This document is the online-only appendix to:

Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem

J. NATHAN FOSTER
University of Pennsylvania
MICHAEL B. GREENWALD
Bell Labs, Lucent Technologies
JONATHAN T. MOORE
University of Pennsylvania
BENJAMIN C. PIERCE
University of Pennsylvania
ALAN SCHMITT
INRIA Rhône-Alpes

1

A. WELL-BEHAVEDNESS, TOTALITY, AND CONTINUITY PROOFS

This appendix contains the proofs for each of the results in our development of the foundations of lenses as well as representative well-behavedness, totality, and continuity proofs for several primitive and derived lenses.

3.5 Lemma: If $l \in C \rightleftharpoons A$, then $l\searrow$ is semi-injective on $\{(a,c) \mid (a,c) \in A \times C \ \wedge \ l\nearrow(l\searrow(a,c))\downarrow\}$.

Proof. Let $P = \{(a,c) \mid (a,c) \in A \times C \ \wedge \ l\nearrow(l\searrow(a,c))\downarrow\}$, and choose $(a,c) \in P$ and $(a',c') \in P$ with $a' \neq a$. Suppose, for a contradiction, that $l\searrow(a,c) = l\searrow(a',c')$. Then, by the definition of P and rule PutGet, we have $a = l\nearrow l\searrow(a,c) = l\nearrow l\searrow(a',c') = a'$; hence $a = a'$, a contradiction. □

3.8 Lemma: If l is oblivious and $l \in C_1 \rightleftharpoons A_1$ and $l \in C_2 \rightleftharpoons A_2$, then $l \in (C_1 \cup C_2) \rightleftharpoons (A_1 \cup A_2)$.

Proof. Straightforward. □

3.9 Lemma: If $l \in C \Longleftrightarrow A$ is oblivious, then $l\nearrow$ is a bijection from C to A.

Proof. If $C = \emptyset$, then, because l is total, A is also empty and $l\nearrow$ is trivially bijective. If C is non-empty, then we can choose an arbitrary $c \in C$ and define the inverse of $l\nearrow$ as $f = \lambda a.\, l\searrow(a,c)$. The fact that $(l\nearrow)\circ f = id$ follows directly from PutGet. The fact that $f \circ (l\nearrow) = id$ follows because $f(l\nearrow c') = l\searrow(l\nearrow c', c) = l\searrow(l\nearrow c', c')$ (by obliviousness) $= c'$ (by GetPut). □

3.11 Lemma: $\prec$ is a partial order on lenses.

PROOF. We show that $\prec$ is reflexive, transitive, and antisymmetric.

Reflexivity:. Immediate.

Transitivity:. Let l, l', and l'' be such that $l \prec l'$ and $l' \prec l''$. We have $\mathsf{dom}(l\nearrow) \subseteq \mathsf{dom}(l'\nearrow) \subseteq \mathsf{dom}(l''\nearrow)$ and $\mathsf{dom}(l\searrow) \subseteq \mathsf{dom}(l'\searrow) \subseteq \mathsf{dom}(l''\searrow)$. Moreover, for all $c \in \mathsf{dom}(l\nearrow)$, we have $l\nearrow c = l'\nearrow c = l''\nearrow c$. Finally, for all $(a, c) \in \mathsf{dom}(l\searrow)$, $l\searrow(a,\, c) = l'\searrow(a,\, c) = l''\searrow(a,\, c)$. Hence $l \prec l''$.

Antisymmetry:. Suppose $l \prec l'$ and $l' \prec l$. Then $\mathsf{dom}(l\nearrow) = \mathsf{dom}(l'\nearrow)$, $\mathsf{dom}(l\searrow) = \mathsf{dom}(l'\searrow)$, for every $c \in \mathsf{dom}(l\nearrow) = \mathsf{dom}(l'\nearrow)$ we have $l\nearrow c = l'\nearrow c$, and for every $(a, c) \in \mathsf{dom}(l\searrow) = \mathsf{dom}(l'\searrow)$ we have $l\searrow(a,\, c) = l'\searrow(a,\, c)$. Hence $l\nearrow = l'\nearrow$, $l\searrow = l'\searrow$, thus $l = l'$. □

3.12 Lemma: Let $l_0 \prec l_1 \prec \ldots \prec l_n \prec \ldots$ be an increasing chain of lenses. The lens l defined by

$$\begin{array}{rl} l\searrow(a,\, c) = l_i\searrow(a,\, c) & \text{if } l_i\searrow(a,\, c)\downarrow \text{ for some } i \\ l\nearrow c = l_i\nearrow c & \text{if } l_i\nearrow c\downarrow \text{ for some } i \end{array}$$

and undefined elsewhere is a least upper bound for the chain.

PROOF. Straightforward. □

3.14 Lemma: Let $l_0 \prec l_1 \prec \ldots \prec l_n \prec \ldots$ be an increasing chain of lenses, and let $C_0 \subseteq C_1 \subseteq \ldots$ and $A_0 \subseteq A_1 \subseteq \ldots$ be increasing chains of subsets of $\mathcal{V}$. Then:

(1) Well-behavedness commutes with limits:
$(\forall i \in \omega.\ l_i \in C_i \rightleftharpoons A_i) \quad \text{implies} \quad \bigsqcup_n l_n \in (\bigcup_i C_i) \rightleftharpoons (\bigcup_i A_i)$.

(2) Totality commutes with limits:
$(\forall i \in \omega.\ l_i \in C_i \Longleftrightarrow A_i) \quad \text{implies} \quad \bigsqcup_n l_n \in (\bigcup_i C_i) \Longleftrightarrow (\bigcup_i A_i)$.

PROOF. Let $l = \bigsqcup_n l_n$, let $C = \bigcup_i C_i$, and let $A = \bigcup_i A_i$.

We rely on the following property (which we call $\star_g$): if $l\nearrow c$ is defined for some $c \in C$, then there is some i such that $c \in C_i$ and $l\nearrow c = l_i\nearrow c$. To see this, let $c \in C$; then there is some j such that $\forall k \geq j.\ c \in C_k$. Moreover, by Corollary 3.13, there exist some j' such that $l\nearrow c = l_{j'}\nearrow c$. Let i be the max of j and j'; then we have (by definition of $\prec$) $l_i\nearrow c = l_{j'}\nearrow c = l\nearrow c$ and $c \in C_i$.

Similarly, we have the property $\star_p$: if $l\searrow(a,\, c)$ is defined for some $a \in A$ and $c \in C$, then there is some i such that $a \in A_i$, $c \in C_i$, and $l\searrow(a,\, c) = l_i\searrow(a,\, c)$. To see this, let $a \in A$ and $c \in C$; then there are some j and j' such that $\forall k \geq j.\ a \in A_k$ and $\forall k \geq j'.\ c \in C_k$. Moreover, by Corollary 3.13, there exists some j'' such that $l\searrow(a,\, c) = l_{j''}\searrow(a,\, c)$. Let i be the max of j, j', and j''; then we have (by definition of $\prec$) $l_i\searrow(a,\, c) = l_{j''}\searrow(a,\, c) = l\searrow(a,\, c)$, with $a \in A_i$ and $c \in C_i$.

We can now show that l satisfies the typing conditions (GET and PUT) of well-behaved lenses. Choose $c \in C$. If $l\nearrow c$ is defined, then by $\star_g$ there is some i such that $c \in C_i$ and $l\nearrow c = l_i\nearrow c$. As l_i is in $A_i \rightleftharpoons C_i$, we have $l_i\nearrow c \in A_i \subseteq A$. Conversely, let $(a, c) \in A \times C$; then if $l\searrow(a,\, c)$ is defined, then by $\star_p$ there is some i such that $(a, c) \in A_i \times C_i$ and $l\searrow(a,\, c) = l_i\searrow(a,\, c)$. As $l_i \in A_i \rightleftharpoons C_i$, we have $l_i\searrow(a,\, c) \in C_i \subseteq C$.

We next show that l satisfies GETPUT and PUTGET. Using $\star_g$ and $\star_p$, we calculate as follows:

GetPut:. Suppose $c \in C$. If $l \searrow (l \nearrow c, c) = \bot$, then we are done. Otherwise there is some i such that $c \in C_i$ and $l_i \nearrow c = l \nearrow c = a \in A_i \subseteq A$. Hence there is some j such that $a \in A_j$ and $l_j \searrow (a, c) = c'$. Let k be the max of i and j, so we have $a \in A_k$ and $c \in C_k$. By definition of $\prec$, we have $l_k \nearrow c = a$ and $l_k \searrow (a, c) = c'$. As GetPut holds for l_k, we have $c' = c$, hence GetPut holds for l.

PutGet:. Suppose $a \in A$ and $c \in C$. If $l \nearrow l \searrow (a, c) = \bot$, then we are done. Otherwise there is some i such that $a \in A_i$, $c \in C_i$, and $l_i \searrow (a, c) = l \searrow (a, c) = c' \in C_i \subseteq C$. Hence there is some j such that $c' \in C_j$ and $l_j \nearrow c' = a'$. Let k be the max of i and j, so we have $a \in A_k$ and $c \in C_k$. By definition of $\prec$, we have $l_k \searrow (a, c) = c'$ and $l_k \nearrow c' = a'$. As PutGet holds for l_k, we have $a' = a$, hence PutGet holds for l.

Finally, we show that l is total if all the l_i are. If $c \in C$, then there is some i such that $c \in C_i$, hence $l_i \nearrow c$ is defined, hence $l \nearrow c$ is defined. If $a \in A$ and $c \in C$, then there is some i such that $a \in A_i$ and $c \in C_i$, hence $l_i \searrow (a, c)$ is defined, thus $l \searrow (a, c)$ is defined. □

3.15 Theorem: Let $\mathcal{L}$ be the set of well-behaved lenses from C to A. Then $(\mathcal{L}, \prec)$ is a cpo with bottom.

Proof. First, recall that $\bot_l$ is the smallest well-behaved lens. Second, if $l_0 \prec l_1 \prec \ldots \prec l_n \prec \ldots$ is an increasing chain of well-behaved lenses, then by Lemma 3.14, it has a least upper bound that is well behaved. □

3.17 Corollary: Suppose f is a continuous function from lenses to lenses.

(1) If $l \in C \rightleftharpoons A$ implies $f(l) \in C \rightleftharpoons A$ for all l, then $\mathit{fix}(f) \in C \rightleftharpoons A$.
(2) Suppose $\emptyset = C_0 \subseteq C_1 \subseteq \ldots$ and $\emptyset = A_0 \subseteq A_1 \subseteq \ldots$ are increasing chains of subsets of $\mathcal{V}$. If $l \in C_i \Longleftrightarrow A_i$ implies $f(l) \in C_{i+1} \Longleftrightarrow A_{i+1}$ for all i and l, then $\mathit{fix}(f) \in (\bigcup_i C_i) \Longleftrightarrow (\bigcup_i A_i)$.

Proof. (1) First recall that $f^0(\bot_l) = \bot_l \in C \rightleftharpoons A$ for any C and A. From this, a simple induction on i (using the given implication at each step and the fact that f is monotonic) yields $f^i(\bot_l) \in C \rightleftharpoons A$ and $f^i(\bot_l) \prec f^{i+1}(\bot_l)$. By Lemma 3.14(1), $(\bigsqcup_i f^i(\bot_l)) \in C \rightleftharpoons A$. By Theorem 3.16, $\mathit{fix}(f) \in C \rightleftharpoons A$.

(2) First note that, since $C_0 = A_0 = \emptyset$, we have $f^0(\bot_l) = \bot_l \in C_0 \Longleftrightarrow A_0$. From this, a simple induction on i (using the given implication at each step) yields $f^i(\bot_l) \in C_i \Longleftrightarrow A_i$ and $f^i(\bot_l) \prec f^{i+1}(\bot_l)$. By Lemma 3.14(2), $(\bigsqcup_i f^i(\bot_l)) \in (\bigcup_i C_i) \Longleftrightarrow (\bigcup_i A_i)$. By Theorem 3.16, $\mathit{fix}(f) \in (\bigcup_i C_i) \Longleftrightarrow (\bigcup_i A_i)$. □

3.19 Lemma: Suppose f is a continuous function from lenses to lenses and $\mathbb{T}_0, \mathbb{T}_1, \ldots$ is a sequence of sets of total types with $\mathbb{T}_0 = \{(\emptyset, \emptyset)\}$. If for all l and i we have $(\forall \tau \in \mathbb{T}_i.\ \ l \in \tau)$ implies $(\forall \tau \in \mathbb{T}_{i+1}.\ \ f(l) \in \tau)$, then for every increasing instance $\tau_0 \subseteq \tau_1 \subseteq \ldots$ of $\mathbb{T}_0, \mathbb{T}_1, \ldots$ we have $\mathit{fix}(f) \in \bigcup_i \tau_i$.

Proof. Let $\tau_0 \subseteq \tau_1 \subseteq \ldots$ be an increasing instance of $\mathbb{T}_0, \mathbb{T}_1, \ldots$. Since $\mathbb{T}_0 = \{(\emptyset, \emptyset)\}$, we have $f^0(\bot_l) = \bot_l \in \tau$ for all $\tau \in \mathbb{T}_0$. From this, a simple induction on i (using the given implication at each step) yields $f^i(\bot_l) \in \tau$ for all $\tau \in \mathbb{T}_i$. Thus, we have $f^i(\bot_l) \in \tau_i$ for all τ_i. Hence by Lemma 3.14 we have $\bigsqcup_n f_n^n \in \bigcup_i \tau_i$. Using Theorem 3.16, we conclude that $\mathit{fix}(f) \in \bigcup_i \tau_i$. □

3.20 Lemma: For any lens l and sets of views C and A: $l \in C \overset{\Omega}{\rightleftharpoons} A$ implies $l \in C \rightleftharpoons A$ and (2) $l \in C \overset{\Omega}{\Longleftrightarrow} A$ implies $l \in C \Longleftrightarrow A$.

PROOF. Let $l \in C \overset{\Omega}{\rightleftharpoons} A$.

(1) We must prove that for all $c \in C$, $l\nearrow c \in A$. As $l\nearrow c \in A_\Omega$, and since $c \neq \Omega$, by convention we have $l\nearrow c \neq \Omega$. Similarly, let a, c in $A \times C$, then $l \searrow (a,\, c) \in C$.
(2) By convention, $C_\Omega \subseteq \mathsf{dom}(l\nearrow)$ implies $C \subseteq \mathsf{dom}(l\nearrow)$, and $A \times C_\Omega \subseteq \mathsf{dom}(l\searrow)$ implies $A \times C \subseteq \mathsf{dom}(l\searrow)$, as required. □

4.1 Lemma [Well-behavedness]: $\forall C{\subseteq}\mathcal{V}.\ \mathtt{id} \in C \overset{\Omega}{\rightleftharpoons} C$

PROOF.
GET: $(\mathtt{id})\nearrow c = c \in C$.
PUT: $(\mathtt{id}) \searrow (a,\, c) = a \in C$.
GETPUT: $(\mathtt{id}) \searrow ((\mathtt{id})\nearrow c,\, c) = (\mathtt{id}) \searrow (c,\, c) = c$.
PUTGET: $(\mathtt{id})\nearrow (\mathtt{id}) \searrow (a,\, c) = (\mathtt{id})\nearrow a = a$. □

4.2 Lemma [Totality]: $\forall C{\subseteq}\mathcal{V}.\ \mathtt{id} \in C \overset{\Omega}{\Longleftrightarrow} C$

PROOF. Immediate: both the *get* and *putback* directions of (id) are total functions. □

4.3 Lemma [Well-behavedness]:
$\forall A, B, C{\subseteq}\mathcal{V}.\ \forall l \in C \overset{\Omega}{\rightleftharpoons} B.\ \forall k \in B \overset{\Omega}{\rightleftharpoons} A. \quad l;k \in C \overset{\Omega}{\rightleftharpoons} A$

PROOF.
GET: If $k\nearrow l\nearrow c = (l;k)\nearrow c$ is defined, then $l\nearrow c \in B$ by GET for l, so $(l;k)\nearrow c \in A$ by GET for k.

PUT: If $l \searrow (k \searrow (a,\, l\nearrow c),\, c) = (l;k) \searrow (a,\, c)$ is defined, then $l\nearrow c \in B_\Omega$ by GET for l and our convention on treatment of Ω by *get* functions, so $k \searrow (a,\, l\nearrow c) \in B$ by PUT for k, so $l \searrow (k \searrow (a,\, l\nearrow c),\, c) \in C$ by PUT for l.

GETPUT: Assume that $(l;k)\nearrow c$ is defined. Then:

$$
\begin{array}{lll}
 & (l;k) \searrow \Big(\underline{(l;k)\nearrow c},\, c\Big) & \\
= & \underline{(l;k) \searrow (k\nearrow (l\nearrow c),\, c)} & \text{by definition (of the underlined expression)} \\
= & l \searrow \Big(\underline{k \searrow (k\nearrow (l\nearrow c),\, l\nearrow c)},\, c\Big) & \text{by definition} \\
\sqsubseteq & \underline{l \searrow (l\nearrow c,\, c)} & \text{by GETPUT for } k \\
\sqsubseteq & c & \text{by GETPUT for } l
\end{array}
$$

PUTGET: Assume that $(l;k) \searrow (a,\, c)$ is defined. Then:

$$
\begin{array}{lll}
 & (l;k)\nearrow \Big(\underline{(l;k) \searrow (a,\, c)}\Big) & \\
= & \underline{(l;k)\nearrow (l \searrow (k \searrow (a,\, l\nearrow c),\, c))} & \text{by definition} \\
= & k\nearrow \Big(\underline{l\nearrow (l \searrow (k \searrow (a,\, l\nearrow c),\, c))}\Big) & \text{by definition} \\
\sqsubseteq & \underline{k\nearrow (k \searrow (a,\, l\nearrow c))} & \text{by PUTGET for } l \\
\sqsubseteq & a & \text{by PUTGET for } k \qquad \square
\end{array}
$$

4.4 Lemma [Totality]:
$\forall A, B, C{\subseteq}\mathcal{V}.\ \forall l \in C \overset{\Omega}{\Longleftrightarrow} B.\ \forall k \in B \overset{\Omega}{\Longleftrightarrow} A. \quad l;k \in C \overset{\Omega}{\Longleftrightarrow} A$

PROOF. Let $c \in C$; then $l \nearrow c$ is defined (by totality of l) and is in B, hence $k \nearrow l \nearrow c = (l;k) \nearrow c$ is defined (by totality of k). Conversely, let $a \in A$ and $c \in C_\Omega$; then $l \nearrow c$ is defined and is in B_Ω. Thus, $k \searrow (a,\, l \nearrow c)$ is defined and is in B, and so $l \searrow (k \searrow (a,\, l \nearrow c),\, c) = (l;k) \searrow (a,\, c)$ is defined. □

4.5 Lemma [Continuity]: Let F and G be continuous functions from lenses to lenses. Then the function $\lambda l.\ (F(l); G(l))$ is continuous.

PROOF. We first argue that $\lambda l.\ (F(l); G(l))$ is monotone. Let l and l' be two lenses with $l \prec l'$. We must show that $F(l); G(l) \prec F(l'); G(l')$. For the *get* direction, let $c \in \mathcal{V}$, and assume that $(F(l); G(l)) \nearrow c$ is defined. We have:

$$\begin{array}{ll} & (F(l); G(l)) \nearrow c \\ = & G(l) \nearrow (F(l) \nearrow c) \\ = & G(l) \nearrow (F(l') \nearrow c) \quad \text{by } F(l) \prec F(l'), \text{ since } F(l) \nearrow c \text{ is defined} \\ = & G(l') \nearrow (F(l') \nearrow c) \quad \text{by } G(l) \prec G(l') \\ = & (F(l'); G(l')) \nearrow c. \end{array}$$

For the *putback* direction, let $(a, c) \in \mathcal{V} \times \mathcal{V}_\Omega$, assume that $(F(l); G(l)) \searrow (a,\, c)$ is defined, and calculate as follows:

$$\begin{array}{ll} & (F(l); G(l)) \searrow (a,\, c) \\ = & F(l) \searrow (G(l) \searrow (a,\, F(l) \nearrow c),\, c) \\ = & F(l) \searrow (G(l) \searrow (a,\, F(l') \nearrow c),\, c) \quad \text{by } F(l) \prec F(l') \\ = & F(l) \searrow (G(l') \searrow (a,\, F(l') \nearrow c),\, c) \quad \text{by } G(l) \prec G(l') \\ = & F(l') \searrow (G(l') \searrow (a,\, F(l') \nearrow c),\, c) \quad \text{by } F(l) \prec F(l') \\ = & (F(l'); G(l')) \searrow (a,\, c). \end{array}$$

Thus $\lambda l.\ (F(l); G(l))$ is monotone. We must now prove that it is continuous.

Let $l_0 \prec l_1 \prec \ldots \prec l_n \prec \ldots$ be an increasing chain of well-behaved lenses. Let $l = \bigsqcup_i l_i$. We have, for $c \in \mathcal{V}$,

$$\begin{array}{lll} & (F(l); G(l)) \nearrow c = v & \\ \iff & G(l) \nearrow F(l) \nearrow c = v & \text{by definition of ;} \\ \iff & G(l) \nearrow F(\bigsqcup_i l_i) \nearrow c = v & \text{by definition of } l \\ \iff & G(l) \nearrow (\bigsqcup_i F(l_i)) \nearrow c = v & \text{by continuity of } F \\ \iff & \exists i_1.\, G(l) \nearrow F(l_{i_1}) \nearrow c = v & \text{by Corollary 3.13 (GET)} \\ \iff & \exists i_1.\, G(\bigsqcup_i l_i) \nearrow F(l_{i_1}) \nearrow c = v & \text{by definition of } l \\ \iff & \exists i_1.\, (\bigsqcup_i G(l_i)) \nearrow F(l_{i_1}) \nearrow c = v & \text{by continuity of } G \\ \iff & \exists i_2, i_1.\, G(l_{i_2}) \nearrow F(l_{i_1}) \nearrow c = v & \text{by Corollary 3.13 (GET)} \\ \iff & \exists i.\, G(l_i) \nearrow F(l_i) \nearrow c = v & \text{by } \left\{ \begin{array}{l} i = \mathtt{max}(i_1, i_2) \\ \text{and } F \text{ and } G \text{ monotone} \end{array} \right. \\ \iff & \exists i.\, (F(l_i); G(l_i)) \nearrow c = v & \text{by definition of ;} \\ \iff & (\bigsqcup_i (F(l_i); G(l_i))) \nearrow c = v & \text{by Corollary 3.13 (GET)} \end{array}$$

and

$$
\begin{array}{lll}
 & (F(l);G(l)) \searrow (a,\, c) = v & \\
\iff & F(l) \searrow (G(l) \searrow (a,\, F(l) \nearrow c),\, c) = v & \text{by definition of ;} \\
\iff & F(l) \searrow (G(l) \searrow (a,\, F(\bigsqcup_i l_i) \nearrow c),\, c) = v & \text{by definition of } l \\
\iff & F(l) \searrow (G(l) \searrow (a,\, (\bigsqcup_i F(l_i)) \nearrow c),\, c) = v & \text{by continuity of } F \\
\iff & \exists i_1.F(l) \searrow (G(l) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by Corollary 3.13 (GET)} \\
\iff & \exists i_1.F(l) \searrow (G(\bigsqcup_i l_i) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by definition of } l \\
\iff & \exists i_1.F(l) \searrow ((\bigsqcup_i G(l_i)) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by continuity of } G \\
\iff & \exists i_2, i_1.F(l) \searrow (G(l_{i_2}) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by Corollary 3.13 (PUT)} \\
\iff & \exists i_2, i_1.F(\bigsqcup_i li) & \\
 & \quad \searrow (G(l_{i_2}) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by definition of } l \\
\iff & \exists i_2, i_1.(\bigsqcup_i F(l_i)) & \\
 & \quad \searrow (G(l_{i_2}) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by continuity of } F \\
\iff & \exists i_3, i_2, i_1.F(l_{i_3}) & \\
 & \quad \searrow (G(l_{i_2}) \searrow (a,\, F(l_{i_1}) \nearrow c),\, c) = v & \text{by Corollary 3.13 (PUT)} \\
\iff & \exists i.F(l_i) \searrow (G(l_i) \searrow (a,\, F(l_i) \nearrow c),\, c) = v & \text{by } \left\{ \begin{array}{l} i = \mathtt{max}(i_1, i_2, i_3) \\ \text{and } F \text{ and } G \text{ monotone} \end{array} \right. \\
\iff & \exists i.(F(l_i);G(l_i)) \searrow (a,\, c) = v & \text{by definition of ;} \\
\iff & (\bigsqcup_i (F(l_i);G(l_i))) \searrow (a,\, c) = v & \text{by Corollary 3.13 (PUT).}
\end{array}
$$

Hence the lenses $\bigsqcup_i (F(l_i);G(l_i))$ and $F(l);G(l)$ are equal. □

5.1 Lemma [Well-behavedness]:
$\forall C, A \subseteq \mathcal{T}$ *with* $C = C^{\circlearrowleft}$, $A = A^{\circlearrowleft}$, $\mathsf{doms}(C) = \mathsf{doms}(A)$.
$\forall m \in (\Pi n \in \mathcal{N}.\, C(n) \stackrel{\Omega}{\rightleftharpoons} A(n))$.
$\mathtt{wmap}\ m \in C \stackrel{\Omega}{\rightleftharpoons} A$

PROOF.
GET: Suppose $c \in C$ and $m(n) \nearrow c(n)$ is defined for each $n \in \mathsf{dom}(c)$. Then, by the (dependent) type of m, we have $m(n) \nearrow c(n) \in A(n)$ for each n. Since $\mathsf{dom}(A) = \mathsf{dom}(C)$, there exists a non-empty subset of A whose elements all have domain $D = \mathsf{dom}(c)$. Also, the tree $\{\!|n \mapsto m(n) \nearrow c(n) \mid n \in \mathsf{dom}(c)|\!\}$ is an element of the set $\{\!|n \mapsto A(n) \mid n \in D|\!\}$, which is itself a subset of A since A is shuffle closed. Hence, $(\mathtt{wmap}\ m) \nearrow c \in A$.

PUT: Let $a \in A$ and $c \in C$. For all $n \in \mathsf{dom}(a)$, we have $m(n) \searrow (a(n),\, c(n)) \in C(n)$ (with $c(n)$ possibly being Ω). Hence, by a similar argument as above, since $\mathsf{dom}(A) = \mathsf{dom}(C)$ and $C = C^{\circlearrowleft}$, we have $(\mathtt{wmap}\ m) \searrow (a,\, c) \in C$.

GETPUT: Assume that $(\mathtt{wmap}\ m) \nearrow c$ is defined. Then

$$
\begin{array}{ll}
 & (\mathtt{wmap}\ m) \searrow ((\mathtt{wmap}\ m) \nearrow c,\, c) \\
= & (\mathtt{wmap}\ m) \searrow (\{\!|n \mapsto m(n) \nearrow c(n) \mid n \in \mathsf{dom}(c)|\!\},\, c) \\
= & \{\!|n \mapsto m(n) \searrow (m(n) \nearrow c(n),\, c(n)) \mid n \in \mathsf{dom}(c)|\!\} \\
\sqsubseteq & \{\!|n \mapsto c(n) \mid n \in \mathsf{dom}(c)|\!\} \qquad \text{by GETPUT for each } m(n) \\
= & c.
\end{array}
$$

PUTGET: Assume that $(\texttt{wmap}\ m)\searrow(a,\,c)$ is defined. Then

$$\begin{array}{ll} & (\texttt{wmap}\ m)\nearrow((\texttt{wmap}\ m)\searrow(a,\,c)) \\ = & (\texttt{wmap}\ m)\nearrow \left\{\!\!\left\{n \mapsto m(n)\searrow(a(n),\,c(n)) \mid n \in \mathsf{dom}(a)\right\}\!\!\right\} \\ = & \left\{\!\!\left\{n \mapsto m(n)\nearrow(l\searrow(a(n),\,c(n))) \mid n \in \mathsf{dom}(a)\right\}\!\!\right\} \\ \sqsubseteq & \left\{\!\!\left\{n \mapsto a(n) \mid n \in \mathsf{dom}(a)\right\}\!\!\right\} \qquad \text{by PUTGET for each } m(n) \\ = & a. \end{array}$$ □

7.2 Lemma: Let $S, T \subseteq \mathcal{T}$. Then

(1) $(S :: T) = (S :: T)^{\circlearrowleft}$

(2) $\texttt{[}T\texttt{]} = \texttt{[}T\texttt{]}^{\circlearrowleft}$.

PROOF. We prove each part of the lemma directly.

(1) We calculate $(S :: T)^{\circlearrowleft}$. From the definition of cons cells, the set $\mathsf{doms}(S :: T)$ of possible domains of trees in $(S :: T)$ is $\{\{\texttt{*h},\ \texttt{*t}\}\}$. We then calculate $(S :: T)^{\circlearrowleft}$ as:

$$\begin{array}{rl} (S :: T)^{\circlearrowleft} & = \bigcup_{D \in \mathsf{doms}(S :: T)} \left\{\!\!\left\{n \mapsto (S :: T)(n) \mid n \in D\right\}\!\!\right\} \\ & = \left\{\!\!\left\{\texttt{*h} \mapsto S,\ \texttt{*t} \mapsto T\right\}\!\!\right\} \end{array}$$

which is equal to $S :: T$.

(2) We calculate $\texttt{[}T\texttt{]}^{\circlearrowleft}$. From the definition of lists, the set $\mathsf{doms}(\texttt{[}T\texttt{]})$ of domains of trees in $\texttt{[}T\texttt{]}$ is $\{\emptyset,\ \{\texttt{*h},\ \texttt{*t}\}\}$. We then calculate $\texttt{[}T\texttt{]}^{\circlearrowleft}$ as:

$$\begin{array}{rl} \texttt{[}T\texttt{]}^{\circlearrowleft} & = \bigcup_{D \in \mathsf{doms}(\texttt{[}T\texttt{]})} \left\{\!\!\left\{n \mapsto \texttt{[}T\texttt{]}(n) \mid n \in D\right\}\!\!\right\} \\ & = \{\!\!\{\}\!\!\} \cup \left\{\!\!\left\{\texttt{*h} \mapsto T,\ \texttt{*t} \mapsto \texttt{[}T\texttt{]}\right\}\!\!\right\} \end{array}$$

which is equal to $\texttt{[}T\texttt{]}$. □

7.3 Lemma [Well-behavedness]:
$\forall C, A \subseteq \mathcal{T}.\ \forall l \in C \overset{\Omega}{\rightleftharpoons} A. \quad \texttt{list\_map}\ l \in \texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$

PROOF. Note that $\texttt{list\_map}\ l$ is the fixed point of the function:

$$f = \lambda k.\ \texttt{wmap}\ \{\texttt{*h} \mapsto l,\ \texttt{*t} \mapsto k\}$$

We use Corollary 3.17(1), which states that if, assuming that $k \in \texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$, we can prove $f(k) \in \texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$, then $\mathit{fix}(f) \in \texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$.

We assume that $k \in \texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$ and show that $f(k)$ has type $\texttt{[}C\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}A\texttt{]}$ directly, using the type of `wmap`. We write m for the total function from names to lenses described by $\{\texttt{*h} \mapsto l,\ \texttt{*h} \mapsto k\}$; i.e., m maps `*h` to l, `*t` to k, and every

other name to `id`. We first show that $m \in \Pi n{\in}\mathcal{N}.\ C(n) \stackrel{\Omega}{=\!\!=} A(n)$:

$$\begin{array}{rr} & m(\texttt{*h}) = l \ \in\ [\![C]\!](\texttt{*h}) \stackrel{\Omega}{=\!\!=} [\![A]\!](\texttt{*h}) \\ \text{i.e.,} & C \stackrel{\Omega}{=\!\!=} A \end{array}$$

by the type of l;

$$\begin{array}{rr} & m(\texttt{*t}) = k \ \in\ [\![C]\!](\texttt{*t}) \stackrel{\Omega}{=\!\!=} [\![A]\!](\texttt{*t}) \\ \text{i.e.,} & [\![C]\!] \stackrel{\Omega}{=\!\!=} [\![A]\!] \end{array}$$

by assumption;

$$\begin{array}{rr} & m(n) = \texttt{id} \ \in\ [\![C]\!](n) \stackrel{\Omega}{=\!\!=} [\![A]\!](n) \qquad \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & \emptyset \stackrel{\Omega}{=\!\!=} \emptyset \end{array}$$

vacuously.

Hence, m has the correct type. The type of `wmap` also requires that both $[\![C]\!]$ and $[\![A]\!]$ be shuffle closed and that $\mathsf{doms}([\![C]\!]) = \mathsf{doms}([\![A]\!])$. The first condition follows from Lemma 7.2(2); the second condition is immediate as both $\mathsf{doms}([\![C]\!])$ and $\mathsf{doms}([\![A]\!])$ are the set $\{\{\texttt{*h}, \texttt{*t}\}, \emptyset\}$.

Using the type of `wmap`, we conclude that $f(k) \in [\![C]\!] \stackrel{\Omega}{=\!\!=} [\![A]\!]$ and by Corollary 3.17, that $\mathit{fix}(f) = \texttt{list\_map}\ l \in [\![C]\!] \stackrel{\Omega}{=\!\!=} [\![A]\!]$. □

7.4 Lemma [Totality]: $\forall C, A{\subseteq}\mathcal{T}.\ \forall l \in C \stackrel{\Omega}{\Longleftrightarrow} A.\ \ \texttt{list\_map}\ l \in [\![C]\!] \stackrel{\Omega}{\Longleftrightarrow} [\![A]\!]$

Proof. We pick these two chains of types:

$$\begin{array}{l} C_0 = A_0 = \emptyset \\ C_{i+1} = [\![C^i]\!] \\ A_{i+1} = [\![A^i]\!] \end{array}$$

Next, we show by induction on i that $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$ implies $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$ for all i.

We calculate the type of $f(l)$ directly from the type of `wmap`. As above, we write m for the function that maps `*h` to l, `*t` to k and every other n to `id`. We analyze two subcases.

For the base case, $i = 0$, we have

$$\begin{array}{rr} & m(n) \ \in\ C_1(n) \stackrel{\Omega}{\Longleftrightarrow} A_1(n) \\ \text{i.e.,} & [\,](n) \stackrel{\Omega}{\Longleftrightarrow} [\,](n) \\ \text{i.e.,} & \emptyset \stackrel{\Omega}{\Longleftrightarrow} \emptyset \end{array}$$

vacuously.

Also, we trivially have that $\emptyset$ is shuffle closed and $\mathsf{doms}(\emptyset) = \mathsf{doms}(\emptyset)$. Using the type of `wmap` we conclude that $f(k) \in C_1 \stackrel{\Omega}{\Longleftrightarrow} A_1$.

For the induction step, we assume that $i > 0$ and $k \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$. From these

facts we have

$$\begin{array}{rl} & m(\texttt{*h}) = l \ \in \ [C^i](\texttt{*h}) \overset{\Omega}{\Longleftrightarrow} [A^i](\texttt{*h}) \\ \text{i.e.,} & C \overset{\Omega}{\Longleftrightarrow} A \\ & \text{by } i > 0 \text{ and the type of } l; \end{array}$$

$$\begin{array}{rl} & m(\texttt{*t}) = k \ \in \ [C^i](\texttt{*t}) \overset{\Omega}{\Longleftrightarrow} [A^i](\texttt{*t}) \\ \text{i.e.,} & [C^{i-1}] \overset{\Omega}{\Longleftrightarrow} [A^{i-1}] \\ & \text{by } i > 0; \\ \text{i.e.,} & C_i \overset{\Omega}{\Longleftrightarrow} A_i \\ & \text{by induction hypothesis;} \end{array}$$

$$\begin{array}{rl} & m(n) = \texttt{id} \ \in \ [C^{i+1}](n) \overset{\Omega}{\Longleftrightarrow} [A^{i+1}](n) \quad \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & \emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset \\ & \text{vacuously.} \end{array}$$

As above, both $[C^i]$ and $[A^i]$ are shuffle closed and have equal domains. Using the type of `wmap`, we conclude that $f(k) \in C_{i+1} \overset{\Omega}{\Longleftrightarrow} A_{i+1}$ which finishes the case and the inductive proof.

By Corollary 3.17(2) we have that

$$\begin{array}{rl} & \texttt{list\_map}\ l \in \ \bigcup_i C_i \overset{\Omega}{\Longleftrightarrow} \bigcup_i A_i \\ \text{i.e.,} & (\emptyset \cup \bigcup_i [C^i]) \overset{\Omega}{\Longleftrightarrow} (\emptyset \cup \bigcup_i [A^i]) \\ \text{i.e.,} & [C] \overset{\Omega}{\Longleftrightarrow} [A], \end{array}$$

which finishes the proof. □

7.5 Lemma [Well-behavedness]: $\forall D \subseteq \mathcal{T}.\ \texttt{rotate} \in [D] \overset{\Omega}{\rightleftharpoons} [D]$

Proof. First, note that `rotate` is the fixed point of the function:

$$\begin{array}{rl} f = \lambda l. & \texttt{acond}\ ([] \cup (D :: [])) \ ([] \cup (D :: [])) \\ & \quad \texttt{id} \\ & \quad (\texttt{rename *h tmp}; \\ & \quad\ \texttt{hoist\_nonunique *t \{*h, *t\}}; \\ & \quad\ \texttt{fork \{*h\} id (rename tmp *h}; \ l; \ \texttt{plunge *t})) \end{array}$$

Let $C = A = [D]$. We assume that $l \in C \overset{\Omega}{\rightleftharpoons} A$ and prove that $f(l) \in C \overset{\Omega}{\rightleftharpoons} A$. Using Corollary 3.17(1), we conclude that $\mathit{fix}(f) = \texttt{rotate}\ m \in C \overset{\Omega}{\rightleftharpoons} A$.

We calculate the type of $f(l)$, working top down. The outermost lens is an `acond` instance. Using the type of `acond`, we must prove that the first branch has this type:

$$\begin{array}{rl} \texttt{id} \in & C \cap ([] \cup (D :: [])) \overset{\Omega}{\rightleftharpoons} A \cap ([] \cup (D :: [])) \\ \text{i.e.,} & [D] \cap ([] \cup (D :: [])) \overset{\Omega}{\rightleftharpoons} [D] \cap ([] \cup (D :: [])) \\ \text{i.e.,} & [] \cup (D :: []) \overset{\Omega}{\rightleftharpoons} [] \cup (D :: []) \end{array}$$

which is immediate from the type of `id`. Similarly, we must show that the second branch has this type:

$$\begin{array}{rl} & \texttt{rename *h tmp;} \\ & \texttt{hoist\_nonunique *t \{*h, *t\};} \\ & \texttt{fork \{*h\}id (rename tmp *h; } l\texttt{; plunge *t)} \\ \in & \quad C \setminus (\texttt{[]} \cup (D :: \texttt{[]})) \stackrel{\Omega}{=\!=} A \setminus (\texttt{[]} \cup (D :: \texttt{[]})) \\ \text{i.e.,} & \quad \texttt{[}D\texttt{]} \setminus (\texttt{[]} \cup (D :: \texttt{[]})) \stackrel{\Omega}{=\!=} \texttt{[}D\texttt{]} \setminus (\texttt{[]} \cup (D :: \texttt{[]})) \\ \text{i.e.,} & \quad D :: D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{=\!=} D :: D :: \texttt{[}D\texttt{]} \end{array}$$

From the type of `rename`, we have

$$\texttt{rename *h tmp} \in D :: D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{=\!=} \{\!|\, \texttt{tmp} \mapsto D,\ \texttt{*t} \mapsto D :: \texttt{[}D\texttt{]} \,|\!\}$$

Moreover, using the type of `hoist_nonunique`, we have

$$\begin{array}{l} \texttt{hoist\_nonunique *t \{*h, *t\}} \\ \quad \in \{\!|\, \texttt{tmp} \mapsto D,\ \texttt{*t} \mapsto D :: \texttt{[}D\texttt{]} \,|\!\} \stackrel{\Omega}{=\!=} \{\!|\, \texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto D,\ \texttt{*t} \mapsto \texttt{[}D\texttt{]} \,|\!\} \end{array}$$

Next we show that the `fork` lens has type

$$\begin{array}{l} \texttt{fork \{*h\} id (rename tmp *h; } l\texttt{; plunge *t)} \\ \quad \in \{\!|\, \texttt{tmp} \mapsto D,\ \texttt{*h} \mapsto D,\ \texttt{*t} \mapsto \texttt{[}D\texttt{]} \,|\!\} \stackrel{\Omega}{=\!=} D :: D :: \texttt{[}D\texttt{]} \end{array}$$

We prove that the first arm has type:

$$\texttt{id} \in \{\!|\, \texttt{*h} \mapsto D \,|\!\} \stackrel{\Omega}{=\!=} \{\!|\, \texttt{*h} \mapsto D \,|\!\}$$

and that the second arm has type:

$$\texttt{rename tmp *h; } l\texttt{; plunge *t} \in \{\!|\, \texttt{tmp} \mapsto D, \texttt{*t} \mapsto \texttt{[}D\texttt{]} \,|\!\} \stackrel{\Omega}{=\!=} \{\!|\, \texttt{*t} \mapsto \texttt{[}D\texttt{]} \,|\!\}$$

The first typing follows from the type of `id` and the second using the types of `rename`, `plunge`, and the composition operator, as well as the type of l we have by hypothesis. Hence, the entire `fork` has the type calculated above. By the type of the composition operator we conclude that the second branch has the correct type.

We conclude that the `acond` lens has type $C \stackrel{\Omega}{=\!=} A$, and so, by Corollary 3.17(1), that $\mathit{fix}(f) = \texttt{rotate}$ has the same type. □

7.6 Lemma [Totality]: $\forall D \subseteq \mathcal{T}.\ \texttt{rotate} \in \texttt{[}D\texttt{]} \stackrel{\Omega}{\Longleftrightarrow} \texttt{[}D\texttt{]}$

Proof. To prove that `rotate` is total, we use Corollary 3.17(2). Let

$$\begin{array}{l} C_0 = A_0 = \emptyset \\ C_{i+1} = A_{i+1} = \texttt{[}D^i\texttt{]} \end{array}$$

be two chains of types. Again, note that `rotate` is the fixed point of the function f described in the well-behavedness proof. We prove, by induction on i, that if $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$ then $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$.

For the base case, $i = 0$, we must show that the $f(l)$ has type $C_1 \Longleftrightarrow^{\Omega} A_1$. The outermost lens in $f(l)$ is an `acond`. We prove that each branch has the correct type:

$$\begin{array}{rr} & \texttt{id} \in\ C_1 \cap (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_1 \cap (\texttt{[]} \cup D :: \texttt{[]}) \\ \text{i.e.,} & \texttt{[]} \cap (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} \texttt{[]} \cap (\texttt{[]} \cup D :: \texttt{[]}) \\ \text{i.e.,} & \texttt{[]} \Longleftrightarrow^{\Omega} \texttt{[]} \end{array}$$

$$\begin{array}{rr} & \texttt{rename *h tmp}; \\ & \texttt{hoist\_nonunique *t \{*h *t\}}; \\ & \texttt{fork \{*h\} id (rename tmp *h; } l\texttt{; plunge *t)} \\ & \quad \in\ C_1 \cap (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_1 \cap (\texttt{[]} \cup D :: \texttt{[]}) \\ \text{i.e.,} & \quad \texttt{[]} \setminus (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} \texttt{[]} \setminus (\texttt{[]} \cup D :: \texttt{[]}) \\ \text{i.e.,} & \quad \emptyset \Longleftrightarrow^{\Omega} \emptyset \end{array}$$

The first fact is immediate by the type of `id`; the second holds vacuously. By the type of `acond`, we have $f(l) \in C_1 \Longleftrightarrow^{\Omega} A_1$, which finishes the case.

For the induction step, we assume that $i > 0$ and that $l \in A_i \Longleftrightarrow^{\Omega} C_i$. Again, we unwind the definition of $f(l)$, revealing an `acond` lens and prove that the each branch has the correct type. For the first branch, we calculate the type as follows:

$$\begin{array}{rrll} & \texttt{id} \in\ C_{i+1} \cap (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_{i+1} \cap (\texttt{[]} \cup D :: \texttt{[]}) & \\ \text{i.e.,} & \texttt{[}D^i\texttt{]} \cap (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} \texttt{[}D^i\texttt{]} \cap (\texttt{[]} \cup D :: \texttt{[]}) & \\ \text{i.e.,} & D :: \texttt{[]} \Longleftrightarrow^{\Omega} D :: \texttt{[]} & \text{if } i = 1 \\ & \emptyset \Longleftrightarrow^{\Omega} \emptyset & \text{otherwise} \end{array}$$

which follows from the type of `id` in either case.

For the second branch we must prove that:

$$\begin{array}{rr} & \texttt{rename *h tmp}; \\ & \texttt{hoist\_nonunique *t \{*h *t\}}; \\ & \texttt{fork \{*h\} id (rename tmp *h; } l\texttt{; plunge *t)} \\ & \in\ C_{i+1} \setminus (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_{i+1} \cap (\texttt{[]} \setminus D :: \texttt{[]}) \\ \text{i.e.,} & \quad \texttt{[}D^i\texttt{]} \setminus (\texttt{[]} \cup D :: \texttt{[]}) \Longleftrightarrow^{\Omega} \texttt{[}D^i\texttt{]} \cap (\texttt{[}D^i\texttt{]} \setminus D :: \texttt{[]}) \\ \text{i.e.,} & \quad D :: D :: \texttt{[}D^{i-2}\texttt{]} \Longleftrightarrow^{\Omega} D :: D :: \texttt{[}D^{i-2}\texttt{]} \end{array}$$

We analyze two subcases.

Case $i = 1$ Since $\texttt{[}D^{i-2}\texttt{]} = \emptyset$, the type $D :: D :: \texttt{[}D^{i-2}\texttt{]}$ is also empty. Hence, the second branch has the required lens type $\emptyset \Longleftrightarrow^{\Omega} \emptyset$ vacuously.

Case $i > 1$*:* From the type of `rename`, we have that

$$\texttt{rename *h tmp} \in D :: D :: \texttt{[}D^{i-2}\texttt{]} \Longleftrightarrow^{\Omega} \{\!|\, \texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: \texttt{[}D^{i-2}\texttt{]} \,|\!\}$$

Using the type of `hoist_nonunique`, we have

$$\begin{array}{l} \texttt{hoist\_nonunique *t \{*h, *t\}} \\ \quad \in \{\!|\, \texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: \texttt{[}D^{i-2}\texttt{]} \,|\!\} \Longleftrightarrow^{\Omega} \{\!|\, \texttt{*h} \mapsto D, \texttt{tmp} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]} \,|\!\} \end{array}$$

To show that the composite lens formed from these lenses has the desired type, we must show that

$$\begin{array}{l} \texttt{fork \{*h\} id (rename tmp *h; } l\texttt{; plunge *t)} \\ \quad \in \{\!|\, \texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]} \,|\!\} \Longleftrightarrow^{\Omega} D :: D :: \texttt{[}D^{i-2}\texttt{]} \end{array}$$

To show this fact using the type of `fork`, we must show that the first branch has type

$$\texttt{id} \in \left\{\!\left|\texttt{*h} \mapsto D\right|\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\left|\texttt{*h} \mapsto D\right|\!\right\}$$

and that the second branch has type

$$\texttt{rename tmp *h};\ l;\ \texttt{plunge *t} \in \left\{\!\left|\texttt{tmp} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}\right|\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}\right|\!\right\}$$

The first typing follows from the type of `id` and the second using the types of `rename`, `plunge`, and the composition operator, together with the type of l we have by induction hypothesis. Hence, the entire `fork` has the total type stated above. By the type of the composition operator, the entire second branch has the correct type calculated above.

Thus, from the type of `acond`, we have $f(l) \in C_{i+1} \overset{\Omega}{\rightleftharpoons} A_{i+1}$, which finishes the case and the inductive proof.

By Corollary 3.17(2), we conclude that

$$\begin{array}{rrr} \mathit{fix}(f) = \texttt{rotate} \in & \bigcup_i C_i & \overset{\Omega}{\Longleftrightarrow} \bigcup_i A_i \\ \text{i.e.,} & \emptyset \cup \bigcup_i \texttt{[}D^i\texttt{]} & \overset{\Omega}{\Longleftrightarrow} \emptyset \cup \bigcup_i \texttt{[}D^i\texttt{]} \\ \text{i.e.,} & \texttt{[}D\texttt{]} & \overset{\Omega}{\Longleftrightarrow} \texttt{[}D\texttt{]} \end{array}$$

which finishes the proof. □

7.7 Lemma [Well-behavedness]: $\forall D \subseteq \mathcal{T}.\ \texttt{list\_reverse} \in \texttt{[}D\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}$

Proof. First, note that `list_reverse` is the fixed point of the function:

$$f = \lambda l.\ \texttt{wmap}\ \{\texttt{*t} \mapsto l\};\ \texttt{rotate}$$

Let $C = A = \texttt{[}D\texttt{]}$. In outline, the proof proceeds as follows. We assume that $l \in C \overset{\Omega}{\rightleftharpoons} A$ and prove that $f(l) \in C \overset{\Omega}{\rightleftharpoons} A$. Using Corollary 3.17(1), we conclude that $\mathit{fix}(f) = \texttt{list\_reverse} \in C \overset{\Omega}{\rightleftharpoons} A$.

The outermost lens combinator is the composition operator. Thus, we must show that the `wmap` instance has type $C \overset{\Omega}{\rightleftharpoons} B$ and that $\texttt{rotate} \in B \overset{\Omega}{\rightleftharpoons} A$ for some type B. We will prove these facts for $B = \texttt{[}D\texttt{]}$. Let m be the total function from names to lenses that maps `*t` to l and every other name to `id`. We first show that $m \in \Pi n \in \mathcal{N}.\ C(n) \overset{\Omega}{\rightleftharpoons} B(n)$:

$$\begin{array}{rrr} & m(\texttt{*h}) = \texttt{id} \in & C(\texttt{*h}) \overset{\Omega}{\rightleftharpoons} B(\texttt{*h}) \\ \text{i.e.,} & & \texttt{[}D\texttt{]}(\texttt{*h}) \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}(\texttt{*h}) \\ \text{i.e.,} & & D \overset{\Omega}{\rightleftharpoons} D \end{array}$$

by the type of `id`;

$$\begin{array}{rrr} & m(\texttt{*t}) = l \in & C(\texttt{*t}) \overset{\Omega}{\rightleftharpoons} B(\texttt{*t}) \\ \text{i.e.,} & & \texttt{[}D\texttt{]}(\texttt{*t}) \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}(\texttt{*t}) \\ \text{i.e.,} & & \texttt{[}D\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]} \end{array}$$

by assumption;

$$\begin{array}{rrrr} & m(n) = \texttt{id} \in & C(n) \overset{\Omega}{\rightleftharpoons} B(n) & \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & & \texttt{[}D\texttt{]}(n) \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}(n) & \\ \text{i.e.,} & & \emptyset \overset{\Omega}{\rightleftharpoons} \emptyset & \end{array}$$

vacuously.

Hence m has the correct type. The type of `wmap` also requires that C and B be shuffle closed and that $\mathsf{doms}(C) = \mathsf{doms}(B)$. The first follows from Lemma 7.2(2) and since $C = B = \texttt{[}D\texttt{]}$.

Next, using the type we just proved for `rotate` we have that

$$\begin{array}{rr} \texttt{rotate} \in & B \stackrel{\Omega}{\rightleftharpoons} A \\ \text{i.e.,} & \texttt{[}D\texttt{]} \stackrel{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]} \end{array}$$

Finally, by the type of the composition operator, we conclude that $f(l) \in C \stackrel{\Omega}{\rightleftharpoons} A$. By Corollary 3.17(1), `list_reverse` has the same type, $C \stackrel{\Omega}{\rightleftharpoons} A$. □

7.8 Lemma [Totality]: $\forall D \subseteq \mathcal{T}.\ \texttt{list\_reverse} \in \texttt{[}D\texttt{]} \stackrel{\Omega}{\Longleftrightarrow} \texttt{[}D\texttt{]}$

Proof. The proof, in outline, is as follows. Let $C = A = \texttt{[}D\texttt{]}$. We first note that `list_reverse` is the fixed point of the function f, defined above in the well-behavedness proof. We then identify two increasing chains of types, C_i and A_i such that $C = \bigcup_i C_i$ and $A = \bigcup_i A_i$. We then prove, for all i, that $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$ assuming that $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$. By Corollary 3.17(2), we conclude that $\mathit{fix}(()f) \in \bigcup_i C_i \stackrel{\Omega}{\Longleftrightarrow} \bigcup_i A_i$; i.e., that $\texttt{list\_reverse} \in C \stackrel{\Omega}{\Longleftrightarrow} A$.

Let C_i and A_i be increasing chains of types:

$$\begin{array}{l} C_0 = A_0 = \emptyset \\ C_{i+1} = A_{i+1} = \texttt{[}D^i\texttt{]}. \end{array}$$

We prove $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$ implies $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$ by induction on i.

For the base case, $i = 0$, we have $C_1 = A_1 = \texttt{[]}$. The outermost lens in $f(l)$ is the composition operator. Thus, we must show that the `wmap` instance has type $C_1 \stackrel{\Omega}{\Longleftrightarrow} B$ and that $\texttt{rotate} \in B \stackrel{\Omega}{\Longleftrightarrow} A_1$ for some type B. Let $B = \texttt{[]}$. We first prove that $m \in \Pi n \in \mathcal{N}.\ C_1(n) \stackrel{\Omega}{\Longleftrightarrow} B(n)$:

$$\begin{array}{rr} & m(n) = \texttt{id} \in C_1(n) \stackrel{\Omega}{\Longleftrightarrow} B(n) \quad \forall n \in \mathcal{N} \\ \text{i.e.,} & \texttt{[]}(n) \stackrel{\Omega}{\Longleftrightarrow} \texttt{[]}(n) \\ \text{i.e.,} & \emptyset \stackrel{\Omega}{\Longleftrightarrow} \emptyset \\ & \text{vacuously.} \end{array}$$

Hence m has the correct type. The type of `wmap` also requires that C_1 and B be shuffle closed and that $\mathsf{doms}(C_1) = \mathsf{doms}(B)$. Both facts are immediate as $\texttt{[]} = \{\{\!|\}\!\}\}$.

Next, using the total type we proved for `rotate` we have

$$\begin{array}{rr} \texttt{rotate} \in & B \stackrel{\Omega}{\Longleftrightarrow} A_1 \\ \text{i.e.,} & \texttt{[]} \stackrel{\Omega}{\Longleftrightarrow} \texttt{[]} \end{array}$$

By the type of the composition operator, we conclude that $f(l) \in C_1 \stackrel{\Omega}{\Longleftrightarrow} A_1$.

For the induction step, assume $i > 0$ and that $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$. Again, outermost lens in $f(l)$ is the composition operator. Let $B = \texttt{[}D^i\texttt{]}$. We first prove that the `wmap` lens has type $C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} B$. To show that that $m \in \Pi n \in \mathcal{N}.\ C_{i+1}(n) \stackrel{\Omega}{\Longleftrightarrow} B(n)$ we

argue as follows:

$$\begin{array}{rrl} & m(\texttt{*h}) = \texttt{id} \in & C_{i+1}(\texttt{*h}) \overset{\Omega}{\Longleftrightarrow} B(\texttt{*h}) \\ \text{i.e.,} & & [D^i](\texttt{*h}) \overset{\Omega}{\Longleftrightarrow} [D^i](\texttt{*h}) \\ \text{i.e.,} & & D \overset{\Omega}{\Longleftrightarrow} D \end{array}$$

as $i > 0$ and by the type of `id`;

$$\begin{array}{rrl} & m(\texttt{*t}) = l \in & C_{i+1}(\texttt{*t}) \overset{\Omega}{\Longleftrightarrow} B(\texttt{*t}) \\ \text{i.e.,} & & [D^i](\texttt{*t}) \overset{\Omega}{\Longleftrightarrow} [D^i](\texttt{*t}) \\ \text{i.e.,} & & [D^i] \overset{\Omega}{\Longleftrightarrow} [D^i] \end{array}$$

as $i > 0$;

$$\text{i.e.,} \quad C_i \overset{\Omega}{\Longleftrightarrow} A_i$$

by induction hypothesis;

$$\begin{array}{rrll} & m(n) = \texttt{id} \in & C_{i+1}(n) \overset{\Omega}{\Longleftrightarrow} B(n) & \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & & [D^i](n) \overset{\Omega}{\Longleftrightarrow} [D^i](n) & \\ \text{i.e.,} & & \emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset & \end{array}$$

vacuously.

Hence m has the correct type. The type of `wmap` also requires that C_{i+1} and B be shuffle closed and that $\mathsf{doms}(C_{i+1}) = \mathsf{doms}(B)$. These facts follow from Lemma 7.2(2) and since $C_{i+1} = B = [D^i]$.

Next, using the total type we proved for `rotate` we have

$$\begin{array}{rrl} & \texttt{rotate} \in & B \overset{\Omega}{\Longleftrightarrow} A_{i+1} \\ \text{i.e.,} & & [D^i] \overset{\Omega}{\Longleftrightarrow} [D^i] \end{array}$$

By the type of the composition operator, we conclude that $f(l) \in C_{i+1} \overset{\Omega}{\Longleftrightarrow} A_{i+1}$.

Finally, by Corollary 3.17(2), we conclude that

$$\begin{array}{rrl} \mathit{fix}(f) = \texttt{list\_reverse} \in & & \bigcup_i C_i \overset{\Omega}{\Longleftrightarrow} \bigcup_i A_i \\ \text{i.e.,} & & \emptyset \cup \bigcup_i [D^i] \overset{\Omega}{\Longleftrightarrow} \emptyset \cup \bigcup_i [D^i] \\ \text{i.e.,} & & [D] \overset{\Omega}{\Longleftrightarrow} [D] \end{array}$$

as required. □

7.9 Lemma [Well-behavedness]:
$\forall D \subseteq \mathcal{T} \quad \texttt{group} \in [D] \overset{\Omega}{=\!=} [D :: D :: []]\texttt{++}([] \cup ((D :: []) :: []))$

Proof. First, note that `group` is the fixed point of the function:

```
f = λl. acond [] []
          id
          (acond (D :: []) ((D :: []) :: [])
            (plunge *h; add *t {|})
            (rename *h tmp;
             hoist_nonunique *t {*h, *t};
             fork {*t}
               (map l)
               (xfork {*h } {*t} (add *t {|; plunge *t) (rename tmp *h);
                plunge *h)))
```

To shorten the proof, we will use the following abbreviations:

$$\begin{array}{lll} S = (D :: \texttt{[]}) :: \texttt{[]} & & P = D :: D :: \texttt{[]} \\ C = \texttt{[}D\texttt{]} & & A = \texttt{[}P\texttt{]++(}\texttt{[]} \cup S\texttt{)} \end{array}$$

In outline, the proof proceeds as follows. We assume that $l \in C \stackrel{\Omega}{\rightleftharpoons} A$ and prove that $f(l) \in C \stackrel{\Omega}{\rightleftharpoons} A$. Using Corollary 3.17(1), we conclude that $\mathit{fix}(f) = \texttt{group} \in C \stackrel{\Omega}{\rightleftharpoons} A$.

The outermost combinator is an `acond`. Thus, we must show that each branch has the correct type. For the first, we have

$$\begin{array}{rrl} & \texttt{id} \in & C \cap \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} A \cap \texttt{[]} \\ \text{i.e.,} & & \texttt{[}D\texttt{]} \cap \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} (\texttt{[}P\texttt{]++(}\texttt{[]} \cup S\texttt{)}) \cap \texttt{[]} \\ \text{i.e.,} & & \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} \texttt{[]} \end{array}$$

using the type of `id`. For the second we must show that the nested `acond` has lens type

$$\begin{array}{rl} & C \setminus \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} A \setminus \texttt{[]} \\ \text{i.e.,} & \texttt{[}D\texttt{]} \setminus \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} (\texttt{[}P\texttt{]++(}\texttt{[]} \cup S\texttt{)}) \setminus \texttt{[]} \\ \text{i.e.,} & D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{\rightleftharpoons} ((P :: \texttt{[}P\texttt{]})\texttt{++(}\texttt{[]} \cup S\texttt{)}) \cup S \end{array}$$

Again, using the type of `acond`, we must show that each branch of the nested conditional has the correct type. For the first branch, we must show:

$$\begin{array}{rl} & (\texttt{plunge *h; add *t } \{\!|\}) \\ & \in (D :: \texttt{[}D\texttt{]}) \cap (D :: \texttt{[]}) \stackrel{\Omega}{\rightleftharpoons} (((P :: \texttt{[}P\texttt{]})\texttt{++(}\texttt{[]} \cup S\texttt{)}) \cup S) \cap ((D :: \texttt{[]}) :: \texttt{[]}) \\ \text{i.e.,} & D :: \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} (D :: \texttt{[]}) :: \texttt{[]} \end{array}$$

which follows from the types of `plunge`, `add` and the composition operator. For the second branch we must prove

$$\begin{array}{rl} & \texttt{rename *h tmp;} \\ & \texttt{hoist\_nonunique *t \{*h, *t\};} \\ & \texttt{fork \{*t\}} \\ & \quad \texttt{(map } l\texttt{)} \\ & \quad \texttt{(xfork \{*h \} \{*t\} (add *t } \{\!|\}\texttt{; plunge *t) (rename tmp *h);} \\ & \quad \texttt{plunge *h)} \\ & \in (D :: \texttt{[}D\texttt{]}) \setminus (D :: \texttt{[]}) \stackrel{\Omega}{\rightleftharpoons} (((P :: \texttt{[}P\texttt{]})\texttt{++(}\texttt{[]} \cup S\texttt{)}) \cup S) \setminus ((D :: \texttt{[]}) :: \texttt{[]}) \\ \text{i.e.,} & D :: D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{\rightleftharpoons} (P :: \texttt{[}P\texttt{]})\texttt{++(}\texttt{[]} \cup S\texttt{)} \end{array}$$

From the type of `rename` we have

$$\texttt{rename *h tmp} \in D :: D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: \texttt{[}D\texttt{]}|\!\}$$

Moreover, using the type of `hoist_nonunique` we have

$$\begin{array}{l} \texttt{hoist\_nonunique *t \{*h, *t\}} \\ \quad \in \{\!|\texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: \texttt{[}D\texttt{]}|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D\texttt{]}|\!\} \end{array}$$

To calculate the type of the `fork` lens, we check the types of each arm. The first arm is (`map` l). We prove that

$$\texttt{map } l \in \{\!|\texttt{*t} \mapsto \texttt{[}D\texttt{]}|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*t} \mapsto \texttt{[}P\texttt{]++(}\texttt{[]} \cup S\texttt{)}|\!\}$$

by checking that l has the correct type:

$$l \in \bigcap_{n \in \mathcal{N}} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\}(n) \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}P\texttt{]++([]} \cup S\texttt{)}\right|\!\right\}(n)$$

as follows:

$$\begin{array}{rl} & l \in \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\}(\texttt{*t}) \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}P\texttt{]++([]} \cup S\texttt{)}\right|\!\right\}(\texttt{*t}) \\ \text{i.e.,} & \texttt{[}D\texttt{]} \stackrel{\Omega}{=\!=} \texttt{[}P\texttt{]++([]} \cup S\texttt{)} \end{array}$$

by assumption;

$$\begin{array}{rl} & l \in \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\}(n) \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}P\texttt{]++([]} \cup S\texttt{)}\right|\!\right\}(n) \quad \forall \quad \forall n \neq \texttt{*t} \\ \text{i.e.,} & \emptyset \stackrel{\Omega}{=\!=} \emptyset \end{array}$$

vacuously.

Moreover, the types $\left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\}$ and $\left\{\!\left|\texttt{*t} \mapsto \texttt{[}P\texttt{]++([]} \cup S\texttt{)}\right|\!\right\}$ are both shuffle closed and have equal domain sets, since every tree in both types has a singleton domain $\{\texttt{*t}\}$. Hence, by the type of `map`, the first arm of the `fork` has the correct type.

For the second arm of the `fork`, we first show that

$$\begin{array}{rl} & \texttt{xfork \{*h \} \{*t\} (add *t \{\!|\}; plunge *t) (rename tmp *h)} \\ & \in \left\{\!\left|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D\right|\!\right\} \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*h} \mapsto D, \texttt{*t} \mapsto \left\{\!\left|\texttt{*h} \mapsto D, \texttt{*t} \mapsto \{\!|\}\right|\!\right\}\right|\!\right\} \\ \text{i.e.,} & \left\{\!\left|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D\right|\!\right\} \stackrel{\Omega}{=\!=} P \end{array}$$

and that

$$\texttt{plunge *h} \in P \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*h} \mapsto P\right|\!\right\}$$

With the type of the composition operator we have that the composition of the `xfork` and `plunge` lenses has type

$$\left\{\!\left|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D\right|\!\right\} \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*h} \mapsto P\right|\!\right\}$$

Putting these pieces together we have that the `fork` lens has type

$$\begin{array}{rl} & \left\{\!\left|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\} \stackrel{\Omega}{=\!=} \left\{\!\left|\texttt{*h} \mapsto P, \texttt{*t} \mapsto \texttt{[}P\texttt{]++([]} \cup S\texttt{)}\right|\!\right\} \\ \text{i.e.,} & \left\{\!\left|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D\texttt{]}\right|\!\right\} \stackrel{\Omega}{=\!=} (P :: \texttt{[}P\texttt{]})\texttt{++([]} \cup S\texttt{)} \end{array}$$

Thus, using the types calculated previously for the `rename` and `hoist_nonunique` lenses, together with the type of the composition operator, we have that the second branch of the nested `acond` has type

$$D :: D :: \texttt{[}D\texttt{]} \stackrel{\Omega}{=\!=} (P :: \texttt{[}P\texttt{]})\texttt{++([]} \cup S\texttt{)}$$

as required.

Hence, using the type for the outer `acond`, we conclude that $f(l) \in C \stackrel{\Omega}{=\!=} A$ and by Corollary 3.17(1), that $\mathit{fix}(f) = \texttt{group}$ has the same type, $C \stackrel{\Omega}{=\!=} A$. □

7.10 Lemma [Totality]:
$\forall D \subseteq \mathcal{T} \quad \texttt{group} \in \texttt{[}D\texttt{]} \stackrel{\Omega}{\Longleftrightarrow} \texttt{[}D :: D :: \texttt{[]]++([]} \cup ((D :: \texttt{[]}) :: \texttt{[]}))$

Proof. The proof, in outline, is as follows. We first note that `group` is the fixed point of the function f, defined in the well-behavedness proof above. We then identify two increasing chains of types, C_i and A_i and prove for all i, that $f(l) \in$

$C_{i+1} \overset{\Omega}{\Longleftrightarrow} A_{i+1}$ assuming that $l \in C_i \overset{\Omega}{\Longleftrightarrow} A_i$. By Corollary 3.17(2), we conclude that $\mathit{fix}(f) \in \bigcup_i C_i \overset{\Omega}{\Longleftrightarrow} \bigcup_i A_i$.

We use the same abbreviations for S and P as in the well-behavedness proof. Define two chains of types:

$$\begin{array}{l} C_0 = A_0 = \emptyset \\ C_{i+1} = \texttt{[}D^i\texttt{]} \\ A_{i+1} = \begin{cases} \texttt{[}P^{i/2}\texttt{]} & \text{if } i \text{ is even} \\ (\texttt{[}P^{\lfloor i/2 \rfloor}\texttt{]})\texttt{++}S & \text{otherwise} \end{cases} \end{array}$$

We prove $l \in C_i \overset{\Omega}{\Longleftrightarrow} A_i$ implies $f(l) \in C_{i+1} \overset{\Omega}{\Longleftrightarrow} A_{i+1}$ by induction on i.

For the base case $i = 0$ we must show that the `acond` lens has type $C_1 \overset{\Omega}{\Longleftrightarrow} A_1$. The required type for the first branch is

$$\begin{array}{rr} & \texttt{id} \in C_1 \cap \texttt{[]} \overset{\Omega}{\Longleftrightarrow} A_1 \cap \texttt{[]} \\ \text{i.e.,} & \texttt{[]} \cap \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \cap \texttt{[]} \\ \text{i.e.,} & \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \end{array}$$

which is immediate by the type of `id`. For the second branch we must show that the nested `acond` lens has type

$$\begin{array}{rr} & C_1 \setminus \texttt{[]} \overset{\Omega}{\Longleftrightarrow} A_1 \setminus \texttt{[]} \\ \text{i.e.,} & \texttt{[]} \setminus \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \setminus \texttt{[]} \\ \text{i.e.,} & \emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset \end{array}$$

which holds vacuously. Thus, by the type of `acond` we have $f(l) \in C_1 \overset{\Omega}{\Longleftrightarrow} A_1$, which finishes the case.

For the induction step, we assume $i > 0$ and analyze the type of the outermost `acond` lens. We must show that each branch has the correct type. For the first branch, we calculate the required type as

$$\begin{array}{rrll} & \texttt{id} \in C_{i+1} \cap \texttt{[]} \overset{\Omega}{\Longleftrightarrow} A_{i+1} \cap \texttt{[]} & & \\ \text{i.e.,} & \texttt{[}D^i\texttt{]} \cap \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}P^{i/2}\texttt{]} \cap \texttt{[]} & & \text{if } i \text{ even} \\ \text{i.e.,} & \texttt{[}D^i\texttt{]} \cap \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}P^{i/2}\texttt{]++}S \cap \texttt{[]} & & \text{if } i \text{ is odd} \\ \text{i.e.,} & \emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset & & \text{in either case, since } i > 0 \end{array}$$

which holds vacuously.

Similarly, we calculate the required type for the second branch (i.e., the `acond` lens) as follows

$$\begin{array}{rrll} & C_{i+1} \setminus \texttt{[]} \overset{\Omega}{\Longleftrightarrow} A_{i+1} \setminus \texttt{[]} & & \\ \text{i.e.,} & \texttt{[}D^i\texttt{]} \setminus \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}P^{i/2}\texttt{]} \setminus \texttt{[]} & & \text{if } i \text{ even} \\ \text{i.e.,} & D :: \texttt{[}D^{i-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} P :: \texttt{[}P^{(i/2)-1}\texttt{]} & & \\ \text{i.e.,} & \texttt{[}D^i\texttt{]} \setminus \texttt{[]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}P^{\lfloor i/2 \rfloor}\texttt{]++}S \setminus \texttt{[]} & & \text{otherwise} \\ \text{i.e.,} & D :: \texttt{[}D^{i-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}P^{\lfloor i/2 \rfloor - 1}\texttt{]++}S & & \end{array}$$

Since this lens is also an `acond`, we must analyze the type of its branches. For the first branch, (`plunge *h; add *t {}`), we calculate the required type as

$$\begin{array}{lrcll} & (D :: [D^{i-1}]) \cap (D :: []) & \overset{\Omega}{\Longleftrightarrow} & (P :: [P^{(i/2)-1}]) \cap ((D :: []) :: []) & \text{if } i \text{ even} \\ \text{i.e.,} & \emptyset & \overset{\Omega}{\Longleftrightarrow} & \emptyset & \\ & (D :: [D^{i-1}]) \cap (D :: []) & \overset{\Omega}{\Longleftrightarrow} & ((P :: [P^{\lfloor i/2 \rfloor -1}])\texttt{++}S) \cap ((D :: []) :: []) & \text{otherwise} \\ \text{i.e.,} & D :: [] & \overset{\Omega}{\Longleftrightarrow} & (D :: []) :: [] & \end{array}$$

The case where i is even is immediate, since every lens has that type vacuously; the other case follows from the types of `plunge`, `add` and the composition operator. For the second branch we must show that

```
rename *h tmp;
hoist_nonunique *t {*h, *t};
fork {*t}
  (map l)
  (xfork {*h } {*t} (add *t {}; plunge *t) (rename tmp *h);
   plunge *h)
```

$$\begin{array}{lrcll} \in & (D :: [D^{i-1}]) \setminus (D :: []) & \overset{\Omega}{\Longleftrightarrow} & (P :: [P^{(i/2)-1}]) \setminus ((D :: []) :: []) & \text{if } i \text{ even} \\ \text{i.e.,} & D :: D :: [D^{i-2}] & \overset{\Omega}{\Longleftrightarrow} & P :: [P^{(i/2)-1}] & \\ & (D :: [D^{i-1}]) \setminus (D :: []) & \overset{\Omega}{\Longleftrightarrow} & ((P :: [P^{\lfloor i/2 \rfloor -1}])\texttt{++}S) \setminus ((D :: []) :: []) & \text{otherwise} \\ \text{i.e.,} & D :: D :: [D^{i-2}] & \overset{\Omega}{\Longleftrightarrow} & (P :: [P^{\lfloor i/2 \rfloor -1}])\texttt{++}S & \end{array}$$

There are several cases. If $i = 1$ then we have the lens typing vacuously. Otherwise, $i > 1$ and we calculate the types for each lens in the composition. From the type of `rename` we have

$$\texttt{rename *h tmp} \in D :: D :: [D^{i-2}] \overset{\Omega}{\Longleftrightarrow} \left\{\!\left| \texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: [D^{i-2}] \right|\!\right\}$$

and, moreover, using the type of `hoist_nonunqique` we have

$$\begin{array}{l} \texttt{hoist\_nonunique *t \{*h, *t\}} \\ \quad \in \left\{\!\left| \texttt{tmp} \mapsto D, \texttt{*t} \mapsto D :: [D^{i-2}] \right|\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\left| \texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto [D^{i-2}] \right|\!\right\} \end{array}$$

Using the type of `fork`, we then verify the types of each arm. The first arm is (`map` l). We prove that

$$\begin{array}{rcll} \texttt{map } l \in \left\{\!\left| \texttt{*t} \mapsto [D^{i-2}] \right|\!\right\} & \overset{\Omega}{\Longleftrightarrow} & \left\{\!\left| \texttt{*t} \mapsto [P^{(i-2)/2}] \right|\!\right\} & \text{if } i \text{ even} \\ \left\{\!\left| \texttt{*t} \mapsto [D^{i-2}] \right|\!\right\} & \overset{\Omega}{\Longleftrightarrow} & \left\{\!\left| \texttt{*t} \mapsto [P^{\lfloor i-2 \rfloor /2}]\texttt{++}S \right|\!\right\} & \text{otherwise} \end{array}$$

by checking that l has the correct type:

$$\begin{array}{rcll} l \in \bigcap_{n \in \mathcal{N}} \left\{\!\left| \texttt{*t} \mapsto [D] \right|\!\right\}(n) & \overset{\Omega}{\Longleftrightarrow} & \left\{\!\left| \texttt{*t} \mapsto [P^{(i-2)/2}] \right|\!\right\}(n) & \text{if } i \text{ even} \\ \bigcap_{n \in \mathcal{N}} \left\{\!\left| \texttt{*t} \mapsto [D] \right|\!\right\}(n) & \overset{\Omega}{\Longleftrightarrow} & \left\{\!\left| \texttt{*t} \mapsto [P^{\lfloor i-2 \rfloor /2}]\texttt{++}S \right|\!\right\}(n) & \text{otherwise} \end{array}$$

For every n not equal to `*t`, this reduces to $l \in \emptyset \overset{\Omega}{\iff} \emptyset$, which vacuously holds. Otherwise we calculate as follows:

$$\begin{array}{rrcll} & l \in \{\!|\texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\}\,(\texttt{*t}) & \overset{\Omega}{\iff} & \{\!|\texttt{*t} \mapsto \texttt{[}P^{(i-2)/2}\texttt{]}|\!\}\,(\texttt{*t}) & \text{if } i \text{ is even} \\ \text{i.e.,} & \texttt{[}D^{i-2}\texttt{]} & \overset{\Omega}{\iff} & \texttt{[}P^{(i-2)/2}\texttt{]} & \\ \text{i.e.,} & C_{i-1} & \overset{\Omega}{\iff} & A_{i-1} & \end{array}$$

$$\begin{array}{rrcll} & \{\!|\texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\}\,(\texttt{*t}) & \overset{\Omega}{\iff} & \{\!|\texttt{*t} \mapsto (\texttt{[}P^{\lfloor (i-2)/2 \rfloor}\texttt{]})\texttt{++}S|\!\}\,(\texttt{*t}) & \text{otherwise} \\ \text{i.e.,} & \texttt{[}D^{i-2}\texttt{]} & \overset{\Omega}{\iff} & (\texttt{[}P^{\lfloor (i-2)/2 \rfloor}\texttt{]})\texttt{++}S & \\ \text{i.e.,} & C_{i-1} & \overset{\Omega}{\iff} & A_{i-1} & \end{array}$$

both facts follow by induction hypothesis. Then, since the domain of every tree in source and target component of the lens type we want to show for the `map` is $\{\texttt{*t}\}$, we have that the types are shuffle closed and have equal domain. Thus, by the type of `map`, the first arm has the correct type.

For the second arm we first prove that

$$\begin{array}{rl} & \texttt{xfork \{*h \} \{*t\} (add *t \{|\}; plunge *t) (rename tmp *h)} \\ \in & \{\!|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D|\!\} \overset{\Omega}{\iff} \{\!|\texttt{*h} \mapsto D, \texttt{*t} \mapsto \{\!|\texttt{*h} \mapsto D, \texttt{*t} \mapsto \{|\}|\!\}|\!\} \\ \text{i.e.,} & \{\!|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D|\!\} \overset{\Omega}{\iff} P \end{array}$$

and that

$$\texttt{plunge *h} \in P \overset{\Omega}{\iff} \{\!|\texttt{*h} \mapsto P|\!\}$$

using the type of `plunge`. With the type of the composition operator, we have that the composition of the `xfork` and `plunge` lenses has type

$$\{\!|\texttt{*tmp} \mapsto D, \texttt{*h} \mapsto D|\!\} \overset{\Omega}{\iff} \{\!|\texttt{*h} \mapsto P|\!\}$$

Putting these pieces together we have that the `fork` lens has type

$$\begin{array}{rrcll} & \{\!|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\} & \overset{\Omega}{\iff} & \{\!|\texttt{*h} \mapsto P, \texttt{*t} \mapsto \texttt{[}P^{(i/2)-1}\texttt{]}|\!\} & \text{if } i \text{ is even} \\ \text{i.e.,} & \{\!|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\} & \overset{\Omega}{\iff} & P\texttt{::[}P^{(i/2)-1}\texttt{]} & \end{array}$$

$$\begin{array}{rrcll} & \{\!|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\} & \overset{\Omega}{\iff} & \{\!|\texttt{*h} \mapsto P, \texttt{*t} \mapsto \texttt{[}P^{\lfloor i/2 \rfloor -1}\texttt{]++}S|\!\} & \text{otherwise} \\ & \{\!|\texttt{tmp} \mapsto D, \texttt{*h} \mapsto D, \texttt{*t} \mapsto \texttt{[}D^{i-2}\texttt{]}|\!\} & \overset{\Omega}{\iff} & P\texttt{::[}P^{\lfloor i/2 \rfloor -1}\texttt{]++}S & \end{array}$$

Thus, using the type of the composition operator along with the types we proved for `rename`, `hoist_nonunique`, and `fork`, we have that the second branch of the inner `acond` belongs to

$$\begin{array}{rrcll} & D\texttt{::[}D^{i-1}\texttt{]} & \overset{\Omega}{\iff} & P\texttt{::[}P^{(i/2)-1}\texttt{]} & \text{if } i \text{ even} \\ & D\texttt{::[}D^{i-1}\texttt{]} & \overset{\Omega}{\iff} & \texttt{[}P^{\lfloor i/2 \rfloor -1}\texttt{]++}S & \text{otherwise} \end{array}$$

as required.

By the type of the composition operator, we have $f(l) \in C_{i+1} \overset{\Omega}{\iff} A_{i+1}$, which finishes the case and the inductive proof.

By Corollary 3.17(2), we conclude that

$$\begin{array}{rrcl} \mathit{fix}(f) = \texttt{group} \in & \bigcup_i C_i & \overset{\Omega}{\iff} & \bigcup_i A_i \\ \text{i.e.,} & \emptyset \cup \bigcup_i \texttt{[}D^i\texttt{]} & \overset{\Omega}{\iff} & \emptyset \cup \bigcup_i \texttt{[}D^i\texttt{]} \\ \text{i.e.,} & \texttt{[}D\texttt{]} & \overset{\Omega}{\iff} & \texttt{[}P\texttt{]++([]} \cup S\texttt{)} \end{array}$$

which completes the proof. □

7.11 Lemma [Well-behavedness]:
$\forall D{\subseteq}\mathcal{T}, t \in \mathcal{T}.\ \textit{with}\ t \notin D.\quad \texttt{concat}\ t \in \texttt{[}D\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]++}(t :: \texttt{[}D\texttt{]})$

PROOF. First, note that `concat` t is the fixed point of the function:

$$\begin{aligned} f = \lambda l.\ &\texttt{acond}\ (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]})\ (t :: \texttt{[}D\texttt{]}) \\ &(\texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{const}\ t\ \texttt{[]}, \texttt{*t} \mapsto \texttt{hd []}\}) \\ &(\texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{hoist *h};\ \texttt{rename *t tmp}); \\ &\ \texttt{fork}\ \{\texttt{*h}\}\ \texttt{id}\ (\texttt{rename tmp *h};\ l;\ \texttt{plunge *t})) \end{aligned}$$

Let $C = \texttt{[}D\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[]}$ and $A = \texttt{[}D\texttt{]++}(t :: \texttt{[}D\texttt{]})$. We assume that $l \in C \stackrel{\Omega}{\rightleftharpoons} A$ and prove that $f(l) \in C \stackrel{\Omega}{\rightleftharpoons} A$. Using Corollary 3.17(1), we conclude that $\mathit{fix}(f) = \texttt{concat}\ t \in C \stackrel{\Omega}{\rightleftharpoons} A$.

The outermost lens is an `acond`. Thus, we must show that each branch has the correct type. For the first, we have

$$\begin{array}{lr} & \texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{const}\ t\ \texttt{[]}, \texttt{*t} \mapsto \texttt{hd []}\} \\ & \in \quad C \cap (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \stackrel{\Omega}{\rightleftharpoons} A \cap (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (\texttt{[}D\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \cap (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \stackrel{\Omega}{\rightleftharpoons} (\texttt{[}D\texttt{]++}(t :: \texttt{[}D\texttt{]})) \cap (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} t :: \texttt{[}D\texttt{]} \end{array}$$

Let m be the total function from names to lenses that maps `*h` to (`const` t `[]`) and `*t` to (`hd []`) and every other name to `id`. We prove that m has the correct type, $\Pi n \in \mathcal{N}.\, \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(n) \stackrel{\Omega}{\rightleftharpoons} t :: \texttt{[}D\texttt{]}(n)$, as follows

$$\begin{array}{lr} & m(\texttt{*h}) = \texttt{const}\ t\ \texttt{[]} \in \ \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(\texttt{*h}) \stackrel{\Omega}{\rightleftharpoons} t :: \texttt{[}D\texttt{]}(\texttt{*h}) \\ \text{i.e.,} & \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} t \end{array}$$

by the type of `const`;

$$\begin{array}{lr} & m(\texttt{*t}) = \texttt{hd []} \in \ \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(\texttt{*t}) \stackrel{\Omega}{\rightleftharpoons} t :: \texttt{[}D\texttt{]}(\texttt{*t}) \\ \text{i.e.,} & \texttt{[}D\texttt{]} :: \texttt{[]} \stackrel{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]} \end{array}$$

by the type of `hd`;

$$\begin{array}{lr} & m(n) = \texttt{id} \in \quad \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(n) \stackrel{\Omega}{\rightleftharpoons} t :: \texttt{[}D\texttt{]}(n) \qquad \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & \emptyset \stackrel{\Omega}{\rightleftharpoons} \emptyset \end{array}$$

vacuously.

Additionally, since both the source and targets types are cons cells, they have equal domains and are shuffle closed by Lemma 7.2. Putting all these facts together, we have that `wmap` has the type calculated above.

For the second branch, we must prove that

$$\begin{array}{lr} & \texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{hoist *h};\ \texttt{rename *t tmp}); \\ & \texttt{fork}\ \{\texttt{*h}\}\ \texttt{id}\ (\texttt{rename tmp *h};\ l;\ \texttt{plunge *t}) \\ & \in \quad C \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \stackrel{\Omega}{\rightleftharpoons} A \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (\texttt{[}D\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \stackrel{\Omega}{\rightleftharpoons} (\texttt{[}D\texttt{]++}(t :: \texttt{[}D\texttt{]})) \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (D :: \texttt{[}D\texttt{]}) :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]} \stackrel{\Omega}{\rightleftharpoons} (D :: \texttt{[}D\texttt{]})\texttt{++}(t :: \texttt{[}D\texttt{]}) \end{array}$$

We calculate the type of the first `fork`. The first arm has type

$$\texttt{id} \in \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]} :: \texttt{[]}\right|\!\right\} \stackrel{\Omega}{\rightleftharpoons} \left\{\!\left|\texttt{*t} \mapsto \texttt{[}D\texttt{]} :: \texttt{[]}\right|\!\right\}$$

and the second arm has type

$$(\texttt{hoist *h; rename *t tmp}) \in \{\!|\texttt{*h} \mapsto D :: [D]|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*h} \mapsto D, \texttt{tmp} \mapsto [D]|\!\}$$

using the types of `id`, `hoist`, `rename` and the composition operator. With these facts and the type of `fork` we have

$$\begin{array}{l}\texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{hoist *h; rename *t tmp}) \\ \quad \in (D :: [D]) :: [D] :: [] \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*h} \mapsto D, \texttt{tmp} \mapsto [D], \texttt{*t} \mapsto [D] :: []|\!\}\end{array}$$

For the first arm of the next `fork` we have

$$\texttt{id} \in \{\!|\texttt{*h} \mapsto D|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*h} \mapsto D|\!\}$$

and moreover, for the second arm, we have

$$\begin{array}{l}(\texttt{rename tmp *h};\ l;\ \texttt{plunge *t}) \\ \quad \in \{\!|\texttt{tmp} \mapsto [D],\ \texttt{*t} \mapsto [D] :: []|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*t} \mapsto \{\!|[D]\texttt{++}(t :: [D])|\!\}|\!\}\end{array}$$

using the types of `rename`, `plunge`, and the type of l we have by induction hypothesis. Thus, using the type of `fork` we have

$$\begin{array}{ll}& \texttt{fork}\ \{\texttt{*h}\}\ \texttt{id}\ (\texttt{rename tmp *h};\ l;\ \texttt{plunge *t}) \\ \in & \{\!|\texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D],\ \texttt{*t} \mapsto [D] :: []|\!\} \stackrel{\Omega}{\rightleftharpoons} \{\!|\texttt{*h} \mapsto D,\ \texttt{*t} \mapsto \{\!|[D]\texttt{++}(t :: [D])|\!\}|\!\} \\ \text{i.e.,} & \{\!|\texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D],\ \texttt{*t} \mapsto [D] :: []|\!\} \stackrel{\Omega}{\rightleftharpoons} D :: ([D]\texttt{++}(t :: [D]))\end{array}$$

as required. Hence, using the typing of the composition operator, we have that the second branch of the `acond`–the composition of both `fork`s–has the type specified above.

With the type of `acond`, we conclude that $f(l) \in C \stackrel{\Omega}{\rightleftharpoons} A$ and by Corollary 3.17(1), that $\mathit{fix}(f) = \texttt{concat}\ t$ has the same type, $C \stackrel{\Omega}{\rightleftharpoons} A$. □

7.12 Lemma [Totality]:
$\forall D \subseteq \mathcal{T}, t \in \mathcal{T}.\ \mathit{with}\ t \notin D.\quad \texttt{concat}\ t \in [D] :: [D] :: [] \stackrel{\Omega}{\Longleftrightarrow} [D]\texttt{++}(t :: [D])$

Proof. The proof, in outline, is as follows. We first note that `concat` t is the fixed point of the function f, defined in the well-behavedness proof above. We then identify two increasing chains of types, C_i and A_i and prove for all i, that $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$ assuming that $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$. By Corollary 3.17(2), we conclude that $\mathit{fix}(f) \in \bigcup_i C_i \stackrel{\Omega}{\Longleftrightarrow} \bigcup_i A_i$.

Define two chains of types:

$$\begin{array}{l}C_0 = A_0 = \emptyset \\ C_{i+1} = [D^i] :: [D] :: [] \\ A_{i+1} = [D^i]\texttt{++}(t :: [D])\end{array}$$

We prove $l \in C_i \stackrel{\Omega}{\Longleftrightarrow} A_i$ implies $f(l) \in C_{i+1} \stackrel{\Omega}{\Longleftrightarrow} A_{i+1}$ by induction on i.

For the base case, $i = 0$, we show that the outermost lens, `acond` has type $C_1 \stackrel{\Omega}{\Longleftrightarrow} A_1$ by proving that each branch has the correct type. For the first branch, we calculate the required type as follows:

$$\begin{array}{ll}& \texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{const}\ t\ [], \texttt{*t} \mapsto \texttt{hd}\ []\} \\ \in & C_1 \cap ([] :: [D] :: []) \stackrel{\Omega}{\Longleftrightarrow} A_1 \cap (t :: [D]) \\ & ([] :: [D] :: []) \cap ([] :: [D] :: []) \stackrel{\Omega}{\Longleftrightarrow} ([]\texttt{++}(t :: [D])) \cap (t :: [D]) \\ \text{i.e.,} & [] :: [D] :: [] \stackrel{\Omega}{\Longleftrightarrow} t :: [D]\end{array}$$

As above, let m be the total function from names to lenses that maps `*h` to (`const` t `[]`) and `*t` to (`hd []`) and every other name to `id`. We prove that m has the correct type, $\Pi n \in \mathcal{N}.\,\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(n) \Longleftrightarrow^{\Omega} t :: \texttt{[}D\texttt{]}(n)$, as follows

$$\begin{array}{rrl} & m(\texttt{*h}) = \texttt{const}\ t\ \texttt{[]} \in & \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(\texttt{*h}) \Longleftrightarrow^{\Omega} t :: \texttt{[}D\texttt{]}(\texttt{*h}) \\ \text{i.e.,} & & \texttt{[]} \Longleftrightarrow^{\Omega} t \end{array}$$

by the type of `const`;

$$\begin{array}{rrl} & m(\texttt{*t}) = \texttt{hd []} \in & \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(\texttt{*t}) \Longleftrightarrow^{\Omega} t :: \texttt{[}D\texttt{]}(\texttt{*t}) \\ \text{i.e.,} & & \texttt{[}D\texttt{]} :: \texttt{[]} \Longleftrightarrow^{\Omega} \texttt{[}D\texttt{]} \end{array}$$

by the type of `hd`;

$$\begin{array}{rrl} & m(n) = \texttt{id} \in & \texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}(n) \Longleftrightarrow^{\Omega} t :: \texttt{[}D\texttt{]}(n) \quad \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \text{i.e.,} & & \emptyset \Longleftrightarrow^{\Omega} \emptyset \end{array}$$

vacuously.

Additionally, since both the source and targets types are cons cells, they have equal sets of domains and are shuffle closed by Lemma 7.2. Putting these facts together, we obtain the correct type for `wmap` calculated above.

For the second branch, we must prove

$$\begin{array}{rrl} & \texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{hoist *h};\ \texttt{rename *t tmp}); & \\ & \texttt{fork}\ \{\texttt{*h}\}\ \texttt{id}\ (\texttt{rename tmp *h};\ l;\ \texttt{plunge *t}) & \\ & \in C_1 \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_1 \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \overset{\Omega}{=\!=} (\texttt{[]++}(t :: \texttt{[}D\texttt{]})) \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & \emptyset \overset{\Omega}{=\!=} \emptyset \end{array}$$

which holds, vacuously. Thus, by the type of `acond` we have $f(l) \in C_1 \Longleftrightarrow^{\Omega} A_1$, which finishes the case.

For the induction step, we assume $i > 0$ and $l \in C_i \Longleftrightarrow^{\Omega} A_i$ and show that $f(l) \in C_{i+1} \Longleftrightarrow^{\Omega} A_{i+1}$. We prove that the outermost lens, `acond` has type $C_1 \Longleftrightarrow^{\Omega} A_1$ by proving that each branch has the correct type. For the first branch, we calculate the required type as follows:

$$\begin{array}{rrl} & \texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{const}\ t\ \texttt{[]}, \texttt{*t} \mapsto \texttt{hd []}\} & \\ & \in C_{i+1} \cap (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_{i+1} \cap (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (\texttt{[}D^i\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \cap (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \Longleftrightarrow^{\Omega} (\texttt{[}D^i\texttt{]++}(t :: \texttt{[}D\texttt{]})) \cap (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & \emptyset \Longleftrightarrow^{\Omega} \emptyset \end{array}$$

This empty typing vacuously holds for any lens. For the second branch we must prove that

$$\begin{array}{rrl} & \texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{hoist *h};\ \texttt{rename *t tmp}); & \\ & \texttt{fork}\ \{\texttt{*t}\}\ \texttt{id}\ (\texttt{rename tmp *h};\ l;\ \texttt{plunge *t}) & \\ & \in C_{i+1} \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[]}) \Longleftrightarrow^{\Omega} A_{i+1} \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (\texttt{[}D^i\texttt{]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \setminus (\texttt{[]} :: \texttt{[}D\texttt{]} :: \texttt{[}\texttt{]}) \Longleftrightarrow^{\Omega} (\texttt{[}D^i\texttt{]++}(t :: \texttt{[}D\texttt{]})) \setminus (t :: \texttt{[}D\texttt{]}) \\ \text{i.e.,} & (D :: \texttt{[}D^{i-1}\texttt{]}) :: \texttt{[}D\texttt{]} :: \texttt{[]} \Longleftrightarrow^{\Omega} (D :: \texttt{[}D^{i-1}\texttt{]})\texttt{++}(t :: \texttt{[}D\texttt{]}) \end{array}$$

We calculate the type of the first `fork` directly. The first arm has type

$$\texttt{id} \in \left\{\!\!\left\{ \texttt{*t} \mapsto \texttt{[}D\texttt{]} :: \texttt{[]} \right\}\!\!\right\} \Longleftrightarrow^{\Omega} \left\{\!\!\left\{ \texttt{*t} \mapsto \texttt{[}D\texttt{]} :: \texttt{[]} \right\}\!\!\right\}$$

and the second arm has type

$$\begin{array}{l}(\texttt{hoist *h; rename *t tmp}) \\ \quad \in \left\{\!\!\left|\, \texttt{*h} \mapsto D :: [D^{i-1}] \,\right|\!\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\!\left|\, \texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D^{i-1}] \,\right|\!\!\right\}\end{array}$$

using the types of `id`, `hoist`, `rename` and the composition operator. With these facts and the type of `fork` we have

$$\begin{array}{l}\texttt{fork \{*t\} id (hoist *h; rename *t tmp)} \\ \quad \in (D :: [D^{i-1}]) :: [D] :: [] \overset{\Omega}{\Longleftrightarrow} \left\{\!\!\left|\, \texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D^{i-1}], \texttt{*t} \mapsto [D] :: [] \,\right|\!\!\right\}\end{array}$$

For the first arm of the next `fork` we have

$$\texttt{id} \in \left\{\!\!\left|\, \texttt{*h} \mapsto D \,\right|\!\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\!\left|\, \texttt{*h} \mapsto D \,\right|\!\!\right\}$$

and moreover, for the second arm, we have

$$\begin{array}{l}(\texttt{rename tmp *h; } l\texttt{; plunge *t}) \\ \quad \in \left\{\!\!\left|\, \texttt{tmp} \mapsto [D^{i-1}],\ \texttt{*t} \mapsto [D] :: [] \,\right|\!\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\!\left|\, \texttt{*t} \mapsto \left\{\!\!\left|\, [D^{i-1}]\texttt{++}(t :: [D]) \,\right|\!\!\right\} \,\right|\!\!\right\}\end{array}$$

using the types of `rename`, `plunge`, and the type of l we have by induction hypothesis. Thus, using the type of `fork` we have

$$\begin{array}{ll} & \texttt{fork \{*h\} id (rename tmp *h; } l\texttt{; plunge *t)} \\ & \in \left\{\!\!\left|\, \texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D^{i-1}],\ \texttt{*t} \mapsto [D] :: [] \,\right|\!\!\right\} \overset{\Omega}{\Longleftrightarrow} \left\{\!\!\left|\, \texttt{*h} \mapsto D,\ \texttt{*t} \mapsto [D^{i-1}]\texttt{++}(t :: [D]) \,\right|\!\!\right\} \\ \text{i.e.,} & \left\{\!\!\left|\, \texttt{*h} \mapsto D,\ \texttt{tmp} \mapsto [D^{i-1}],\ \texttt{*t} \mapsto [D] :: [] \,\right|\!\!\right\} \overset{\Omega}{\Longleftrightarrow} D :: ([D^{i-1}]\texttt{++}(t :: [D]))\end{array}$$

as required. Hence, using the typing of the composition operator, we have that the second branch of the `acond`–the composition of both `fork`s–has the total type specified above. Hence, $f(l) \in C_{i+1} \overset{\Omega}{\Longleftrightarrow} A_{i+1}$, which finishes the case and the inductive proof.

Using Corollary 3.17(2), we conclude that

$$\begin{array}{rcll} \mathit{fix}(f) & = & \texttt{concat } t \in & \bigcup_i C_i \overset{\Omega}{\Longleftrightarrow} \bigcup_i A_i \\ & \text{i.e.,} & & \emptyset \cup \bigcup_i [D^i] \overset{\Omega}{\Longleftrightarrow} \emptyset \cup \bigcup_i [D^i] \\ & \text{i.e.,} & & [D] :: [D] :: [] \overset{\Omega}{\Longleftrightarrow} [D]\texttt{++}(t :: [D])\end{array}$$

which finishes the proof. □

Special Types for Conditional Lenses

In this section, we record some additional types that our conditional lenses inhabit, which we need for our proof that `list_filter`, defined in Section 7, is total.

The first lemma presents an alternate total type for `cond` where the target sets in the types of l_1, l_2 and the entire `cond` lens are intersected with an arbitrary set, A. Recall that the standard type for `ccond` takes two lenses with type $C \cap C_1 \overset{\Omega}{\Longleftrightarrow} A_1$ and $C \setminus C_1 \overset{\Omega}{\Longleftrightarrow} A_2$ (as well as conversion functions f_{21} and f_{12}) and produces a lens with type $C \overset{\Omega}{\Longleftrightarrow} A_1 \cup A_2$. This type is usually the type that we want. However, in some situations (when reasoning about totality), we need to show a *fixed* instance of `cond` has many different types. The abstract components of some of these types may be smaller than $(A_1 \cup A_2)$, where A_1 and A_2 appear literally in the *syntax* of the `ccond` instance. The new type presented here allows us to simplify some of these cases by only considering the lens type that is intersected with the abstract type we want, reducing the proof burden.

A.28 Lemma: The `cond` lens has the following type:
$\forall C, C_1, A, A_1, A_2 \subseteq \mathcal{V}$.
$\forall l_1 \in (C \cap C_1) \overset{\Omega}{\Longleftrightarrow} (A \cap A_1)$. $\forall l_2 \in (C \backslash C_1) \overset{\Omega}{\Longleftrightarrow} (A \cap A_2)$.
$\forall f_{21} \in (C \backslash C_1) \to (C \cap C_1)_\Omega$. $\forall f_{12} \in (C \cap C_1) \to (C \backslash C_1)_\Omega$.
$\quad$ `cond` $C_1\ A_1\ A_2\ f_{21}\ f_{12}\ l_1\ l_2 \in C \overset{\Omega}{\Longleftrightarrow} (A \cap (A_1 \cup A_2))$.

Proof. We prove (1) by showing that the `cond` lens is well-behaved at $C \overset{\Omega}{=\!=} (A \cap (A_1 \cup A_2))$, and then prove (2) by showing that that the lens is also total if both l_1 and l_2 are total. We abbreviate `cond` $C_1\ A_1\ f_{21}\ f_{12}\ l_1\ l_2$ as l.

Get: Suppose $c \in C$ and $l \nearrow c$ is defined. (Again, for brevity, we write l for (`cond` $C_1\ A_1\ A_2\ f_{21}\ f_{12}\ l_1\ l_2$)). If $c \in C_1$, then $l \nearrow c = l_1 \nearrow c \in (A \cap A_1) \subseteq (A \cap (A_1 \cup A_2))$ by the type of l_1. Otherwise, $l \nearrow c = l_2 \nearrow c \in (A \cap A_2) \subseteq (A \cap (A_1 \cup A_2))$ by the type of l_2.

Put: Suppose $(a, c) \in (A \cap (A_1 \cup A_2)) \times C_\Omega$ and $l \searrow (a, c)$ is defined. There are six cases to consider, one for each clause in the definition, and the result in each case is immediate from the typing of l_1 or l_2, as the case may be. Note, in particular, that the range of f_{21} falls within the source of l_1 in the fourth clause, and similarly for f_{12} and l_2 in the sixth clause.

GetPut: Suppose $c \in C$ and $l \searrow (l \nearrow c, c)$ is defined. If $c \in C_1$, then $l \nearrow c = l_1 \nearrow c$, which, by the type of l_1, belongs to $(A \cap A_1)$. So $l \searrow (l_1 \nearrow c, c) = l_1 \searrow (l_1 \nearrow c, c)$ by either the first or the third clause in the definition of $l \searrow$. This, in turn, is equal to c by GetPut for l_1. On the other hand, if $c \notin C_1$, then $l \nearrow c = l_2 \nearrow c$, which, by the type of l_2, belongs to $(A \cap A_2)$. So $l \searrow (l_2 \nearrow c, c) = l_2 \searrow (l_2 \nearrow c, c)$ by either the second or the fourth clause in the definition of $l \searrow$. This is equal to c by GetPut for l_2.

PutGet Suppose $(a, c) \in (A \cap (A_1 \cup A_2)) \times C_\Omega$ and $l \nearrow (l \searrow (a, c))$ is defined. There are again six cases to consider:

(1) If $a \in (A \cap (A_1 \cap A_2))$ and $c \in C_1$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_1 \searrow (a, c))$. But $l_1 \searrow (a, c) \in C_1$ by the type of l_1, so $l \nearrow (l_1 \searrow (a, c)) = l_1 \nearrow (l_1 \searrow (a, c)) = a$ by PutGet for l_1.

(2) If $a \in (A \cap (A_1 \cap A_2))$ and $c \notin C_1$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_2 \searrow (a, c))$. But $l_2 \searrow (a, c) \in C_2$ by the type of l_2, so $l \nearrow (l_2 \searrow (a, c)) = l_2 \nearrow (l_2 \searrow (a, c)) = a$ by PutGet for l_2.

(3) If $a \in (A \cap (A_1 \backslash A_2))$ and $c \in (C_1)_\Omega$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_1 \searrow (a, c))$. But $l_1 \searrow (a, c) \in C_1$ by the type of l_1, so $l \nearrow (l_1 \searrow (a, c)) = l_1 \nearrow (l_1 \searrow (a, c)) = a$ by PutGet for l_1.

(4) If $a \in (A \cap (A_1 \backslash A_2))$ and $c \notin (C_1)_\Omega$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_1 \searrow (a, f_{21}(a, c)))$. But $l_1 \searrow (a, f_{21}(a, c)) \in C_1$ by the types of f_{21} and l_1, so $l \nearrow (l_1 \searrow (a, f_{21}(a, c))) = l_1 \nearrow (l_1 \searrow (a, f_{21}(a, c))) = a$ by PutGet for l_1.

(5) If $a \in (A \cap (A_2 \backslash A_1))$ and $c \notin C_1$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_2 \searrow (a, c))$. But $l_2 \searrow (a, c) \in C_2$ by the type of l_2, so $l \nearrow (l_2 \searrow (a, c)) = l_2 \nearrow (l_2 \searrow (a, c)) = a$ by PutGet for l_2.

(6) If $a \in (A \cap (A_2 \backslash A_1))$ and $c \in C_1$, then $l \nearrow (l \searrow (a, c)) = l \nearrow (l_2 \searrow (a, f_{12}(a, c)))$. But $l_2 \searrow (a, f_{12}(a, c)) \in C_2$ by the types of f_{12} and l_2, so $l \nearrow (l_2 \searrow (a, f_{12}(a, c))) = l_2 \nearrow (l_2 \searrow (a, f_{12}(a, c))) = a$ by PutGet for l_2.

Hence, $l \in C \overset{\Omega}{\rightleftharpoons} A \cap (A_1 \cup A_2)$. Next we prove that l is total at that type if l_1 and l_2 are total, by showing that its *get* and *putback* functions are totally defined on their domains.

We first show that the *get* function is totally defined on C. Pick $c \in C$. If $c \in C_1$ then $l\nearrow c = l_1\nearrow c$. As $l_1 \in C \cap C_1 \overset{\Omega}{\Longleftrightarrow} A \cap A_1$, it follows that $l_1\nearrow c$ is defined. Similarly, if $c \in (C \setminus C_1)$, then $l\nearrow c = l_2\nearrow c$. As $l_2 \in C \setminus C_1 \overset{\Omega}{\Longleftrightarrow} A \cap A_2$, it follows that $l_2\nearrow c$ is defined. Hence, $l\nearrow$ is a total function.

Second, we prove that the *putback* function is totally defined on $(A \cap (A_1 \cup A_2)) \times C_\Omega$. There are six cases, corresponding to the six cases in the definition of the *putback* function:

(1) If $a \in (A \cap (A_1 \cap A_2))$ and $c \in C_1$, then $l\searrow(a,\, c) = l_1\searrow(a,\, c)$ is defined as $l_1\searrow$ is total on $(A \cap A_1) \times (C \cap C_1)_\Omega$.

(2) If $a \in (A \cap (A_1 \cap A_2))$ and $c \notin C_1$, then $l\searrow(a,\, c) = l_2\searrow(a,\, c)$ is defined as $l_2\searrow$ is total on $(A \cap A_2) \times (C \setminus C_1)_\Omega$.

(3) If $a \in (A \cap (A_1 \setminus A_2))$ and $c \in (C_1)_\Omega$, then $l\searrow(a,\, c) = l_1\searrow(a,\, c)$ is defined as $l_1\searrow$ is total on $(A \cap A_1) \times (C \cap C_1)_\Omega$.

(4) If $a \in (A \cap (A_1 \setminus A_2))$ and $c \notin (C_1)_\Omega$, then $l\searrow(a,\, c) = l_1\searrow(a,\, f_{21}(c))$ is defined as f_{21} is a totally defined function with type: $(C \setminus C_1) \to (C \cap C_1)_\Omega$ and $l_1\searrow$ is total on $(A \cap A_1) \times (C \cap C_1)_\Omega$.

(5) If $a \in (A \cap (A_2 \setminus A_1))$ and $c \notin C_1$, then then $l\searrow(a,\, c) = l_2\searrow(a,\, c)$ is defined as $l_2\searrow$ is total on $(A \cap A_2) \times (C \setminus C_1)_\Omega$.

(6) If $a \in (A \cap (A_2 \setminus A_1))$ and $c \in C_1$, then $l\searrow(a,\, c) = l_2\searrow(a,\, f_{12}(c))$ is defined as f_{12} is a totally defined function with type: $(C \cap C_1) \to (C \setminus C_1)_\Omega$ and $l_2\searrow$ is total on $(A \cap A_2) \times (C \setminus C_1)_\Omega$.

Hence, $l\searrow$ is a total function.

We conclude that $(\texttt{cond}\ C_1\ A_1\ A_2\ f_{21}\ f_{12}\ l_1\ l_2) \in C \overset{\Omega}{\Longleftrightarrow} (A \cap (A_1 \cup A_2))$. □

The next lemma record types for conditional lenses in special cases where the conditional *always* selects one lens or the other (in both directions). In these situations, we can use a more flexible typing rule that makes no assumptions about the branch that is never used. The first describes `ccond` instances where the second branch is always taken.

A.29 Lemma [Always-False `ccond`]:
$\forall C, C_1, A \subseteq \mathcal{V}.$ *with* $C \cap C_1 = \emptyset . \forall l_2 \in C \setminus C_1 \overset{\Omega}{\Longleftrightarrow} A.\ \texttt{ccond}\ C_1\ l_1\ l_2 \in C \overset{\Omega}{\Longleftrightarrow} A.$

Proof. First we argue that $(\texttt{ccond}\ C_1\ l_1\ l_2) = l_2$ by showing that their respective *get* and *putback* functions are identical. For any $c \in C$, we must have $c \notin (C_1 \cap C)$ (because it is empty) and so $c \in (C \setminus C_1)$. Hence, $(\texttt{ccond}\ C_1\ l_1\ l_2)\nearrow c = l_2\nearrow c$. Similarly, for any (a, c) in $A \times C_\Omega$, we must have $c \notin (C \cap C_1)$. By definition, $(\texttt{ccond}\ C_1\ l_1\ l_2)\searrow(a,\, c) = l_2\searrow(a,\, c)$.

Since $(\texttt{ccond}\ C_1\ l_1\ l_2) = l_2$, the well-behavedness and totality of the `ccond` lens follow from the well-behavedness and totality of l_2. In particular, since l_1 is never used, we do not need any assumptions about it. □

Note that there is no corresponding *always-true* rule for `ccond`. Even if $C \setminus C_1 = \emptyset$, in the *putback* direction, the Ω tree still gets sent through l_2.

7.13 Lemma [Well-behavedness]:
$\forall D, E \subseteq \mathcal{T}$. *with* $D \cap E = \emptyset$ *and* $D \neq \emptyset$ *and* $E \neq \emptyset$.
$\texttt{inner\_filter}\ D\ E \in [D^{1..\omega}] \& [E] \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}]$
$\texttt{list\_filter}\ D\ E \in [D] \& [E] \stackrel{\Omega}{\rightleftharpoons} [D]$

PROOF. To start, note that ($\texttt{inner\_filter}\ D\ E$) is the fixed point of the following function f from lenses to lenses:

$$\begin{array}{rl} f\ =\ \lambda l.\, \texttt{ccond}\ E :: ([D^{1..\omega}] \& [E]) & \\ \quad (\texttt{tl}\ any_E;\ l) & \\ \quad (\texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, & \\ \qquad \texttt{*t} \mapsto (\texttt{cond}\ [E]\ []\ [D^{1..\omega}]\ fltr_E\ (\lambda c.\ c\texttt{++}[any_D]) & \\ \qquad\quad (\texttt{const}\ []\ []) & \\ \qquad\quad l)\}) & \end{array}$$

To shorten the proof, we sometimes abbreviate the entire `cond` instance as k.

We prove the type for `inner_filter` using Corollary 3.17(1). We assume that $l \in ([D^{1..\omega}] \& [E]) \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}]$ and show that $f(l)$ also has type $([D^{1..\omega}] \& [E]) \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}]$.

The outermost lens is a `ccond` combinator. We must show that each branch has the correct type.

$$\begin{array}{rl} & (\texttt{tl}\ any_E;\ l) \\ & \in\ ([D^{1..\omega}] \& [E]) \cap (E :: ([D^{1..\omega}] \& [E])) \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}] \\ \text{i.e.,} & \qquad\qquad E :: ([D^{1..\omega}] \& [E]) \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}] \end{array}$$

$$\begin{array}{rl} & \texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, \texttt{*t} \mapsto k\} \\ & \in\ ([D^{1..\omega}] \& [E]) \setminus (E :: ([D^{1..\omega}] \& [E])) \stackrel{\Omega}{\rightleftharpoons} [D^{1..\omega}] \\ \text{i.e.,} & \qquad\qquad D :: ([D] \& [E]) \stackrel{\Omega}{\rightleftharpoons} D :: [D] \end{array}$$

The first fact follows from the type of `tl` with $any_E \in E$, the composition operator, and the hypothesis about the type of l. To prove the second, we use the type of `wmap`. Let m be the total function from names to lenses that maps `*h` to `id`, `*t` to k, and every other name to `id`. We show that $m \in \Pi n \in \mathcal{N}. D :: ([D] \& [E])(n) \stackrel{\Omega}{\rightleftharpoons} D :: [D](n)$ as follows:

$$\begin{array}{rl} & m(\texttt{*h}) = \texttt{id} \in\ D :: ([D] \& [E])(\texttt{*h}) \stackrel{\Omega}{\rightleftharpoons} D :: [D](\texttt{*h}) \\ \text{i.e.,} & \qquad\qquad D \stackrel{\Omega}{\rightleftharpoons} D \end{array}$$

by the type of `id`;

$$\begin{array}{rl} & m(\texttt{*t}) = k \in\ D :: ([D] \& [E])(\texttt{*t}) \stackrel{\Omega}{\rightleftharpoons} D :: [D](\texttt{*t}) \\ \text{i.e.,} & \qquad\qquad [D] \& [E] \stackrel{\Omega}{\rightleftharpoons} [D] \end{array}$$

by the argument below;

$$\begin{array}{rl} & m(n) = \texttt{id} \in\ \ D :: ([D] \& [E])(n) \stackrel{\Omega}{\rightleftharpoons} D :: [D](n) \\ \text{i.e.,} & \qquad\qquad \emptyset \stackrel{\Omega}{\rightleftharpoons} \emptyset \end{array}$$

vacuously.

For the tail tag, we must show that k, the `cond` lens, has the lens type $[D] \& [E] \stackrel{\Omega}{\rightleftharpoons} [D]$. The concrete predicate and abstract predicates for the conditional are $C_1 =$

$\texttt{[}E\texttt{]}$, $A_1 = \texttt{[]}$, and $A_2 = \texttt{[}D^{1..\omega}\texttt{]}$. For the first branch, we have that

$$\begin{array}{rr} \texttt{const [] []} \in & \texttt{[}D\texttt{]\&[}E\texttt{]} \cap C_1 \overset{\Omega}{\rightleftharpoons} A_1 \\ \text{i.e.,} & (\texttt{[}D\texttt{]\&[}E\texttt{]}) \cap \texttt{[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[]} \\ \text{i.e.,} & \texttt{[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[]} \end{array}$$

from the type of `const`. For the second, we have that

$$\begin{array}{rr} l \in & \texttt{[}D\texttt{]\&[}E\texttt{]} \setminus C_1 \overset{\Omega}{\rightleftharpoons} A_2 \\ \text{i.e.,} & (\texttt{[}D\texttt{]\&[}E\texttt{]}) \setminus \texttt{[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D^{1..\omega}\texttt{]} \\ \text{i.e.,} & \texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D^{1..\omega}\texttt{]} \end{array}$$

by hypothesis. Next we check that the functions fltr_E and $(\lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]})$ have the correct types:

$$\begin{array}{rl} \mathit{fltr}_E & \in\ (\texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]}) \to (\texttt{[}E\texttt{]})_\Omega \\ \lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]} & \in\ (\texttt{[}E\texttt{]}) \to (\texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]})_\Omega \end{array}$$

Both facts are immediate. Thus, by the type of `cond` we have $m(\texttt{*t}) = k \in \texttt{[}D\texttt{]\&[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}$. Additionally since $\mathsf{doms}(D :: (\texttt{[}D\texttt{]\&[}E\texttt{]})) = \{\{\texttt{*h},\ \texttt{*t}\}\} = \mathsf{doms}(\texttt{[}D^{1..\omega}\texttt{]})$, with Lemma 7.2(1) we have that both types are shuffle closed and have equal sets of domains. Putting all these facts together, we have that the `wmap` instance has type $D :: (\texttt{[}D\texttt{]\&[}E\texttt{]}) \overset{\Omega}{\rightleftharpoons} D :: \texttt{[}D\texttt{]}$ as required. Finally, using the type of `ccond`, we conclude that $f(l) \in (\texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]}) \overset{\Omega}{\rightleftharpoons} \texttt{[}D^{1..\omega}\texttt{]}$. By Corollary 3.17(1) we have $\mathit{fix}(f) = \texttt{inner\_filter}\ D\ E$ has the same type.

The proof that $\texttt{list\_filter}\ D\ E \in \texttt{[}D\texttt{]\&[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}$ is identical to the proof above, that $k \in \texttt{[}D\texttt{]\&[}E\texttt{]} \overset{\Omega}{\rightleftharpoons} \texttt{[}D\texttt{]}$, except that we use the type of `inner_filter` directly rather than our hypothesis about the type of l. □

7.14 Lemma [Totality]:
$\forall D, E \subseteq \mathcal{T}$. with $D \cap E = \emptyset$ and $D \neq \emptyset$ and $E \neq \emptyset$.

$$\texttt{inner\_filter}\ D\ E \in \texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..\omega}\texttt{]}$$
$$\texttt{list\_filter}\ D\ E \in \texttt{[}D\texttt{]\&[}E\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D\texttt{]}$$

Proof. Note that $\texttt{inner\_filter}\ D\ E$ is the fixed point of the same function f defined in the well-behavedness proof.

In outline, the proof goes as follows. We start by choosing a sequence of total type sets $\mathbb{T}_0, \mathbb{T}_1, \ldots$. (Recall that each $\mathbb{T}_i$ is a set of total types and a total type is itself a pair (C, A).) Next, we prove a key property of f: that, when we apply it to a lens possessing all the types in some $\mathbb{T}_i$, the result is a lens possessing all the types in $\mathbb{T}_{i+1}$. Next we choose an increasing instance of the sequence—i.e., a chain $\tau_0 \subseteq \tau_1 \subseteq \ldots$ where each $\tau_i \in \mathbb{T}_i$. We argue that the limit of this increasing instance, $\bigcup_i \tau_i$, is the total type we want—i.e.,

$$(\texttt{[}D^{1..\omega}\texttt{]\&[}E\texttt{]}, \texttt{[}D^{1..\omega}\texttt{]}).$$

We conclude by Lemma 3.19 that the fixed point of f—i.e., the lens $\texttt{inner\_filter}\ D\ E$—has this type, finishing the proof. We now proceed to the details.

We first define the sequence of pairs of total type sets:

$$\begin{array}{rcl} \mathbb{T}_0 & = & \{(\emptyset, \emptyset)\} \\ \mathbb{T}_{i+1} & = & \{(\texttt{[}D^{1..x}\texttt{]\&[}E^{0..y}\texttt{]}, \texttt{[}D^{1..x}\texttt{]}) \mid x + y = i\} \end{array}$$

Let us calculate the first few elements of this sequence explicitly:

$$\begin{array}{rcl}\mathbb{T}_1 &=& \{(\emptyset,\emptyset)\}\\ \mathbb{T}_2 &=& \{(\texttt{[}D^{1..1}\texttt{]},\texttt{[}D^{1..1}\texttt{]})\}\\ \mathbb{T}_3 &=& \{(\texttt{[}D^{1..2}\texttt{]},\texttt{[}D^{1..2}\texttt{]}),(\texttt{[}D^{1..1}\texttt{]}\&\texttt{[}E^{0..1}\texttt{]},\texttt{[}D^{1..1}\texttt{]})\}\end{array}$$

In the proof, we use some abbreviations to lighten the presentation. We abbreviate the type argument to the `ccond` lens as: $C_1 = E :: (\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]})$ and the type arguments to k, the `cond` lens as: $C_1' = \texttt{[}E\texttt{]}$, $A_1' = \texttt{[]}$, and $A_2' = \texttt{[}D^{1..\omega}\texttt{]}$. In each case of the inductive proof below, we introduce local definitions of the source and target type for the typing we are trying to establish as C and A.

We now prove, by induction on i the fact about f needed to apply Lemma 3.19: that if l has every total type in $\mathbb{T}_i$, then $f(l)$ has every total type in $\mathbb{T}_{i+1}$.

For the base case ($i = 0$), we must first show that $f(l)$ has every total type in the singleton set $\mathbb{T}_1 = \{(\emptyset,\emptyset)\}$. This is immediate, since every lens has type $\emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset$.

For the induction step ($i > 0$), we prove that $f(l)$ has every total type in $\mathbb{T}_{i+1}$, assuming that l has every total type in $\mathbb{T}_i$. Pick an arbitrary total type τ from $\mathbb{T}_{i+1}$. We analyze three cases.

Case $x = 0$: Recall that the set $\mathbb{T}_{i+1}$ is $\{(\texttt{[}D^{1..x}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]},\texttt{[}D^{1..x}\texttt{]}) \mid x + y = i\}$. The only element τ in this set with $x = 0$ is the empty total type:

$$(\texttt{[}D^{1..0}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]},\texttt{[}D^{1..0}\texttt{]}) = (\emptyset\&\texttt{[}E^{0..y}\texttt{]},\emptyset) = (\emptyset,\emptyset).$$

Immediately, the lens $f(l)$ has type $\emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset$, finishing the case.

Case $x > 0$ *and* $y = 0$: By construction, τ is of the form (C, A) with $C = \texttt{[}D^{1..x}\texttt{]}$ and $A = \texttt{[}D^{1..x}\texttt{]}$. To verify the type of the `ccond`, we first observe that $C \cap C_1 = \texttt{[}D^{1..x}\texttt{]} \cap E :: (\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]}) = \emptyset$. As a result, the `ccond` always selects the second branch in both the *get* and *putback* directions. By then *always-false* typing for `ccond`, given in Lemma A.29, it suffices to show that the second branch has type $C \overset{\Omega}{\Longleftrightarrow} A$:

$$\begin{array}{lrl}\texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, \texttt{*t} \mapsto k\} \in & C &\overset{\Omega}{\Longleftrightarrow} A\\ \text{i.e.,} & \texttt{[}D^{1..x}\texttt{]} &\overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..x}\texttt{]}\\ \text{i.e.,} & D :: \texttt{[}D^{0..x-1}\texttt{]} &\overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}\end{array}$$

Let m be the total function from names to lenses that maps `*h` to `id`, `*t` to k, and every other name to `id`. We show that $m \in \Pi n \in \mathcal{N}.D :: (\texttt{[}D\texttt{]}\&\texttt{[}E\texttt{]})(n) \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D\texttt{]}(n)$ as follows:

$$\begin{array}{lrl}& m(\texttt{*h}) = \texttt{id} \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]})(\texttt{*h}) &\overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(\texttt{*h})\\ \text{i.e.,} & D &\overset{\Omega}{\Longleftrightarrow} D\end{array}$$

by the type of `id`;

$$\begin{array}{lrl}& m(\texttt{*t}) = k \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]})(\texttt{*t}) &\overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(\texttt{*t})\\ \text{i.e.,} & \texttt{[}D^{0..x-1}\texttt{]} &\overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{0..x-1}\texttt{]}\end{array}$$

by the argument below;

$$\begin{array}{lrll}& m(n) = \texttt{id} \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]})(n) &\overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(n) & \forall n \notin \{\texttt{*h}, \texttt{*t}\}\\ \text{i.e.,} & \emptyset &\overset{\Omega}{\Longleftrightarrow} \emptyset &\end{array}$$

vacuously.

For the tail tag, we must show that the conditional lens, $k \in \texttt{[}D^{0..x-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{0..x-1}\texttt{]}$.

We use the extended typing for `cond` given by Lemma A.28. The standard type for `cond` requires a typing for the branches where the abstract component of each lens type is the corresponding abstract predicate supplied to `cond`. For example, using the standard typing, we would have to show that the second branch, l, has type $\texttt{[}D^{1..x-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..\omega}\texttt{]}$. Unfortunately our induction hypothesis only gives lens types for l where the length of the abstract list is bounded by $x-1$. Notice however, that since we only want to show that the whole `cond` instance has a lens type where the length of the abstract list is at most $x-1$, the lens type that standard type for `cond` requires for l is certainly stronger than we actually need. We avoid this problem using Lemma A.28, which only requires a typing for each branch where the abstract component is calculated from the *intersection* of the abstract predicates supplied to `cond` and the abstract type we wish to show for the whole lens. In this instance, since the `cond` lens has concrete predicate $C_1' = \texttt{[}E\texttt{]}$ and abstract predicates $A_1' = \texttt{[]}$ and $A_2' = \texttt{[}D^{1..\omega}\texttt{]}$, we must prove

$$\begin{array}{rrl} & \texttt{const [] []} \in \quad \texttt{[}D^{0..x-1}\texttt{]} \cap C_1' & \overset{\Omega}{\Longleftrightarrow} A_1' \cap \texttt{[}D^{0..x-1}\texttt{]} \\ \text{i.e.,} & \texttt{[}D^{0..x-1}\texttt{]} \cap \texttt{[}E\texttt{]} & \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \cap \texttt{[}D^{0..x-1}\texttt{]} \\ \text{i.e.,} & \texttt{[]} & \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \end{array}$$

and

$$\begin{array}{rrl} & l \in \quad \texttt{[}D^{0..x-1}\texttt{]} \setminus C_1' & \overset{\Omega}{\Longleftrightarrow} A_2' \cap \texttt{[}D^{0..x-1}\texttt{]} \\ \text{i.e.,} & \texttt{[}D^{0..x-1}\texttt{]} \setminus \texttt{[}E\texttt{]} & \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..\omega}\texttt{]} \cap \texttt{[}D^{0..x-1}\texttt{]} \\ \text{i.e.,} & \texttt{[}D^{1..x-1}\texttt{]} & \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..x-1}\texttt{]}. \end{array}$$

The first fact follows from the type of `const`; the second is immediate by induction hypothesis. We must also show that the functions fltr_E and $(\lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]})$ have the correct types:

$$\begin{array}{rcl} \mathit{fltr}_E & \in & (\texttt{[}D^{1..x-1}\texttt{]}) \to (\texttt{[]})_\Omega \\ \lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]} & \in & (\texttt{[]}) \to (\texttt{[}D^{1..x-1}\texttt{]})_\Omega \end{array}$$

Thus, $k \in \texttt{[}D^{0..x-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{0..x-1}\texttt{]}$.

By the type of `wmap`, together $x > 0$ and Lemma 7.2(1), which states that $D :: \texttt{[}D^{0..x-1}\texttt{]}$ is shuffle closed, we have

$$\texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, \texttt{*t} \mapsto k\} \in D :: \texttt{[}D^{0..x-1}\texttt{]} \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}.$$

Finally, using the *always-false* type of `ccond`, we conclude that $f(l) \in C \overset{\Omega}{\Longleftrightarrow} A$, finishing the case.

Case $x > 0$ and $y > 0$: Here τ has the form (C, A) with $C = (\texttt{[}D^{1..x}\texttt{]} \& \texttt{[}E^{0..y}\texttt{]})$ and $A = \texttt{[}D^{1..x}\texttt{]}$. The outermost lens in $f(l)$ is a `ccond` lens. The typing rule for `ccond` requires that we prove that the branches have the following types:

$$\begin{array}{rrl} & (\texttt{tl}\ \mathit{any}_E;\ l) \in & C \cap C_1 \overset{\Omega}{\Longleftrightarrow} A \\ \text{i.e.,} & (\texttt{[}D^{1..x}\texttt{]} \& \texttt{[}E^{0..y}\texttt{]}) \cap (E :: (\texttt{[}D^{1..\omega}\texttt{]} \& \texttt{[}E\texttt{]})) & \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..x}\texttt{]} \\ \text{i.e.,} & E :: (\texttt{[}D^{1..x}\texttt{]} \& \texttt{[}E^{0..y-1}\texttt{]}) & \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..x}\texttt{]} \end{array}$$

which follows from $y > 0$ using the type of `tl` and the induction hypothesis.

For the second branch we must show

$$\begin{array}{rr} & \texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, \texttt{*t} \mapsto k\} \\ & \in \\ \text{i.e.,} & (\texttt{[}D^{1..x}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \setminus E :: (\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]}) \\ \text{i.e.,} & D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \end{array} \begin{array}{l} C \setminus C_1 \overset{\Omega}{\Longleftrightarrow} A \\ \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..x}\texttt{]} \\ \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}. \end{array}$$

We use the type of `wmap`, together with the facts that $x > 0$, Lemma 7.2, which implies that $D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})$ and $D :: \texttt{[}D^{0..x-1}\texttt{]}$, are shuffle closed, and that the set of domains of trees in two cons cell types are identical.

Let m be the same total function from names to lenses as in the previous case. We prove that $m \in \Pi n \in \mathcal{N}. D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})(n) \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(n)$ as follows:

$$\begin{array}{rr} & m(\texttt{*h}) = \texttt{id} \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})(\texttt{*h}) \\ \text{i.e.,} & \end{array} \begin{array}{l} \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(\texttt{*h}) \\ D \overset{\Omega}{\Longleftrightarrow} D \end{array}$$

by the type of `id`;

$$\begin{array}{rr} & m(\texttt{*t}) = k \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})(\texttt{*t}) \\ \text{i.e.,} & \end{array} \begin{array}{l} \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(\texttt{*t}) \\ \texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{0..x-1}\texttt{]} \end{array}$$

by the argument below;

$$\begin{array}{rr} & m(n) = \texttt{id} \in\ D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})(n) \\ \text{i.e.,} & \end{array} \begin{array}{l} \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}(n) \quad \forall n \notin \{\texttt{*h}, \texttt{*t}\} \\ \emptyset \overset{\Omega}{\Longleftrightarrow} \emptyset \end{array}$$

vacuously.

For the tail tag, we must show that the conditional lens, k has type $\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{0..x-1}\texttt{]}$. Again we use the extended typing for `cond` given by Lemma A.28. We must prove

$$\begin{array}{rr} & \texttt{const [] []} \in \\ \text{i.e.,} & \\ \text{i.e.,} & \end{array} \begin{array}{l} (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \cap C'_1 \overset{\Omega}{\Longleftrightarrow} A'_1 \cap (\texttt{[}D^{0..x-1}\texttt{]}) \\ (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \cap \texttt{[}E\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \cap (\texttt{[}D^{0..x-1}\texttt{]}) \\ \texttt{[}E^{0..y}\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[]} \end{array}$$

and

$$\begin{array}{rr} & l \in \\ \text{i.e.,} & \\ \text{i.e.,} & \end{array} \begin{array}{l} (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \setminus C'_1 \overset{\Omega}{\Longleftrightarrow} A'_2 \cap \texttt{[}D^{0..x-1}\texttt{]} \\ (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \setminus \texttt{[}E\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..\omega}\texttt{]} \cap \texttt{[}D^{0..x-1}\texttt{]} \\ (\texttt{[}D^{1..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \overset{\Omega}{\Longleftrightarrow} \texttt{[}D^{1..\omega}\texttt{]}. \end{array}$$

The first fact follows from the type of `const`; the second is immediate by induction hypothesis. We must also show that the functions fltr_E and $(\lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]})$ have the correct types:

$$\begin{array}{rl} \mathit{fltr}_E & \in\ (\texttt{[}D^{1..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \to (\texttt{[}E^{0..y}\texttt{]})_\Omega \\ \lambda c.\ c\texttt{++[}\mathit{any}_D\texttt{]} & \in\ (\texttt{[}E^{0..y}\texttt{]}) \to (\texttt{[}D^{1..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]})_\Omega \end{array}$$

Both typings are immediate. Putting all these facts together, we have

$$\texttt{wmap}\ \{\texttt{*h} \mapsto \texttt{id}, \texttt{*t} \mapsto k\} \in D :: (\texttt{[}D^{0..x-1}\texttt{]}\&\texttt{[}E^{0..y}\texttt{]}) \overset{\Omega}{\Longleftrightarrow} D :: \texttt{[}D^{0..x-1}\texttt{]}.$$

Finally, using the type of `ccond`, we conclude that $f(l) \in C \overset{\Omega}{\Longleftrightarrow} A$, finishing the case and the inductive proof.

To conclude using Lemma 3.19, we must show that the $(\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]}, \texttt{[}D^{1..\omega}\texttt{]})$ is the limit of an increasing instance of elements of $\overline{\mathbb{T}}$. Let $\tau_0 \subseteq \tau_1 \subseteq \ldots$ be defined

as

$$\begin{array}{rcl} \tau_0 & = & (\emptyset, \emptyset) \qquad\qquad\qquad\qquad\qquad\qquad \in \mathbb{T}_0 \\ \vdots & & \\ \tau_{i+1} & = & (\texttt{[}D^{1..((i+1)/2)}\texttt{]}\&\texttt{[}E^{0..(i/2)}\texttt{]}, \texttt{[}D^{1..((i+1)/2)}\texttt{]}) \in \mathbb{T}_{i+1} \end{array}$$

where i/n is integer division of i by n. To show that the limit is the pair of total types we want, we prove that each set is contained in the other. First, observe that, for any $c \in (\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]})$ and $a \in \texttt{[}D^{1..\omega}\texttt{]}$, we can find an i such that $(c, a) \in \tau_i$ (lifting $\in$ to pairs of sets in the obvious way) by choosing i so that $i/2$ is greater than the maximum number of elements of D in c, the number of elements of E in c, and the number of elements in a. The other inclusion is immediate: every τ_i is a subset of $(\texttt{[}D^{1..\omega}\texttt{]}\&\texttt{[}E\texttt{]}, \texttt{[}D^{1..\omega}\texttt{]})$ (lifting $\subseteq$ to pairs of pairs of sets twice, pointwise). □

The proof that `list_filter` $D\ E \in \texttt{[}D\texttt{]}\&\texttt{[}E\texttt{]} \overset{\Omega}{\Longleftrightarrow} \texttt{[}D\texttt{]}$ is identical to its proof of well-behavedness, except that we use the total type of `inner_filter`.