ chkHPrt,\ innocence.}\} \\
\quad = \ \mathit{lift}^{l}_{K}\ (\mathit{next}\ \mathit{lo}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{l}_{1}, \kappa^{l}). \\
\qquad\ \mathit{lift}^{h}_{K}\ (\mathit{next}\ \mathit{hi}) \quad \texttt{>>=} \quad \lambda \mathit{Right}\ (o^{h}_{1}, \kappa^{h}). \\
\qquad\ \mathit{chkLPrt}\ o^{l}_{1} \quad \texttt{>>} \\
\qquad\ \mathit{pull}\ (\mathit{os} +\!\!+ [o^{l}_{1}])\ [i_2, \ldots, i_n]\ (\kappa^{l}\ i^{l})\ (\mathit{skip}\ o_0\ i_0) \quad \texttt{>>=} \quad \lambda v.\ \mathit{mask}_{H} \texttt{ >> } \mathtt{return}\ v \\
\{\mathit{Defn.} \texttt{ >> };\ o^{h}_{1}, \kappa^{h}\ \mathit{free.}\}
\end{array}
$$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ \mathit{hi}) & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{\mathit{mask}_H$ *idempotent*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ \mathit{hi}) & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Consequence of atomic noninterference*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ \mathit{hi}) & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{mask}_H & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Consequence of atomic noninterference*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ \mathit{hi}) & \mathtt{>>} & \\
\quad \mathit{mask}_H & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Consequence of clobber*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>} & \\
\quad \mathit{mask}_H & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Reversing previous steps*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>} & \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Defn.* $\mathtt{>>}$; $o^h_1, \kappa^h$ *free*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^h_1, \kappa^h). \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Consequence of innocence*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^h_1, \kappa^h). \\
\quad \mathit{chkHPrt}\ \ i^h\ \ o^h_1 & \mathtt{>>=} & \lambda \hat{i}^h. \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ i_0) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Defn. of skip*.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^h_1, \kappa^h). \\
\quad \mathit{chkHPrt}\ \ i^h\ \ o^h_1 & \mathtt{>>=} & \lambda \hat{i}^h. \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\mathit{skip}\ o_0\ \hat{i}^h) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Defn. skip,next*; $\mathtt{return}\, v \mathtt{>>=} \lambda x.e\ =\ \mathtt{return}\, v \mathtt{>>=} \lambda x.e[x/v]$.$\}$

$$\begin{array}{lll}
= \mathit{lift}^l_K\ (\mathit{next}\ \mathit{lo}) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^l_1, \kappa^l). \\
\quad \mathit{lift}^h_K\ (\mathit{next}\ (\mathit{skip}\ o_0\ i_0)) & \mathtt{>>=} & \lambda \mathit{Right}\ (o^h_1, \kappa^h). \\
\quad \mathit{chkHPrt}\ \ i^h\ \ o^h_1 & \mathtt{>>=} & \lambda \hat{i}^h. \\
\quad \mathit{chkLPrt}\ \ o^l_1 & \mathtt{>>} & \\
\quad \mathit{pull}\ \ (\mathit{os} +\!\!+ [o^l_1])\ \ [i_2, \ldots, i_n]\ \ (\kappa^l\ \ i^l)\ \ (\kappa^h\ \hat{i}^h) & \mathtt{>>=} & \lambda v.\ \mathit{mask}_H \mathtt{>>} \mathtt{return}\ v
\end{array}$$

$\{$*Lemma* 4.$\}$

$$= \mathtt{pull\ os}\ \ ((\mathtt{i}^{\mathtt{l}}_1, \mathtt{i}^{\mathtt{h}}_1) : \mathtt{is})\ \ (\mathtt{harness\ lo}\ \ (\mathtt{skip}\ \mathtt{o}_0\ \mathtt{i}_0))\ \mathtt{>>=}\ \lambda \mathtt{v}.\ \mathit{mask}_{\mathtt{H}} \mathtt{>>}\ \mathtt{return\ v}$$

□

## B. Lemma 4 Proof

*Proof.* Lemma 4.

$$pull\ os\ [i_1,\dots,i_n]\ (harness\ lo\ hi)$$

{*Lemma 1.*}

$$\begin{array}{lllll} = & next\ (harness\ lo\ hi) & \texttt{>>=} & \lambda Right(o_1,\kappa_1). \\ & next\ (\kappa_1\ i_1) & \texttt{>>=} & \lambda Right(o_2,\kappa_2). \\ & & \vdots & \\ & next\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda Right(o_n,\kappa_n). \\ & \texttt{return}(os +\!\!+ [o_1,\dots,o_n]) & & \end{array}$$

{*Lemma 2.*}

$$\begin{array}{ll} = next \left( \begin{array}{l} (lift\ .\ lift^l_K)\ (next\ lo) \quad \texttt{>>=} \quad \lambda Right\ (o^l,\kappa^l). \\ (lift\ .\ lift^h_K)\ (next\ hi) \quad \texttt{>>=} \quad \lambda Right\ (o^h,\kappa^h) \\ \textbf{let} \\ \quad \texttt{f} = \lambda(\texttt{i}^\texttt{l},\texttt{i}^\texttt{h}).\ \begin{array}{lll} checkHiPort\ \ i^h\ \ o^h & \texttt{>>=} & \lambda \hat{i}^h. \\ checkLoPort\ \ o^l & \texttt{>>} & \\ harness\ (\kappa^l\ \ i^l)\ (\kappa^h\ \hat{i}^h) & & \end{array} \\ \textbf{in} \\ \texttt{return (Right ((o}^\texttt{l}\texttt{, o}^\texttt{h}\texttt{), f))} \end{array} \right) & \texttt{>>=} \quad \lambda Right(o_1,\kappa_1). \\ \quad next\ (\kappa_1\ i_1) \qquad \texttt{>>=} \quad \lambda Right(o_2,\kappa_2). & \\ \qquad\qquad \vdots & \\ \quad next\ (\kappa_{n-1}\ i_{n-1}) \qquad \texttt{>>=} \quad \lambda Right(o_n,\kappa_n). & \\ \quad \texttt{return}(os +\!\!+ [o_1,\dots,o_n]) & \end{array}$$

{*Associativity of* >>=, *Lemma 3, Simplification.*}

$$\begin{array}{ll} = & lift^l_K\ (next\ lo) \quad \texttt{>>=} \quad \lambda Right\ (o^l,\kappa^l). \\ & lift^h_K\ (next\ hi) \quad \texttt{>>=} \quad \lambda Right\ (o^h,\kappa^h) \\ & \textbf{let} \\ & \quad \texttt{f} = \lambda(\texttt{i}^\texttt{l},\texttt{i}^\texttt{h}).\ \begin{array}{lll} checkHiPort\ \ i^h\ \ o^h & \texttt{>>=} & \lambda \hat{i}^h. \\ checkLoPort\ \ o^l & \texttt{>>} & \\ harness\ (\kappa^l\ \ i^l)\ (\kappa^h\ \hat{i}^h) & & \end{array} \\ & \textbf{in} \\ & \begin{array}{lll} next\ (\texttt{return}\ (Right\ ((o^l,\ o^h),\ f))) & \texttt{>>=} & \lambda Right(o_1,\kappa_1). \\ next\ (\kappa_1\ i_1) & \texttt{>>=} & \lambda Right(o_2,\kappa_2). \\ & \vdots & \\ next\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda Right(o_n,\kappa_n). \\ \texttt{return}(os +\!\!+ [o_1,\dots,o_n]) & & \end{array} \end{array}$$

{*Lemma 3*, $\texttt{return} = lift \circ \texttt{return}_K$.}

$$\begin{array}{ll} = & lift^l_K\ (next\ lo) \quad \texttt{>>=} \quad \lambda Right\ (o^l,\kappa^l). \\ & lift^h_K\ (next\ hi) \quad \texttt{>>=} \quad \lambda Right\ (o^h,\kappa^h) \\ & \textbf{let} \\ & \quad \texttt{f} = \lambda(\texttt{i}^\texttt{l},\texttt{i}^\texttt{h}).\ \begin{array}{lll} checkHiPort\ \ i^h\ \ o^h & \texttt{>>=} & \lambda \hat{i}^h. \\ checkLoPort\ \ o^l & \texttt{>>} & \\ harness\ (\kappa^l\ \ i^l)\ (\kappa^h\ \hat{i}^h) & & \end{array} \\ & \textbf{in} \\ & \begin{array}{lll} \texttt{return}_K\ (Right\ ((o^l,\ o^h),\ f)) & \texttt{>>=} & \lambda Right(o_1,\kappa_1). \\ next\ (\kappa_1\ i_1) & \texttt{>>=} & \lambda Right(o_2,\kappa_2). \\ & \vdots & \\ next\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda Right(o_n,\kappa_n). \\ \texttt{return}(os +\!\!+ [o_1,\dots,o_n]) & & \end{array} \end{array}$$

{*Left unit.*}

$$
\begin{array}{lll}
= \; \mathit{lift}^{l}_{K}\ (\mathit{next}\ lo) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{l}, \kappa^{l}). \\
\phantom{=} \; \mathit{lift}^{h}_{K}\ (\mathit{next}\ hi) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{h}, \kappa^{h}). \\
\phantom{=} \; \textbf{let} & & \\
\quad \texttt{f} = \lambda(\texttt{i}^{\texttt{l}}, \texttt{i}^{\texttt{h}}).\ \mathit{checkHiPort}\ i^{h}\ o^{h} & \texttt{>>=} & \lambda \hat{i}^{h}. \\
\quad \mathit{checkLoPort}\ o^{l} & \texttt{>>} & \\
\quad \mathit{harness}\ (\kappa^{l}\ i^{l})\ (\kappa^{h}\ \hat{i}^{h}) & & \\
\phantom{=} \; \textbf{in} & & \\
\quad \mathit{next}\ (f\ i_1) & \texttt{>>=} & \lambda \mathit{Right}(o_2, \kappa_2). \\
 & \vdots & \\
\quad \mathit{next}\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda \mathit{Right}(o_n, \kappa_n). \\
\quad \texttt{return}(os +\!\!+ [(o^{l},\ o^{h})_1, \ldots, o_n]) & &
\end{array}
$$

{*Consequence of Lemma 3.*}

$$
\begin{array}{lll}
= \; \mathit{lift}^{l}_{K}\ (\mathit{next}\ lo) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{l}, \kappa^{l}). \\
\phantom{=} \; \mathit{lift}^{h}_{K}\ (\mathit{next}\ hi) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{h}, \kappa^{h}). \\
\phantom{=} \; \mathit{chkHPrt}\ i^{h}\ o^{h} & \texttt{>>=} & \lambda \hat{i}^{h}. \\
\phantom{=} \; \mathit{chkLPrt}\ o^{l} & \texttt{>>} & \\
\phantom{=} \; \textbf{let} & & \\
\quad \texttt{f} = \lambda(\texttt{i}^{\texttt{l}}, \texttt{i}^{\texttt{h}}).\ \mathit{harness}\ (\kappa^{l}\ i^{l})\ (\kappa^{h}\ \hat{i}^{h}) & & \\
\phantom{=} \; \textbf{in} & & \\
\quad \mathit{next}\ (f\ i_1) & \texttt{>>=} & \lambda \mathit{Right}(o_2, \kappa_2). \\
 & \vdots & \\
\quad \mathit{next}\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda \mathit{Right}(o_n, \kappa_n). \\
\quad \texttt{return}(os +\!\!+ [(o^{l},\ o^{h})_1, \ldots, o_n]) & &
\end{array}
$$

{*Substitution of* **let** *binding.*}

$$
\begin{array}{lll}
= \; \mathit{lift}^{l}_{K}\ (\mathit{next}\ lo) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{l}, \kappa^{l}). \\
\phantom{=} \; \mathit{lift}^{h}_{K}\ (\mathit{next}\ hi) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{h}, \kappa^{h}). \\
\phantom{=} \; \mathit{chkHPrt}\ i^{h}\ o^{h} & \texttt{>>=} & \lambda \hat{i}^{h}. \\
\phantom{=} \; \mathit{chkLPrt}\ o^{l} & \texttt{>>} & \\
\phantom{=} \; \mathit{next}\ (\mathit{harness}\ (\kappa^{l}\ i^{l})\ (\kappa^{h}\ \hat{i}^{h})) & \texttt{>>=} & \lambda \mathit{Right}(o_2, \kappa_2). \\
 & \vdots & \\
\phantom{=} \; \mathit{next}\ (\kappa_{n-1}\ i_{n-1}) & \texttt{>>=} & \lambda \mathit{Right}(o_n, \kappa_n). \\
\phantom{=} \; \texttt{return}(os +\!\!+ [(o^{l},\ o^{h})_1, \ldots, o_n]) & &
\end{array}
$$

{*Lemma 1.*}

$$
\begin{array}{lll}
= \; \mathit{lift}^{l}_{K}\ (\mathit{next}\ lo) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{l}, \kappa^{l}). \\
\phantom{=} \; \mathit{lift}^{h}_{K}\ (\mathit{next}\ hi) & \texttt{>>=} & \lambda \mathit{Right}\ (o^{h}, \kappa^{h}). \\
\phantom{=} \; \mathit{chkHPrt}\ i^{h}\ o^{h} & \texttt{>>=} & \lambda \hat{i}^{h}. \\
\phantom{=} \; \mathit{chkLPrt}\ o^{l} & \texttt{>>} & \\
\phantom{=} \; \mathit{pull}\ \ (os +\!\!+ [(o^{l},\ o^{h})_1, \ldots, o_n])\ \ [i_2, \ldots, i_n]\ \ (\kappa^{l}\ \ i^{l})\ \ (\kappa^{h}\ \ \hat{i}^{h})) & &
\end{array}
$$

□