Appendix
– Program Logic and Program Analysis –

Kohei Honda Nobuko Yoshida Martin Berger

Contents

A Reduction Rules

This section lists the reduction rules of the languages defined in Sections 2 and 4.

A.1 Global State

We begin with reduction rules for values in the system with global state.

$$\frac{-}{(\lambda x.M)V \to M[V/x]} \qquad \frac{j \in \{1,2\}}{\pi_j(\langle V_1, V_2\rangle) \to V_j}$$

$$\frac{V = \mathtt{inj}_j(W) \quad \text{for some } j \in \{1,2\}}{\mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i).M_i\}_{i\in\{1,2\}} \to M_j[W/x_j]}$$

$$\frac{-}{\mathtt{if\ t\ then}\ M_1\ \mathtt{else}\ M_2 \to M_1} \qquad \frac{m \neq n}{m = n \to \mathtt{f}}$$

$$\frac{V = \mu f.\lambda g.N}{VW \to N[W/g][V/f]} \qquad \frac{-}{(\Lambda X.V)\alpha \to V[\alpha/X]}$$

$$\frac{M = \mathtt{pack}\,\langle \beta, V\rangle\,\mathtt{as}\,\exists X.\alpha}{\mathtt{unpack}\ M\ \mathtt{as}\,\langle X, x\rangle\ \mathtt{in}\ N \to N[\beta/X][V/x]}$$

$$\frac{-}{\mathtt{let}\ x = V\ \mathtt{in}\ N \to N[V/x]}$$

Next are the stateful reduction steps.

$$\frac{\sigma(x) = V}{(!x, \sigma) \to (V, \sigma)} \qquad \frac{x \in \mathsf{dom}(\sigma)}{(x := V, \sigma) \to ((), \sigma[x \mapsto V])}$$

Finally we list the rules for the contexts.

$$\frac{M \to M'}{(M, \sigma) \to (M', \sigma')}$$

$$\frac{(M, \sigma) \to (M', \sigma')}{(\mathtt{let}\ x = M\ \mathtt{in}\ N, \sigma) \to (\mathtt{let}\ x = M'\ \mathtt{in}\ N, \sigma')}$$

A.2 Local State

The reduction steps for values are unchanged. Stateful rules are now between configurations $(\nu\varphi)(M,\sigma)$ where $\varphi \subseteq \mathsf{dom}(\sigma)$. We start by defining the *garbage collection congruence* $\equiv_{\mathsf{gc}}$ which is the least typed congruence closed under the following implication.

$$\frac{x \notin \mathsf{fv}(M)}{(\nu\varphi \cup \{x\})(M, \sigma \cdot x \mapsto V) \equiv_{\mathsf{gc}} (\nu\varphi)(M, \sigma)}$$

Now reductions are defined modulo $\equiv_{\mathsf{gc}}$-congruence classes of configurations. We have one genuinely new rule

$$\frac{x \notin \varphi \cup \mathsf{dom}(\sigma)}{(\nu\varphi)(\mathtt{new\ var}\ x := V\ \mathtt{in}\ M, \sigma) \to (\nu x\varphi)(M, \sigma \cdot x \mapsto V)}$$

The previous subsection's reduction rules are recycled, but with added contextual rules for the local names.

$$\frac{(\nu\varphi_1)(M_1,\sigma_1) \equiv_{\mathsf{gc}} (\nu\varphi_1')(M_1',\sigma_1') \quad (\nu\varphi_1')(M_1',\sigma_1') \to (\nu\varphi_2')(M_2',\sigma_2') \quad (\nu\varphi_2')(M_2',\sigma_2') \equiv_{\mathsf{gc}} (\nu\varphi_2)(M_2,\sigma_2)}{(\nu\varphi_1)(M_1,\sigma_1) \to (\nu\varphi_2)(M_2,\sigma_2)}$$

$$\frac{(\nu\varphi)(M,\sigma) \to (\nu\varphi')(M',\sigma')}{(\nu\varphi\cup\varphi_0)(M,\sigma) \to (\nu\varphi'\cup\varphi_0)(M',\sigma')}$$

$$\frac{(\nu\varphi)(M,\sigma) \to (\nu\varphi')(M',\sigma') \quad \varphi\cap\mathsf{fv}(N)=\emptyset}{(\nu\varphi)(\mathtt{let}\ x=M\ \mathtt{in}\ N,\sigma) \to (\nu\varphi')(\mathtt{let}\ x=M'\ \mathtt{in}\ N,\sigma')}$$

Lemma 1 *1. If $(\nu\vec{x}y)(M,\sigma) \to (\nu\vec{z})(M',\sigma')$ then also $(\nu\vec{x})(M,\sigma) \to (\nu\vec{z}\setminus y)(M',\sigma')$.*

2. If $(\nu\vec{x}y)(M,\sigma) \Downarrow (\nu\vec{z})(V,\sigma')$ then also $(\nu\vec{x})(M,\sigma) \to (\nu\vec{z}\setminus y)(V,\sigma')$.

3. Assume y is of non-reference type. Then

$$(\nu\vec{x})(M,\sigma) \to (\nu\vec{x})(N,\sigma') \quad \textit{implies} \quad (\nu\vec{x})(M[V/y],\sigma) \to (\nu\vec{x})(N[V/y],\sigma')$$

4.

$$(\nu\vec{x})(M,\sigma) \to (\nu\vec{x})(N,\sigma') \quad \textit{implies} \quad (\nu\vec{x})(M[\alpha/\mathrm{X}],\sigma) \to (\nu\vec{x})(N[\alpha/\mathrm{X}],\sigma')$$

B Typed Logical Terms and Typed Assertions

This section gives the detailed definitions of typed assertions and formulae in Sections 3 and 4.

- $()$, $\mathbf{n}$, $\mathbf{b}$ and x^ρ are well-typed with types Unit, Nat, Bool and ρ, respectively.
- $\neg e$ is well-typed with type Bool, if e is well-typed with type Bool. e_1+e_2 is well-typed with type Nat if $e_{1,2}$ are well-typed with type Nat. Similarly for $\mathtt{succ}$, $\times$, $-$, etc.
- $!(x^{\mathsf{Ref}(\alpha)})$ is well-typed with type α.
- $\langle e_1,e_2\rangle$ is well-typed with type $\alpha_1\times\alpha_2$ if $e_{1,2}$ are well-typed with types $\alpha_{1,2}$. $\pi_1(e)$ is well-typed with type α if e is well-typed with type $\alpha\times\beta$, similarly for $\pi_2(e)$.
- $\mathsf{inj}_1^{\alpha+\beta}(e)$ is well-typed with type $\alpha+\beta$ if e is well-typed with type α. Similarly for $\mathsf{inj}_2^{\alpha+\beta}(e)$.
- $e\bullet[\beta]$ is well-typed with type $\alpha[\beta/\mathrm{X}]$ if e is well-typed with type $\forall\mathrm{X}.\alpha$. $\langle\beta,e\rangle^{\exists\mathrm{X}.\alpha}$ is well-typed with type $\exists\mathrm{X}.\alpha$ if e is well-typed with type $\alpha[\beta/\mathrm{X}]$.

We write $\Gamma\vdash e:\rho$ if, under the environment Γ, e has type α. Next we type formulae, or rather we define $\Gamma\vdash A$, which says A is well-formed under the environment Γ.

- If $\Gamma\vdash e_i:\rho$ for $i=1,2$, then $\Gamma\vdash e_1=e_2$.
- If $\Gamma\vdash A$ then also $\Gamma\vdash\neg A$. Similarly, $\Gamma\vdash A_i$ for $i=1,2$, then $\Gamma\vdash A_1\wedge A_2$, etc.
- If $\Gamma,x:\rho\vdash A$ then $\Gamma\vdash\forall x^\rho.A$ and $\Gamma\vdash\exists x^\rho.A$.
- If $\Gamma\vdash A$ and $\mathrm{X}\notin\mathsf{ftv}(\Gamma)$, then $\Gamma\vdash\forall\mathrm{X}.A$ and $\Gamma\vdash\exists\mathrm{X}.A$.
- Suppose $\Gamma\vdash e_1:\alpha\stackrel{\varphi}{\Rightarrow}\beta$ and $\Gamma\vdash e_2:\alpha$. Suppose also $\Gamma\vdash C$ and $\Gamma,z:\beta\vdash C'$. Then $\Gamma\vdash\{C\}\ e_1\bullet e_2\searrow z\ \{C'\}$.
- Suppose $\Gamma,\vec{x}:\vec{\rho}\vdash C$ and $\Gamma\vdash y_i:\rho_i'$. Suppose also $\Gamma=\Delta,\vec{y}:\vec{\rho}'$ and $x_j\notin\mathsf{fv}(\Delta)$. Then $\Gamma/\vec{x}\vdash(\nu\vec{x}^{\vec{\rho}})\vec{y}$ in C.

We will write $\Gamma;\Theta\vdash C$ to indicate that C is typable under Γ,Θ, with all the primary name being in the domain of Γ while all auxiliaries can be found in Θ. We write

$$A^{\Gamma,\Theta}\ \vdash M^{\Gamma;\rho;\varphi}:_v B^{v:\rho,\Theta}$$

provided Γ and Θ are two environments with disjoint domains such that $\Gamma,\Theta\vdash A$, $\Gamma,\Theta\vdash M:\rho,\varphi$ and $v:\rho,\Theta\vdash B$. Similarly, we write

$$\{C_1^{\Gamma,\Theta}\}\ M^{\Gamma;\alpha}:_u\{C_2^{u:\alpha,\Theta}\}$$

whenever $\Gamma,\Theta\vdash C_1$, $\Gamma,\Theta\vdash M:\alpha$ and $u:\alpha,\Theta\vdash C_2$. Often some or all of the superscripts will be omitted.

C Further Reasoning Examples

Binary trees are at the heart of the next example:

$$\mathsf{tree}_\alpha=\mu\mathrm{X}.(\mathsf{Unit}+(\alpha\times(\mathrm{X}\times\mathrm{X}))) \qquad \mathsf{tree}=\forall\mathrm{X}.\mathsf{tree}_\mathrm{X}$$

The empty tree is represented by $\mathsf{leaf}\stackrel{\mathrm{def}}{=}\mathtt{inj}_1(())$ while the tree carrying the datum a of type α in its root node and having left subtree l and right subtree, r is $\langle l,a,r\rangle\stackrel{\mathrm{def}}{=}\mathtt{inj}_2(\langle a,\langle l,r\rangle\rangle)$. By $\mathsf{tree}_\alpha\approx\mathsf{Unit}+(\alpha\times(\mathsf{tree}_\alpha\times\mathsf{tree}_\alpha))$ this is well-typed. Considering $\mu\mathrm{X}.\mathtt{inj}_2(\langle a,\langle\mathrm{X},\mathrm{X}\rangle\rangle)$ one sees that trees can be infinite. Many tree algorithms are generic and the depth-first tree visitor is a popular example of this. It receives a type α, a function f of type $\alpha\stackrel{\varphi}{\Rightarrow}\mathsf{Unit}$ and a tree of type α. Then f is applied to each node in t in depth-first order. The algorithm df_φ, defined next, is generic in the type of tree-nodes but not in the effect of the function applied to each node. This is because our language does not offer effect polymorphism.

$$\begin{aligned}
\mathsf{df}_{\alpha,\varphi} &= \mu g^{\mathsf{tree}_\alpha\stackrel{\varphi}{\Rightarrow}\mathsf{Unit}}.\lambda f^{\alpha\stackrel{\varphi}{\Rightarrow}\mathsf{Unit}}.\lambda t^{\mathsf{tree}_\alpha}.M\\
M &= \mathtt{case}\ t\ \mathtt{of}\ (\mathsf{leaf}\triangleright\mathtt{skip}\mid\langle l,a,r\rangle\triangleright g(l);f(a);g(r))\\
\mathsf{df}_\varphi &= \Lambda\mathrm{X}.\mathsf{df}_{\mathrm{X},\varphi}
\end{aligned}$$

The case statement is syntactic sugar for

$$\mathtt{case}\ t\ \mathtt{of}\ \{\mathtt{inj}_i(y_i).M_i\}_{i\in\{1,2\}}$$

$$[Leaf]\ \frac{-}{A[\mathsf{leaf}/u]\ \vdash \mathsf{leaf} :_u A}$$

$$[Node]\ \frac{A \vdash V_i :_{u_i} B_i\ (i=1,2,3) \quad (B_1 \wedge B_2 \wedge B_3) \supset B[\langle u_1,u_2,u_3\rangle/u]}{A \vdash \langle V_1,V_2,V_3\rangle :_u B}$$

$$[Case]\ \frac{\{C\}\ V :_v \{A\} \qquad \begin{array}{l}\{C \wedge A[\mathsf{leaf}/v]\}\ M_1 :_u \{C'\}\\ \{C \wedge A[\langle l,a,r\rangle/v]\}\ M_2 :_u \{C'\}\end{array}}{\{C\}\ \mathtt{case}\ V\ \mathtt{of}\ (\mathsf{leaf} \triangleright M_1 \mid \langle l,a,r\rangle \triangleright M_2) :_u \{C'\}}$$

Figure 1: Proof rules for trees.

where y_1,y_2 are fresh, $M_1 = \mathsf{leaf}$ and

$$M_2 = (\lambda l.\lambda a.\lambda r.N)(\pi_1(\pi_2(y_2)))(\pi_1(y_2))(\pi_2(\pi_2(y_2))).$$

Figure 1 proposes three derived rules for reasoning about trees. Next we give a specification for df_φ. We begin by defining a predicate for the number of nodes in the tree.

$$\begin{array}{rcl}\mathsf{size}(t,0) & \equiv & t = \mathsf{leaf}\\ \mathsf{size}(t,n+1) & \equiv & \exists l,a,r,m_l,m_r.t = \langle l,a,r\rangle\\ & & \wedge\ \mathsf{size}(l,m_l)\\ & & \wedge \mathsf{size}(r,m_r) \wedge m_l + m_r = n\\ \mathsf{finite}(t) & \equiv & \exists n.\mathsf{size}(t,n)\end{array}$$

Now we can specify the algorithm. Essentially all we guarantee is that f is applied to eactly all nodes in depth-first order and the effects of these applications are properly chained.

$$\begin{array}{rcl}\mathsf{DF}_{\varphi u} & \equiv & \forall \mathrm{X}.\forall f^{\mathrm{X} \overset{\varphi}{\Rightarrow} \mathsf{Unit}}.\forall t^{\mathrm{X}}.A\\ A & \stackrel{\text{def}}{=} & \mathsf{finite}(t) \supset (\mathsf{DF}^{\mathsf{l}}{}_{\varphi \mathrm{X} uft} \wedge \mathsf{DF}^{\mathsf{n}}{}_{\varphi \mathrm{X} uft})\\ \mathsf{DF}^{\mathsf{l}}{}_{\varphi \mathrm{X} uft} & \equiv & t = \mathsf{leaf} \supset \{C\}\ u \bullet \mathrm{X} \bullet f \bullet t \{C\}\\ \mathsf{DF}^{\mathsf{n}}{}_{\varphi \mathrm{X} uft} & \equiv & \exists l,a,r.(t = \langle l,a,r\rangle \supset B_\varphi(\mathrm{X},u,f,t,l,a,r)\\ B_{\varphi \mathrm{X} uftlar} & \equiv & \left(\begin{array}{c}\{C\}\ u \bullet \mathrm{X} \bullet f \bullet l \{C_1\}\\ \wedge\\ \{C_1\}\ f \bullet a\ \{C_2\}\\ \wedge\\ \{C_2\}\ u \bullet \mathrm{X} \bullet f \bullet r \{C'\}\end{array}\right) \supset \{C\}\ u \bullet \mathrm{X} \bullet f \bullet t \{C'\}\end{array}$$

Now we can assert

$$\mathsf{T} \vdash \mathsf{df}_\varphi :_u \mathsf{DF}_{\varphi u}.$$

Counting Nodes by Depth-First Searching In itself df_φ does not cause effects, apart from what the function applied to each node does. Now we implement a node-counting algorithm that increments a local variable every time a tree-node is being acessed.

$$\begin{array}{rcl}\mathsf{df}' & \stackrel{\text{def}}{=} & \mathtt{new\ var}\ n := 0\ \mathtt{in}\ (M; !n)\\ M & \stackrel{\text{def}}{=} & \mathtt{let}\ counter = \Lambda \mathrm{X}.\lambda x^{\mathrm{X}}.n := 1 + !n\ \mathtt{in}\ N\\ N & \stackrel{\text{def}}{=} & \mathsf{df}_{\{n\}}\ [\mathrm{X}]\ counter\end{array}$$

Then df' we can derive the following judgement.

$$\mathsf{T} \vdash \mathsf{df}' :_u \forall \mathrm{X}.\forall t^{\mathsf{treex}}.\forall n.(\mathsf{size}(t,n) \supset u \bullet [\mathrm{X}] \bullet t = n)$$

Once more we have a purely functional specification for an imperative algorithm, because the non-functional behaviour is not externally visible.

D Models and Soundness

This section first gives the detailed definitions of the models (Section 5), then proves the soundness result (Theorem 1).

D.1 Models

This section gives the detailed definitions for the models. First we introduce a couple of definitions. The set $\mathsf{fen}(\rho)$ of ρ's *free effect names* is inductively defined as follows.

$$\mathsf{fen}(\alpha \overset{\varphi}{\Rightarrow} \beta) = \varphi \qquad \mathsf{fen}(\alpha \star \beta) = \mathsf{fen}(\alpha) \cup \mathsf{fen}(\beta)$$
$$\mathsf{fen}(\mathcal{Q}\mathrm{X}.\alpha) = \mathsf{fen}(\mu \mathrm{X}.\alpha) = \mathsf{fen}(\mathsf{Ref}(\alpha)) = \mathsf{fen}(\alpha)$$

where $\mathcal{Q} \in \{\forall,\exists\}$ $\star$ is a binary operator such as $+$ or $\times$. For all other cases we set $\mathsf{fen}(\alpha) = \emptyset$. Next we define the set $\mathsf{notOpen}(A)$ of names *not open* in A.

$$\mathsf{notOpen}((\nu \vec{x})\vec{y}\ \mathsf{in}\ A) = \{y_i^\alpha \in \vec{y} \mid \mathsf{fen}(\alpha) \cap \{\vec{x}\} = \emptyset\}.$$
$$\mathsf{notOpen}(\mathcal{Q}x.C) = \mathsf{notOpen}(C) \setminus \{x\}.$$

where $\mathcal{Q} \in \{\forall,\exists\}$. For all other cases we proceed homomorphically, e.g. $\mathsf{notOpen}(C \wedge C') = \mathsf{notOpen}(C) \cup \mathsf{notOpen}(C')$ and so forth. Now we set

$$\mathsf{opened}(C) = \mathsf{fv}(C) \setminus \mathsf{notOpen}(C).$$

Models Recall that a program $\Gamma \vdash M : \alpha,\varphi$ is *semi-closed* (resp. *closed*) when $\mathsf{dom}(\Gamma) = \varphi$ (resp. $\mathsf{dom}(\Gamma) = \varphi = \emptyset$). An *open model* is a pair (ξ,σ) where ξ is a finite map from names to semi-closed values and σ is another map of the same kind. Open models are typed as follows.

$$\vec{x} : \vec{\alpha}, \vec{y} : \mathsf{Ref}(\vec{\beta}) \vdash \vec{x} : \vec{V}, \vec{y} \mapsto \vec{W}$$

Here V_i is a value of type α_i and for on. A *model* $(\mathcal{M},\dots)$ is a pair of a finite set of names and an open model, written $(\nu \vec{x})(\xi,\sigma)$ such that $\{\vec{x}\} \subseteq \mathsf{dom}(\sigma)$. The typing $\Gamma \vdash \mathcal{M}$ is naturally defined as follows: if we have

$$\vec{x} : \vec{\alpha}, \vec{y} : \mathsf{Ref}(\vec{\beta}), \vec{z} : \mathsf{Ref}(\vec{\gamma}) \vdash \vec{x} : \vec{V}, \vec{y} \mapsto \vec{W}, \vec{z} \mapsto \vec{U}$$

for an open model, then

$$\vec{x} : \vec{\alpha} \setminus \vec{z}, \vec{y} : \mathsf{Ref}(\vec{\beta}) \vdash (\nu \vec{z})(\vec{x} : \vec{V}, \vec{y} \mapsto \vec{W}, \vec{z} \mapsto \vec{U})$$

Formally we define, given Γ,

$$\frac{\Gamma \vdash x_i : W_i \quad \Gamma \vdash \sigma}{\Gamma \vdash (\vec{x} : \vec{W}, \sigma)} \qquad \frac{\Gamma, x : \mathsf{Ref}(\alpha) \vdash \mathcal{M}}{\Gamma \setminus x \vdash (\nu x)\mathcal{M}}$$

If $\mathcal{M} = (\nu\vec{x})(\xi,\sigma)$ such that $\xi(y) = V$ and $\mathsf{fv}(V)\cap\{\vec{x}\} = \emptyset$, we write $\mathcal{M}(y) = V$ (such y is called *open*). If $\vec{y}$ is open in $\mathcal{M}$, we write $\mathcal{M}\setminus\vec{y}$ for the result of taking off the $\vec{y}$-part from $\mathcal{M}$. $\mathcal{M}_1\cdot\mathcal{M}_2$ is their component-wise union. For example,

$$(\nu\vec{z}_1)(\xi_1,\sigma_1)\cdot(\nu\vec{z}_2)(\xi_2,\sigma_2) = (\nu\vec{z}_1\vec{z}_2)(\xi_1\cup\xi_2,\sigma_1\cup\sigma_2)$$

with $\vec{z}_1\cap\vec{z}_2 = \emptyset$ and

$$(\nu\vec{z})(\xi\cdot x:V,\sigma\cdot y\mapsto W)\setminus xy = (\nu\vec{z})(\xi,\sigma)$$

Satisfaction Let $\Gamma\vdash\mathcal{M}$ be a model and $\Theta\vdash I$ be an open model such that $\mathsf{dom}(\Gamma)\cap\mathsf{dom}(\Theta) = \emptyset$. Then the *interpretation* $[\![e]\!]\mathcal{M}\cdot I$ is well-defined, provided $\Gamma,\Theta\vdash e:\rho$ and is given as:

$$\begin{array}{rcll}
[\![x^{\alpha}]\!]\mathcal{M}\cdot I & \stackrel{\text{def}}{=} & V & (\text{either } \mathcal{M}(x) = V \text{ or } I(x) = V)\\
[\![!x]\!]\mathcal{M}\cdot I & \stackrel{\text{def}}{=} & V & (\text{either } \mathcal{M}(x) = V \text{ or } I(x) = V)\\
[\![x^{\mathsf{Ref}\,(\alpha)}]\!]\mathcal{M}\cdot I & \stackrel{\text{def}}{=} & x & \\
[\![\mathbf{c}]\!]\mathcal{M}\cdot I & = & \mathtt{c} & \\
[\![\mathbf{op}(\vec{e})]\!]\mathcal{M}\cdot I & = & \mathbf{op}([\![\vec{e}]\!]\mathcal{M}\cdot I) & \\
[\![\langle e,e'\rangle]\!]\mathcal{M}\cdot I & = & \langle[\![e]\!]\mathcal{M}\cdot I,[\![e']\!]\mathcal{M}\cdot I\rangle & \\
[\![\pi_i(e)]\!]\mathcal{M}\cdot I & = & \pi_i([\![e]\!]\mathcal{M}\cdot I) & \\
[\![\mathsf{inj}^i_{\alpha+\beta}(e)]\!]\mathcal{M}\cdot I & = & \mathtt{inj}_i([\![e]\!]\mathcal{M}\cdot I) & \\
[\![e\bullet[\beta]]\!]\mathcal{M}\cdot I & = & ([\![e]\!]\mathcal{M}\cdot I)([\![\beta]\!]I) & \\
[\![\langle\beta,e\rangle^{\exists\mathrm{X}.\alpha}]\!]\mathcal{M}\cdot I & = & \mathtt{pack}\,\langle[\![\beta]\!]I,[\![e]\!]\mathcal{M}\cdot I\rangle\,\mathtt{as}\,\exists\mathrm{X}.\alpha &
\end{array}$$

In this definition we also evaluated non-reference types as $[\![\beta]\!]I$. We omit the obvious definition. The satisfaction relation $\mathcal{M}\models^I A$ is defined precisely when Γ,Θ can be found such that $\Gamma\vdash\mathcal{M}$, $\Theta\vdash I$ and $\Gamma,\Theta\vdash C$. Its clauses follow the classical treatments of satisfaction, cf. [5, § 3.3]; the definitions for the additional clauses are found in Section 5 in the main paper. We list all rules for the reference. Logical connectives are given the standard (classical) interpretation (below connectives on the right-hand side are about validity):

$$\begin{array}{rcl}
\mathcal{M}\models^I C_1\wedge C_2 & \equiv & (\mathcal{M}\models^I C_1) \quad\wedge\quad (\mathcal{M}\models^I C_2)\\
\mathcal{M}\models^I\neg C & \equiv & \neg\,(\mathcal{M}\models^I C)\\
\mathcal{M}\models^I\forall x^{\alpha}.C & \equiv & \forall\mathtt{c}\in\alpha.\ \mathcal{M}\models^{I\cdot x:\mathtt{c}} C,\\
\mathcal{M}\models^I\forall x^{\alpha}.C & \equiv & \forall V\in[\![\alpha]\!].\ \mathcal{M}\models^{I\cdot x:V} C,\\
\mathcal{M}\models^I\forall\mathrm{X}.C & \equiv & \forall\alpha.\ \mathcal{M}\models^{I\cdot x:\alpha} C
\end{array}$$

where, in the third line, α is either Nat, Unit or Bool, and in the second last line, $[\![\alpha]\!]$ denotes the set of all closed values of type α. The rest is de Morgan duality. The additional clauses follow:

- $\mathcal{M}\models^I\{C\}e_1\bullet e_2\searrow x\{C'\}$ with e_1 typed $\alpha\stackrel{\varphi}{\Rightarrow}\beta$, is given as, for each appropriately typed σ with $\varphi\subseteq\mathsf{dom}(\sigma)$, $(\mathcal{M}\setminus\varphi)\cdot\sigma\models^I C$ and $(V_1V_2,\sigma)\Downarrow(\nu\vec{y})(V,\sigma')$ imply $(\mathcal{M}\setminus\varphi)\cdot(\nu\vec{y})(x:V,\sigma')\models^I C'$ with $V_i = [\![e_i]\!]\mathcal{M}\cdot I$ $(i = 1,2)$.
- $\mathcal{M}\models^I(\nu\vec{x})\vec{u}$ in A if $\mathcal{M}'\models^I A$ and $\mathcal{M} = (\nu\vec{x})\mathcal{M}'$ where $\vec{u}$ should be open in $\mathcal{M}'$ and $\{\vec{x}\}\cap\mathsf{fn}(\mathcal{M}'/\vec{u}) = \emptyset$, that is $\vec{x}$ are used only by the image of $\vec{u}$.

Finally we set:

$$\mathcal{M}\models^I A \quad\text{if}\quad \mathcal{M}'\models^I A\ \wedge\ \mathcal{M}'\cong\mathcal{M}$$

by which we take the satisfaction relation up to $\cong$.

Semantics of Judgements and Soundness We define the validity as follows. In the following, we write I_a for an interpretation of the auxiliary names and I_t for that of the free type variables. We also abuse ξ and I as the substitution, i.e. we write $M\xi$ for $M[\vec{V}/\vec{x}]$ with $\xi = \vec{x}:\vec{V}$.

Definition 3
- *$A\models M^{\Gamma;\rho}:_u B$ iff, for each $I = I_a\cdot I_t$ and ξ, whenever $\xi\models^I A$, we have $M\xi I_t\Downarrow V$ such that $u:V\models^I B$.*
- *$\models\{C\}M:_u\{C'\}$ iff, for each $\mathcal{M} = (\nu\vec{y})(\xi,\sigma)$ and $I = I_a\cdot I_t$, whenever $\mathcal{M}\models^I C$, we have $(\nu\vec{y})(M\xi I_t,\sigma)\Downarrow(\nu\vec{z})(V,\sigma')$ such that $(\nu\vec{z})(u:V,\sigma')\models^I C'$.*

D.2 Soundness Proofs

In this subsection, we prove:

Theorem 1. (1) $A\vdash M^{\Gamma;\rho}:_u B$ implies $A\models M^{\Gamma;\rho}:_u B$.
(2) $\vdash\{C\}M^{\Gamma;\rho;\varphi}:_u\{C'\}$ implies $\models\{C\}M^{\Gamma;\rho;\varphi}:_u\{C'\}$.

We first start from (1) (assuming (2) holds as the inductive hypothesis). Below an *injective renaming* is an injective function on the set of names. In addition to names, we apply the injective renaming to environments, expressions, formulae, assertions, models and interpretations, usually in postfix form.

Lemma 2 *Let $\Gamma,\Theta\vdash A:\alpha$ and assume given $\Gamma\vdash\mathcal{M}$, $\Theta\vdash I$.*

1. (thinning) *Assume x does not occur in the domains of Γ and Θ. Then*

$$\mathcal{M}\cdot x:V\models^I A \quad\textit{implies}\quad \mathcal{M}\models^I A$$

2. (weakening) *For all $\Gamma'\vdash\mathcal{M}'$ and $\Theta'\vdash I'$ such that $\mathsf{dom}(\Gamma)\cap\mathsf{dom}(\Gamma') = \mathsf{dom}(\Theta)\cap\mathsf{dom}(\Theta') = \emptyset$,*

$$\mathcal{M}\models^I A \quad\textit{implies}\quad \mathcal{M}\cdot\mathcal{M}'\models^{I\cdot I'} A$$

Lemma 3 *1.* (injective renaming) *For all injective renamings ξ,*

$$\mathcal{M}\models^I C \quad\textit{iff}\quad \mathcal{M}\xi\models^{I\xi} C\xi$$

Note that $\mathcal{M}$ on the right of the equivalence is typed under $\Gamma\xi$ and $I\xi$ under $\Theta\xi$.

2. (iso) $C \approx C'$ *and* $\mathcal{M} \models^I C$ *imply* $\mathcal{M} \models^I C'$.

Lemma 4 (substitution) *Assume* $x \notin \mathsf{fv}(e) \cap \mathsf{dom}(\mathcal{M})$.

$$\mathcal{M} \models^I C[e/x] \quad \textit{iff} \quad \mathcal{M} \cdot x : [\![e]\!]\mathcal{M} \cdot I \models^I C$$

Similarly, if $x \notin \mathsf{ftv}(\alpha) \cap \mathsf{dom}(I)$,

$$\mathcal{M} \models^I C[\alpha/\mathrm{X}] \quad \textit{iff} \quad \mathcal{M} \models^{I \cdot \mathrm{X} \mapsto \alpha} C$$

We call a judgement $A_1 \vdash V :_v A_2$ *well-formed* if all primary names x in A_i are (1) of non-reference type and (2) $x \in \mathsf{opened}(A_i)$.

Proofs of Theorem 1 (1) By induction on the derivation of $A \vdash V :_u B$. Since substitution into type variables, I_t, is not significant for most of cases, we only demonstrate its need in [*T-App,T-Pack*]; in other cases, we just need to replace ξ by $\xi \cdot I_t$, hence we omit I_t in the proofs.

We begin with [*Var*]. We prove:

$$A[x/u] \models V :_u A$$

By definition, this is equivalent to prove: for each I and ξ, whenever $\xi \models^I A[x/u]$, we have $x\xi \Downarrow V$ such that $u : V \models^I A$. We define $\xi = \xi', x : W$. Then we know $x\xi = W$. Hence we prove:

$$\xi', x : W \models^I A[x/u] \quad \text{implies} \quad u : W \models^I A$$

First we note that x is the sole primary name in $A[x/u]$. Hence applying the thinning (Lemma 2(1)), we have:

$$\xi', x : W \models^I A[x/u] \text{ implies } x : W \models^I A[x/u]$$

From this, using injective renaming (Lemma 3(1)), we have:

$$(x : W)[u/x] \models^I A[x/u][u/x]$$

which implies $u : W \models^I A$, as required.

Next we prove the case [*Op*]. For simplicity we'll demonstrate the proof using the binary addition function. [*Const*], [*Pair*], [*Proj$_i$*] and [*In$_i$*] are similarly proved. Assume

$$A \vdash V_1 + V_n :_u B \quad \text{and} \quad \xi \models^I A.$$

Since $(V_1 + V_2)\xi \Downarrow (V_1 + V_2)\xi$, we prove

$$u : (V_1 + V_2)\xi \models^I B.$$

By the inductive hypothesis, with $\xi \models^I A$, we know:

$$v_i : V_i\xi \models^I B_i$$

where $\mathsf{fv}(V_1) \cup \mathsf{fv}(V_2) = \{\tilde{x}\}$ and $W_i = \xi(x_i)$. By assumptions. $v_1 \neq v_2$, so with thinning we infer

$$\xi' \models^I B_i$$

where $\xi' = v_1 : V_1\xi, v_2 : V_2\xi$. Thus

$$\xi' \models^I B_1 \wedge B_2$$

Then from the rightmost premise of the rule, $B_1 \wedge B_2 \supset B[v_1 + v_n/u]$, we have:

$$\xi' \models^I B[v_1 + v_2/u]$$

Applying Lemma 4 gives

$$\xi', u : [\![v_1 + v_2]\!]\xi' \cdot I \models^I B$$

where $u \notin \{v_1, v_2\}$. Evaluation of $[\![v_1 + v_2]\!]\xi' \cdot I$ leads to

$$\xi', u : (V_1 + V_2)\xi \models^I B$$

But u is B's sole primary name, so we apply thinning once more and obtain:

$$u : (V_1 + V_2)\xi \models^I B.$$

Next we prove [*Rec*]. First define $W \stackrel{\text{def}}{=} (\lambda y.M)[\mu x.\lambda y.M/x]$, $\{\vec{z}\} \stackrel{\text{def}}{=} \mathsf{fv}(M) \setminus \{x, y\}$ and $V_i \stackrel{\text{def}}{=} \xi(z_i)$. Note that

$$(\lambda y.M)[W/x] \;\cong\; (\mu x.\lambda y.M)$$

Hence we shall show, for all I and ξ, whenever $\xi \models^I A$,

$$u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(n) \text{ for all } n.$$

For the inductive base we reason using thinning and the outer (IH).

$$\begin{aligned}\xi \models^I A^{\text{-}x} \quad &\Rightarrow \quad \xi, x : W \models^I A \\ &\Rightarrow \quad u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(0).\end{aligned}$$

For the inductive step we have:

$$\begin{aligned}&u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(n) \\ &\quad \Rightarrow \quad \xi, u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(n) && (1) \\ &\quad \Rightarrow \quad \xi, x : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(n)[x/u] && (2) \\ &\quad \Rightarrow \quad u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I B(n+1)\end{aligned}$$

Here implication (1) is thinning and (2) uses injective renaming. The last implication is just the outer (IH). Thus we have shown

$$u : (\lambda y.M)[\vec{V}/\vec{z}][W/x] \models^I \forall n.B(n),$$

as required.

For [*T-Abs*] we start with defining $V' \stackrel{\text{def}}{=} V[\vec{W}/\vec{y}]$ where $\xi(y_i) = W_i$. Then we prove for all I and ξ, whenever $\xi \models^I A^{\text{-}\mathrm{X}}$, we have:

$$v : \forall \mathrm{X}.V' \models^I \forall \mathrm{X}.B[u \bullet [\mathrm{X}]/v]$$

First by (IH), we have:

$$\xi \models^I A^{\text{-X}} \;\Rightarrow\; v : V' \models^I B$$

Then we can use weakening to have:

$$u : \Lambda \mathrm{X}.V', v : V' \models^I B$$

Next we use Lemma 4 to obtain:

$$u : \Lambda \mathrm{X}.V', v : V'[\alpha/\mathrm{X}] \models^{I,\mathrm{X}\mapsto\alpha} B$$

But clearly $V'[\alpha/\mathrm{X}] = [\![u \bullet [\mathrm{X}]]\!](\xi, u : \Lambda\mathrm{X}.V) \cdot (I, \mathrm{X} \mapsto \alpha)$, so in fact

$$u : \Lambda \mathrm{X}.V' \models^{I,\mathrm{X}\mapsto\alpha} B[u \bullet [\mathrm{X}]/v]$$

by Lemma 4. Hence by definition of $\mathcal{M} \models \forall \mathrm{X}.C$, we have the required result.

Next we prove [*T-App*]. Let $\xi = \vec{z} : \vec{W}$ and $I_t = \vec{\mathrm{Y}} : \vec{\alpha}$ with $I = I_a \cdot I_t$. Let $\xi' = \xi \cdot I_t$. We prove for each I and ξ,

$$\xi \models^I A \text{ implies } u : (V\beta)\xi' \models^I B[\beta I_t/\mathrm{X}]$$

We have:

$$\begin{array}{lll}
\xi \models^I A & & \\
\quad \Rightarrow & v : V\xi' \models^I \forall \mathrm{X}.B[v \bullet [\mathrm{X}]/u] & (3) \\
\quad \Rightarrow & v : V\xi' \models^{I,\mathrm{X}\mapsto\beta} B[v \bullet [\mathrm{X}]/u] & (4) \\
\quad \Rightarrow & v : V\xi', u : [\![v \bullet [\mathrm{X}]]\!]_{v:V\xi'\cdot(I\cdot\mathrm{X}\mapsto\beta I_t)} \models^{I,\mathrm{X}\mapsto\beta I_t} B & (5) \\
\quad \Rightarrow & v : V\xi', u : (V\beta)\xi' \models^{I,\mathrm{X}\mapsto\beta I_t} B & (6) \\
\quad \Rightarrow & u : (V\beta)\xi' \models^I B[\beta I_t/\mathrm{X}] & (7)
\end{array}$$

Here (3) is by (IH), (4) is from the definitions, (5) is an application of Lemma 4.

Next is [*T-Pack*]. Let $\xi = \vec{z} : \vec{W}$ and $I_t = \vec{\mathrm{Y}} : \vec{\alpha}$ with $I = I_a \cdot I_t$. Let $\xi' = \xi \cdot I_t$. We prove for each I and ξ,

$$\xi \models^I A \text{ implies } u : (\mathtt{pack}\,\langle \beta, V \rangle\,\mathtt{as}\,\exists \mathrm{X}.\alpha)\xi \models^I \exists \mathrm{X}.B$$

Then we have:

$$\begin{array}{lll}
\xi \models^I A & & \\
\quad \Rightarrow & v : V\xi' \models^I B[\langle \mathrm{X}, v\rangle/u][\beta I_t/\mathrm{X}] & (8) \\
\quad \Rightarrow & v : V\xi' \models^{I,\mathrm{X}\mapsto\beta I_t} B[\langle \mathrm{X}, v\rangle/u] & (9) \\
\quad \Rightarrow & v : V\xi', u : [\![\langle \mathrm{X}, v\rangle]\!]_{v:V\xi\cdot(I\cdot\mathrm{X}\mapsto\beta I_t)} \models^{I,\mathrm{X}\mapsto\beta I_t} B & (10) \\
\quad \Rightarrow & v : V\xi', u : \mathtt{pack}\,\langle \beta I_t, V\xi' \rangle\,\mathtt{as}\,\exists \mathrm{X}.\alpha I_t \models^{I,\mathrm{X}\mapsto\beta I_t} B & \\
\quad \Rightarrow & u : (\mathtt{pack}\,\langle \beta, V \rangle\,\mathtt{as}\,\exists \mathrm{X}.\alpha)\xi' \models^{I,\mathrm{X}\mapsto\beta I_t} B & (11) \\
\quad \Rightarrow & u : (\mathtt{pack}\,\langle \beta, V \rangle\,\mathtt{as}\,\exists \mathrm{X}.\alpha)\xi' \models^I \exists \mathrm{X}.B & (12)
\end{array}$$

Here (8) is by (IH), (9) and (10) are by Lemma 4 since $u \neq v$, $u \notin \langle \mathrm{X}, v\rangle$ by our convention on anchors and $\mathrm{X} \notin \mathsf{ftv}(\alpha) \cap \mathsf{dom}(I)$ by the variable convention. Finally (11) follows by thinning as $v \notin \mathsf{fv}(B)$.

[*T-Rec*] is immediate. ([*Abs*] is proved in the next paragraph).

Proofs of Theorem 1 (2) We begin with a couple of lemmas.

Lemma 5 *Let $\Gamma, \Theta \vdash C$ and assume given $\Gamma \vdash (\nu\vec{y})(\xi, \sigma)$, $\Theta \vdash I$.*

1. (thinning) *Assume $x \notin \mathsf{fv}(\Gamma) \cup \mathsf{fv}(\Theta)$. Then*

$$(\nu\vec{y})(\xi, \sigma \cdot x \mapsto V) \models^I C \quad \textit{implies} \quad (\nu\vec{y} \setminus x)(\xi, \sigma) \models^I C$$

2. (weakening) *For all $\Gamma' \vdash (\nu\vec{y})(\xi', \sigma')$ and $\Theta' \vdash I$ such that $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Gamma') = \mathsf{dom}(\Theta) \cap \mathsf{dom}(\Theta') = \{\vec{x}\} \cap (\mathsf{dom}(\Gamma') \cup \mathsf{dom}(\Theta') \cup \{\vec{y}\}) = \{\vec{y}\} \cap (\mathsf{dom}(\Gamma) \cup \mathsf{dom}(\Theta) \cup \{\vec{x}\}) = \emptyset$, we have:*

$$(\nu\vec{x})(\xi, \sigma) \models^I A \textit{ implies } (\nu\vec{x}\vec{y})(\xi \cdot \xi', \sigma \cdot \sigma') \models^{I \cdot I'} C$$

Lemma 6 (substitution) *Assume $x \notin \{\vec{y}\} \cup \mathsf{fv}(e) \cup \mathsf{dom}(\mathcal{M})$ and $\{\vec{y}\} \cap (\mathsf{fv}(e) \cup \mathsf{dom}(\mathcal{M})) = \emptyset$. If $\mathcal{M} = (\nu\vec{y})(\xi, \sigma)$ then*

$$\mathcal{M} \models^I C[e/x] \quad \textit{iff} \quad (\nu\vec{y})(\xi \cdot x : [\![e]\!]\mathcal{M} \cdot I, \sigma) \models^I C.$$

Lemma 7 *Suppose $C \setminus \nu x$ is defined. Then*

1. *$\mathcal{M} \models^I C \setminus \nu x$ implies $(\nu x)\mathcal{M} \models^I C$.*
2. *$\mathcal{M} \models^I C$ implies there exists $\mathcal{M}'$ such that $\mathcal{M}' \models C \setminus \nu x$ and $(\nu x)\mathcal{M}' \cong \mathcal{M}$.*

PROOF: (1) Let $C \stackrel{\text{def}}{=} (\nu x\vec{y})\vec{z}$ in C'. Then by assumption, we have $\mathcal{M} \models^I (\nu\vec{y})\vec{z}$ in C', which implies $\mathcal{M}' \models^I C'$ with $\mathcal{M} \cong (\nu\vec{y})\mathcal{M}'$ and $\mathcal{M}' \stackrel{\text{def}}{=} \xi \cdot \vec{z} : \vec{W}\vec{y} \mapsto \vec{V} \cdot x \mapsto V' \cdot \sigma$ for some $\xi, \vec{W}, \vec{V}, V'$ and σ. This implies $(\nu\vec{y}x)\mathcal{M}' \models^T C$. Hence $(\nu x)\mathcal{M} \models^T C$, as required.
(2) If $\mathcal{M} \models^I (\nu x\vec{y})\vec{z}$ in C', then $\mathcal{M} \cong (\nu x\vec{y})\mathcal{M}'$ and $\mathcal{M}' \models^I C'$ such that $\mathcal{M}' \stackrel{\text{def}}{=} \xi \cdot \vec{z} : \vec{W}\vec{y} \mapsto \vec{V} \cdot x \mapsto V' \cdot \sigma$ for some $\xi, \vec{W}, \vec{V}, V'$ and σ. This implies $(\nu\vec{y})\mathcal{M}' \models^I (\nu\vec{y})\vec{z}$ in $C' \stackrel{\text{def}}{=} C \setminus \nu x$ where $(\nu x\vec{y})\mathcal{M}_1 \cong \mathcal{M}$, as required. □

Lemma 8 *Assume $\mathcal{M}_i = (\nu\vec{y})(\xi', \sigma_i)$ for $i = 1, 2$. Then*

$$\mathcal{M}_1 \models^I A \quad \textit{iff} \quad \mathcal{M}_2 \models^I A.$$

Lemma 9 *Assume $\mathcal{M}_1 \equiv_{\mathsf{gc}} \mathcal{M}_2$. Then*

$$\mathcal{M}_1 \models^I C \quad \textit{iff} \quad \mathcal{M}_2 \models^I C.$$

PROOF: By $\equiv_{\mathsf{gc}} \subseteq \cong$. □

Now we prove (2). By Lemma 9, we ignore garbage collections.

Next we prove [*Abs*]. Assume $\mathcal{M} \models^I C^{\text{-}x} \wedge A^{\text{-}x} \wedge A_1^x$ with $\mathcal{M} = (\nu\vec{y})(\xi, \sigma)$. Then we prove:

$$u : (\lambda x.M)\xi \models^I B$$

with $\forall x.(A_1 \supset \{C\}\, u \bullet x \searrow m\, \{C'\}) \supset B$. By (IH) and noting $x \notin \mathsf{fv}(A) \cup \mathsf{fv}(C)$, we know, under the assumption $\xi \models^I A \wedge x : V\xi \models^I A_1 \wedge \sigma \models^I C$, we have:

$$(\nu \vec{y})(M[V/x]\xi, \sigma) \Downarrow (\nu \vec{z})(W, \sigma') \text{ s.t. } (\nu \vec{z})(m : W, \sigma') \models^I C'$$

Note that $(\nu \vec{y})((\lambda x.M)V)\xi, \sigma) \Downarrow (\nu \vec{z})(W, \sigma')$. From this and by definition of the satisfaction, under the assumption $\xi \models^I A$, we have:

$$x : V\xi \models^I A_1 \supset \mathcal{M} \cdot x : V\xi \cdot u : \lambda x.M\xi \models^I \{C\}\, u \bullet x \searrow m\, \{C'\}$$

Then by definition of the satisfaction, we have:

$$\mathcal{M} \cdot u : \lambda x.M\xi \models^I \forall x.(A_1 \supset \{C\}\, u \bullet x \searrow m\, \{C'\})$$

Since u is the sole primary name, the thinning gives us:

$$u : \lambda x.M\xi \models^I \forall x.(A_1 \supset \{C\}\, u \bullet x \searrow m\, \{C'\})$$

Finally by $\forall x.(A_1 \supset \{C\}\, u \bullet x \searrow m\, \{C'\}) \supset B$, we have the required result.

Next we prove [*App*]. We assume $\Gamma \vdash V : \alpha \overset{\varphi}{\Rightarrow} \beta$ and $\Gamma \vdash W : \alpha$. Assume

$$\mathcal{M} = (\nu \vec{y})(\xi, \sigma_\varphi \cdot \sigma)$$

where $\mathsf{dom}(\sigma_\varphi) = \varphi$. Then if

$$(\nu \vec{y})((VW)\xi, \sigma_\varphi \cdot \sigma) \Downarrow (\nu \vec{z})(V', \sigma'_\varphi \cdot \sigma)$$

(where the substitution ξ makes VW appropriately semi-closed) then we have to show

$$(\nu \vec{y})(u : V', \sigma'_\varphi, \sigma) \models^I C'$$

By the (IH) and weakening, we have

$$(\nu \vec{y})(\xi \cdot v : V\xi, \sigma_\varphi \cdot \sigma) \models^I C_1$$
$$(\nu \vec{y})(\xi \cdot w : W\xi, \sigma_\varphi \cdot \sigma) \models^I C_2$$

so the definition of the satisfaction of $\wedge$ delivers

$$\mathcal{M}' \stackrel{\text{def}}{=} (\nu \vec{y})(\xi \cdot v : V\xi \cdot w : W\xi, \sigma_\varphi \cdot \sigma) \models^I C_1 \wedge C_2$$

By the other premise then

$$\mathcal{M}' \models^I \{C\}\, v \bullet w \searrow u\, \{C'\}$$

Clearly $[\![v]\!]\mathcal{M}' \cdot I = V\xi$ and $[\![w]\!]\mathcal{M}' \cdot I = W\xi$. Then by the definition of the satisfaction (noting $\mathcal{M} \setminus \varphi = (\nu \vec{y})(\xi, \sigma)$), we must have

$$(\nu \vec{z})(\xi, v : V\xi \cdot w : W\xi \cdot u : V', \sigma'_\varphi \cdot \sigma) \models^I C'$$

Finally by thinning, we have

$$(\nu \vec{y})(u : V', \sigma'_\varphi \cdot \sigma) \models^I C'$$

because neither v nor w can occur in C'.

[*If*] and [*Case*] are mechanical and similar to the proofs of (1).

Next case is [*Assign*]. We'll prove it for the following slightly more general form of the rule.

$$\frac{\{C\}\, V :_v \{A\} \quad (C[i/x] \wedge A[!x/v]) \supset C'[()/u] \quad i \text{ fresh}}{\{C\}\, x := V :_u \{C'\}}$$

With $\mathcal{M} = (\nu \vec{y})(\xi, \sigma \cdot x \mapsto W)$, assume $\mathcal{M} \models^I C$. Then we prove:

$$(\nu \vec{y})(\xi \cdot u : (), x \mapsto V\xi \cdot \sigma) \models^I C'$$

We derive:

$$\begin{array}{llr} & \mathcal{M} \models^I C & \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto W \cdot i \mapsto W) \models^I C & (13) \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot i \mapsto W) \models^I C[i/x] & (14) \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V\xi \cdot i \mapsto W) \models^I C[i/x] & (15) \end{array}$$

Here (13) and (14) are weakening, which is applicable as i, is fresh in the former and x in the latter application; (15) is Lemma 6. By (IH) and $\mathcal{M} \models^I C$ we get

$$\begin{array}{llr} & (\nu \vec{y})(\xi, \sigma \cdot v \mapsto V\xi \cdot x \mapsto W) \models^I A & \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V\xi) \models^I A[!x/v] & (16) \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V\xi \cdot i \mapsto W) \models^I A[!x/v] & (17) \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V\xi \cdot i \mapsto W) \models^I C[i/x] \wedge A & (18) \\ \Rightarrow & (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V\xi \cdot i \mapsto W) \models^I C'[()/u] & (19) \\ \Rightarrow & (\nu \vec{y})(\xi \cdot u : (), \sigma \cdot x \mapsto V\xi \cdot i \mapsto W) \models^I C' & (20) \\ \Rightarrow & (\nu \vec{y})(u : (), \sigma \cdot x \mapsto V\xi) \models^I C' & (21) \end{array}$$

Here (16) is Lemma 6 together with Lemma 8. (17) is weakening. (18) uses (15). (19) comes from the second premise $(C[i/x] \wedge A[!x/v]) \supset C'[()/u]$. Then (20) is by the substitution lemma. The last inference is an application of thinning, which is permissible as i does not occur in C'.

For [*Deref*] assume $\mathcal{M} = (\nu \vec{y})(\xi, \sigma \cdot x \mapsto V)$. and

$$(\nu \vec{y})(!x, \sigma \cdot x \mapsto V) \Downarrow (\nu \vec{y})(V, \sigma \cdot x \mapsto V)$$

Then we prove:

$$(\nu \vec{y})(u : V, \sigma \cdot x \mapsto V) \models^I C'$$

We observe:

$$\begin{array}{llr} & \mathcal{M} \models^I C & \\ \Rightarrow & \mathcal{M} \models^I C'[!x/u] & (22) \\ \Rightarrow & (\nu \vec{y})(\xi \cdot u : [\![!x]\!]\mathcal{M} \cdot I, \sigma \cdot x \mapsto V) \models^I C' & (23) \\ \Rightarrow & (\nu \vec{y})(\xi \cdot u : V, \sigma \cdot x \mapsto V) \models^I C' & \end{array}$$

Here (22) follows from the premise of the rule, (23) is a direct consequence of Lemma 4 because u does not appear in ξ, V and σ. The final equivalence comes from evaluating $[\![!x]\!]\mathcal{M} \cdot I = V$. Then thinning derives the required result.

[*Unpack*] is similar to [*T-Pack*], hence we omit.

[*Promte*] is obvious by (IH) and the definition of the satisfaction.

For [*Weak*] let $\mathcal{M} = (\nu\vec{y})(\xi,\sigma)$. Then $(\nu\vec{y})(M\xi,\sigma) \Downarrow (\nu\vec{z})(V,\sigma')$ together with the (IH) implies

$$(\nu\vec{z})(\xi \cdot m : V,\sigma') \models^I C'. \tag{24}$$

Under the assumption $\varphi \cap \varphi' = \emptyset$ and $\mathcal{M} \models^I C \wedge C_1$, We have to prove

$$(\nu\vec{z})(\xi \cdot m : V,\sigma') \models^I C' \wedge C_1$$

Define $\sigma|_\varphi \stackrel{\text{def}}{=} \{\sigma(w) \mid w \in \varphi\}$. Then we have:

$$\begin{aligned}
\mathcal{M} \models^I C \wedge C_1 \quad &\Rightarrow \quad \mathcal{M} \models^I C_1 && (25)\\
&\Rightarrow \quad (\nu\vec{y} \cap \varphi')(\xi,\sigma|_{\varphi'}) \models^I C_1 && (26)\\
&\Rightarrow \quad (\nu\vec{z} \cap \varphi')(\xi,\sigma'|_{\varphi'}) \models^I C_1 && (27)\\
&\Rightarrow \quad (\nu\vec{z})(\xi,\sigma') \models^I C_1 && (28)\\
&\Rightarrow \quad (\nu\vec{z})(\xi \cdot m : V,\sigma') \models^I C_1 && (29)
\end{aligned}$$

where $\sigma|_{\varphi'} = \sigma'|_{\varphi'}$ and $\vec{y} \cap \varphi' = \vec{z} \cap \varphi'$. Hence we have (27). Then weakening gives (28). Now by the definition of $\wedge$, we derive the required result.

The rules [*Consequence*], [$Aux_\forall$] and [$Aux_\exists$] are immediate from the definition of the satisfaction relation and the laws of first-order logic.

Next is [*New*]. Let $\mathcal{M} = (\nu\vec{y})(\xi,\sigma^{-x})$. Clearly

$$\begin{aligned}
&(\nu\vec{y})(\texttt{new var}\ x := V\xi\ \texttt{in}\ N\xi)\\
&\quad \rightarrow \quad (\nu\vec{y}x)(N\xi,\sigma \cdot x \mapsto V\xi) \Downarrow (\nu\vec{z})(W,\sigma')
\end{aligned}$$

is the only way to converge to a value. Here we prove:

$$(\nu\vec{y})(u : W,\sigma') \models^I C'$$

By the first premise $\{C\}V :_v \{C_1\}$, we have:

$$(\nu\vec{y})(\xi \cdot v : V\xi,\sigma) \models^I C_1$$

By the (IH) on the second premise and Lemma 6, we have:

$$\begin{aligned}
&(\nu\vec{y})(\xi \cdot v : V\xi,\sigma) \models^I C_1\\
&\quad \Rightarrow \quad (\nu\vec{y})(\xi,\sigma \cdot x \mapsto V\xi) \models^I C_1[!x/v] && (30)
\end{aligned}$$

Then (30) and Lemma 1 gives

$$\begin{aligned}
&(\nu\vec{y})(N\xi,\sigma \cdot x \mapsto V\xi) \Downarrow (\nu\vec{z} \setminus x)(W,\sigma')\\
&\quad \Rightarrow \quad (\nu\vec{y} \setminus x)(\xi \cdot u : W,\sigma') \models^I C' \setminus \nu x && (31)\\
&\quad \Rightarrow \quad (\nu\vec{y})(\xi \cdot u : W,\sigma') \models^I C' && (32)
\end{aligned}$$

Here (31) is by the second premise and the (IH), while (32) follows from Lemma 7.

The next rule is [*CoNew*]. With $\mathcal{M} = (\nu\vec{y})(\xi,\sigma^{-x})$, we derive

$$\begin{aligned}
&(\nu\vec{y})(M\xi,\sigma) \Downarrow (\nu\vec{z})(V,\sigma')\\
&\quad \Rightarrow \quad (\nu\vec{y} \setminus x)(M\xi,\sigma) \Downarrow (\nu\vec{z} \setminus x)(V,\sigma') && (33)\\
&\quad \Rightarrow \quad (\nu\vec{z} \setminus x)(\xi \cdot u : V,\sigma') \models^I C' \setminus \nu x && (34)\\
&\quad \Rightarrow \quad (\nu\vec{z})(\xi \cdot u : V,\sigma') \models^I C' && (35)\\
&\quad \Rightarrow \quad (\nu\vec{z})(u : V,\sigma') \models^I C' && (36)
\end{aligned}$$

Here (33) is by Lemma 6, (34) follows by (IH), and (35) is a consequence of Lemma 7. (36) is by thinning.

We now prove [*LetN*]. As usual we assume with $\mathcal{M} = (\nu\vec{y})(\xi,\sigma)$. By determinacy or reductions here is only the following potential reduction sequence towards a value.

$$\begin{aligned}
&(\nu\vec{y})(M\xi,\sigma) \rightarrow^* (\nu\vec{y}\vec{a})(W\xi,\sigma'). && (37)\\
&\quad \Rightarrow \quad (\nu\vec{y})((\texttt{let}\ x = M\ \texttt{in}\ N)\xi,\sigma)\\
&\quad \rightarrow^* \quad (\nu\vec{y}\vec{a})((\texttt{let}\ x = W\ \texttt{in}\ N)\xi,\sigma')\\
&\quad \rightarrow \quad (\nu\vec{y}\vec{a})((N[W/x])\xi,\sigma')\\
&\quad \Downarrow \quad (\nu\vec{b})(V,\sigma'')
\end{aligned}$$

In this derivation, (37) is given by Lemma 1 (3) and has the side condition $\{\vec{a}\} \cap \mathsf{fv}(N) = \emptyset$. With this assumption, we prove:

$$(\nu\vec{b})(x : V,\sigma'') \models^I C'$$

By (IH), we have:

$$\begin{aligned}
&(\nu\vec{y}\vec{a})(\xi \cdot x : W\xi,\sigma') \models^I (\nu\vec{a})x\vec{b}\ \mathsf{in}\ C_1\\
&\quad \Rightarrow \quad (\nu\vec{y})(\xi \cdot x : W\xi,\sigma') \models^I C_1 && (38)\\
&\quad \Rightarrow \quad (\nu\vec{b} \setminus \vec{a})(\xi \cdot x : V,\sigma') \models^I C' \setminus \nu\vec{a} && (39)\\
&\quad \Rightarrow \quad (\nu\vec{b})(\xi \cdot x : V,\sigma'') \models^I C' && (40)
\end{aligned}$$

Here (38) and (40) are a consequence of Lemma 7 while (39) comes from the second premise and the (IH). Finally thinning gives us the required result.

E Proofs for Section 6 (Noninterference Proofs)

In the following we present the proofs of noninterference for three different classes of programs (pure functions, functions with global state, and those with local state), as well as associated definitions. At the end we also discuss extensions and refinements (polymorphism, recursive types, explicit treatment of secrecy, alterative treatment of termination).

E.1 Pure Functions (1): Preparation

For the proof of Theorem 3 (noninterference), we use two lemmas in addition to Proposition 6. We also list secrecy typing rules in full. *All substitutions in the following technical development use closed values as their codomain*. We also recall $\Gamma \vdash \sigma$ says σ is typed under Γ.

Notation 1 Assuming $\Gamma \vdash \sigma$, $\Gamma \models_{\mathsf{sec}} \sigma$ stands for $\models_{\mathsf{sec}} V : \alpha$ for each $\sigma(x) = V$ with $\Gamma(x) = \alpha$. Ω^{α} denotes the least terminating closed program in α (if any). $\mathtt{mnd}(M,N:\alpha,\, s)$ stands for $\models M,N :_{u,v} \mathtt{mnd}(u,v:\alpha,\, s)$ with M and N closed. Further $\mathtt{mnd}(\sigma_1,\sigma_2:\Gamma,\, s)$ means each pair of values in $\sigma_{1,2}$ are thus related.

Lemma 10 *(1)* $\Gamma, x{:}\alpha \models_{\mathsf{sec}} M : \beta$ *iff* $\models_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \Rightarrow \beta$.

(2) $\models_{\mathsf{sec}} M : \alpha$ *iff* $\models M :_u \mathtt{NI}(u{:}\alpha)$.

(3) If $\Gamma \models_{\mathsf{sec}} M : \alpha$ *and* $\Gamma \models_{\mathsf{sec}} \sigma$ *then* $\models M\sigma :_u \mathtt{NI}(u{:}\alpha)$.

PROOF: All are direct from the definition. (3) uses (2).□

For the next lemma, recall a formula A is *admissible at* u^{α} when (i) $\models \Omega :_u A$ and (ii) for a sequence $\{V_i\}_{i \in \mathsf{Nat}}$ approximating V_ω, if $\models V_i :_u A$ for each i then $\models V_\omega :_u A$ where $\sqsubseteq$ and approximation are defined as follows. In (1) below, $C[\;]^{\beta}_{\alpha}$ is a context with hole typed α and result typed β. A lub is least upper bound (in the induced partial order).

Definition 4 *(1)* (termination ordering [9]) $M_1 \sqsubseteq M_2$ *with* $\vdash M_{1,2} \triangleright \alpha$ *when* $C[M_1] \Downarrow$ *implies* $C[M_2] \Downarrow$ *for each* $C[\cdot]^{\llcorner\mathsf{Unit}\lrcorner}_{\alpha}$.

(2) (approximation) *A sequence* $\{V_i\}_{i \in \mathsf{Nat}}$ approximate V_ω *if: (2-a)* $V_i \sqsubseteq V_{i+1} \sqsubseteq V_\omega$ *for each i, and (2-b)* $C[V_\omega]$ *is a lub of* $\{C[V_i]\}_{i \in \mathbb{N}}$ *for each well-typed* $C[\,\cdot\,]$.

In (2-b), we close the ordering by each context so that a pathological lub (cf.[9, §4]) is excluded (actually the only approximating chain we use is $\{V_i\}_{i \in \mathbb{N}}$ with each V_i having a pointed type α, such that, for some $x{:}\alpha \vdash V : \alpha$: $V_0 = \Omega$ and $V_{i+1} = V[V_i/x]$, which easily approximates $\mu x.V$).

Below $x \bullet y_1..y_n$ stands for $(..(x \bullet y_1) \bullet y_2) \bullet \ldots \bullet y_n)$. $\beta_1..\beta_n \Rightarrow \alpha$ stands for $\beta_1 \Rightarrow (\beta_2 \Rightarrow \ldots.(\beta_n \Rightarrow \alpha)..)$.

Lemma 11 *(1)* $\mathtt{NI}(u{:}\alpha)$ *is admissible at u for each* α.

(2) If $\vec{y}{:}\vec{\beta}, x{:}\alpha \models_{\mathsf{sec}} M : \gamma$ *then* $\vec{y}{:}\vec{\beta}, z{:}\vec{\beta} \Rightarrow \alpha \models_{\mathsf{sec}} M[z \bullet \vec{y}/x] : \gamma$.

(3) If $\models_{\mathsf{sec}} \mu z.\lambda\vec{y}.V[z \bullet \vec{y}/x] : \vec{\beta} \Rightarrow \alpha$ *then* $\models_{\mathsf{sec}} \lambda\vec{y}.\mu x.V : \vec{\beta} \Rightarrow \alpha$.

PROOF: For (1), for the first condition, the least program Ω^{α} surely satisfies the predicate. For the second condition, we use induction on types to show $\mathtt{mnd}(\pi_1(u),\pi_1(u):\alpha,\, s)$ has the approximation property for each α. Assume a chain $\{V_i\}$ (resp. $\{W_i\}$) approximates V_ω (resp. W_ω) and $\models V_i, W_i :_{v,w} \mathtt{mnd}(v,w:\alpha,\, s)$ for each i. The only interesting case is $\alpha = \beta \Rightarrow \llcorner\gamma\lrcorner$. Let $\models T,U :_{t,u} \mathtt{mnd}'(t,u:\beta,s)$. If

$$[\mathit{Var}]\ \frac{x{:}\alpha \in \Gamma}{\Gamma \vdash_{\mathsf{sec}} x : (\alpha)_s} \qquad [\mathit{Op}]\ \frac{\Gamma \vdash_{\mathsf{sec}} M_i : \alpha_i \quad \mathsf{protect}(\beta) \sqsubseteq \mathsf{protect}(\alpha_i)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{op}(\vec{M}) : \beta}$$

$$[\mathit{Abs}]\ \frac{\Gamma, x{:}\alpha \vdash_{\mathsf{sec}} M : \beta}{\Gamma \vdash_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \Rightarrow \beta} \qquad [\mathit{App}]\ \frac{\Gamma \vdash V : \alpha \Rightarrow \beta \quad \Gamma \vdash_{\mathsf{sec}} W : \alpha}{\Gamma \vdash_{\mathsf{sec}} VW : \beta}$$

$$[\mathit{Rec}]\ \frac{\Gamma, x : \alpha \Rightarrow \beta \vdash_{\mathsf{sec}} \lambda y^{\alpha}.M : \alpha \Rightarrow \beta \quad \alpha \Rightarrow \beta \text{ pointed}}{\Gamma \vdash_{\mathsf{sec}} \mu x^{\alpha \Rightarrow \beta}.\lambda y^{\alpha}.M : \alpha \Rightarrow \beta}$$

$$[\mathit{If}]\ \frac{\Gamma \vdash_{\mathsf{sec}} V : \mathsf{Bool}_s \quad \Gamma \vdash_{\mathsf{sec}} N_{1,2} : \alpha \quad s \sqsubseteq \mathsf{protect}(\alpha)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{if}\ V\ \mathtt{then}\ N_1\ \mathtt{else}\ N_2 : \alpha}$$

$$[\mathit{Inj}]\ \frac{\Gamma \vdash_{\mathsf{sec}} V : \alpha_i}{\Gamma \vdash_{\mathsf{sec}} \mathtt{inj}_i(V) : \alpha_1 +_s \alpha_2}$$

$$[\mathit{Case}]\ \frac{\Gamma \vdash_{\mathsf{sec}} V : \alpha_1 +_s \alpha_2 \quad \Gamma, x_i : \alpha_i \vdash_{\mathsf{sec}} M_i : \beta \quad s \sqsubseteq \mathsf{protect}(\beta)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i^{\alpha_i}).M_i\}_{i \in \{1,2\}} : \beta}$$

$$[\mathit{Pair}]\ \frac{\Gamma \vdash_{\mathsf{sec}} V_i : \alpha_i\ (i = 1,2)}{\Gamma \vdash_{\mathsf{sec}} \langle V_1, V_2 \rangle : \alpha_1 \times \alpha_2} \qquad [\mathit{Proj}]\ \frac{\Gamma \vdash_{\mathsf{sec}} V : \alpha_1 \times \alpha_2}{\Gamma \vdash_{\mathsf{sec}} \pi_i(V) : \alpha_i\ (i = 1,2)}$$

$$[\mathit{Let}]\ \frac{\Gamma, x : \alpha \vdash_{\mathsf{sec}} M : \beta \quad \Gamma \vdash_{\mathsf{sec}} N : \alpha}{\Gamma \vdash_{\mathsf{sec}} \mathtt{let}\ x^{\alpha} = N\ \mathtt{in}\ M : \beta} \qquad [\mathit{Lift}]\ \frac{\Gamma \vdash_{\mathsf{sec}} M : \alpha}{\Gamma \vdash_{\mathsf{sec}} M : \llcorner\alpha\lrcorner_s}$$

$$[\mathit{Seq}]\ \frac{\Gamma \vdash_{\mathsf{sec}} N : \llcorner\alpha\lrcorner_s \quad \Gamma, x{:}\alpha \vdash_{\mathsf{sec}} M : \beta \quad \beta \text{ partial} \quad s \sqsubseteq \mathsf{protect}(\beta)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{seq}\ x^{\alpha} = N\ \mathtt{in}\ M : \beta}$$

Figure 2: Secrecy Typing Rules for PCFv

$V_\omega T \Downarrow$ then by $V_\omega T$ being the lub of $\{V_i T\}_i$ we have $V_i T \Downarrow$ for some i hence $W_i U \Downarrow$. By induction we know $\models V_\omega T, W_\omega U :_{v,w} \mathtt{mnd}(v,w:\alpha,\, s)$. If $V_\omega T \Uparrow$ the result is immediate. The remaining cases are easy. (2) is direct from the definition. (3) is standard [3, 8].□

Proposition 7 (Proposition 6 in the main paper)

(1) $\mathsf{protect}((\alpha)_s) = \mathsf{protect}(\alpha) \sqcup s$.

(2) $\forall u, v.\ (\mathtt{mnd}(u,v{:}\alpha,\, s) \supset \mathtt{mnd}(u,v{:}(\alpha)_{s'},\, s))$ *for any* s'.

(3) $\mathsf{protect}(\alpha) \not\sqsubseteq s \supset (\mathtt{NI}(u{:}\alpha) \wedge \mathtt{NI}(v{:}\alpha) \supset \mathtt{mnd}(u,v{:}\alpha,\, s))$.

PROOF: All are easy by induction on types.□

The secrecy typing rules for PCFv are given in Figure 2. $[op]$ is a rule schema to be instantiated into each concrete n-ary first-order operation (operations include constants when the arity is zero). Detailed illustration of each rule is given in [4, §3]. Extensions and refinements are discussed in later subsections.

E.2 Pure Functions (2): Proof of Theorem 3

Theorem 3 (Theorem 3 in the main paper) $\Gamma \vdash_{\mathsf{sec}} M : \alpha$ *implies* $\Gamma \models_{\mathsf{sec}} M : \alpha$.

PROOF: By rule induction on the secrecy typing rules in Figure 2. Below (IH) stands for "by induction hypothesis".

(Def) stands for "by definition".

(Var) $\mathtt{mnd}'(\sigma_1,\sigma_2:\Gamma, s)$

$$\begin{array}{lll} \supset & \mathtt{mnd}'(x\sigma_1,x\sigma_2:\alpha, s) & \text{(Notation 1)} \\ \supset & \mathtt{mnd}'(x\sigma_1,x\sigma_2:(\alpha)_{s'}, s) & \text{(Prop. 6 (3)).} \end{array}$$

(Op) Let T and T' be atomic types of level s'. Let op be the operation of type $\vec{T}\Rightarrow T'$. It suffices to show:

$$\mathtt{mnd}'(\vec{x},\vec{y}:\vec{T}, s) \quad\supset\quad \mathtt{mnd}'(\mathsf{op}(\vec{x}),\mathsf{op}(\vec{y}):T', s)$$

If $s'\sqsubseteq s$ then $\mathtt{mnd}'(\vec{x},\vec{y}:T, s)$ means $\vec{x}=\vec{y}$ hence done. If $s'\not\sqsubseteq s$ then $\mathtt{mnd}'(\mathsf{op}(\vec{x}),\mathsf{op}(\vec{y}):T', s)$ by Proposition 6(2).

(Abs) Direct from Lemma A (1).

(App) $\mathtt{mnd}'(\sigma_1,\sigma_2:\Gamma, s)$

$$\begin{array}{lll} \supset & \mathtt{mnd}(V\sigma_1,V\sigma_2:\alpha\Rightarrow\beta, s)\ \wedge & \\ & \mathtt{mnd}'(W\sigma_1,W\sigma_2:\alpha, s) & \text{(IH, Lem. A (2))} \\ \supset & \mathtt{mnd}((VW)\sigma_1,(VW)\sigma_2:\beta, s) & \text{(Def).} \end{array}$$

In detail, the second inference uses the entailment:

$$\begin{array}{ll} \multicolumn{2}{l}{\mathtt{mnd}'(v,v':\alpha\Rightarrow\beta, s)\wedge\mathtt{mnd}'(w,w':\alpha, s)} \\ \equiv & \forall x^{\alpha},y^{\alpha}.(\ \mathtt{mnd}'(x,y:\alpha, s)\supset\mathtt{mnd}(v\bullet x,v'\bullet y:\beta, s)\ ) \\ & \wedge\ \mathtt{mnd}'(w,w':\alpha, s) \\ \supset & \mathtt{mnd}(v\bullet w,v'\bullet w':\beta, s). \end{array}$$

which is already given in the main section.

(Rec) We first claim, with α pointed:

Claim. $x:\alpha\models_{\mathsf{sec}}\lambda y.M:\alpha$ implies $\models_{\mathsf{sec}}\mu x.\lambda y.M:\alpha$.

The proof of the claim relies on the soundness of [Rec-$\Uparrow$].

$x:\alpha\models_{\mathsf{sec}}\lambda y.M:\alpha$

$$\begin{array}{lll} \supset & \mathtt{NI}(x:\alpha)\models\lambda y.M:_u\mathtt{NI}(u:\alpha) & \text{(IH, Lem. A (3))} \\ \supset & \models\mu x.\lambda y.M:_u\mathtt{NI}(u:\alpha)u & \text{(Lem. B (1), [Rec-}\Uparrow\text{])} \end{array}$$

as required (end of the proof of the Claim).

We can now infer as follows, setting $V\stackrel{\text{def}}{=}\lambda y.M$.

$\vec{w}:\vec{\beta},x:\alpha\models_{\mathsf{sec}}V:\alpha$

$$\begin{array}{lll} \supset & \vec{w}:\vec{\beta},z:\vec{\beta}\Rightarrow\alpha\models_{\mathsf{sec}}V[z\bullet\vec{w}/x]:\alpha & \text{(Lem. B (2))} \\ \supset & z:\vec{\beta}\Rightarrow\alpha\models_{\mathsf{sec}}\lambda\vec{w}.(V[z\bullet\vec{w}/x]):\vec{\beta}\Rightarrow\alpha & \text{(Lem. A (1))} \\ \supset & \models_{\mathsf{sec}}\mu z.\lambda\vec{w}.(V[z\bullet\vec{w}/x]):\vec{\beta}\Rightarrow\alpha & \text{(Claim above)} \\ \supset & \models_{\mathsf{sec}}\lambda\vec{w}.\mu x.V:\vec{\beta}\Rightarrow\alpha & \text{(Lem. B (3))} \\ \supset & \vec{w}:\vec{\beta}\models_{\mathsf{sec}}\mu x.V:\alpha & \text{(Lem. A (1)).} \end{array}$$

(If) Subsumed by **(Case)** below.

(Inj) It suffices to show, for each $i\in\{1,2\}$:

$$\mathtt{mnd}(x,y:\alpha_i, s)\ \supset\ \mathtt{mnd}(\mathtt{inj}_i(x),\mathtt{inj}_i(y):\alpha_1+_{s'}\alpha_2, s)$$

which is immediate from the definition.

(Case) Consider the sum type $\gamma=\alpha_1+_{s'}\alpha_2$ (note s' replaces s in the typing rule). Accordingly the secrecy condition becomes $s'\sqsubseteq\mathsf{protect}(\beta)$. We have two cases. Below let $N\stackrel{\text{def}}{=}\mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i^{\alpha_i}).M_i\}_{i\in\{1,2\}}$. If $s'\sqsubseteq s$, we infer:

$$\begin{array}{lll} \multicolumn{3}{l}{\mathtt{mnd}'(\sigma,\sigma':\Gamma, s)} \\ \supset & \mathtt{mnd}'(V\sigma,V\sigma':\gamma, s) & \text{(IH, Lem. A (3))} \\ \supset & V\sigma\cong\mathtt{inj}_i(W_i)\ \wedge\ V\sigma'\cong\mathtt{inj}_i(W_i')\ \wedge & \\ & \mathtt{mnd}'(W_i,W_i':\alpha_i, s) & (s'\sqsubseteq s) \\ \supset & \mathtt{mnd}(M_i\sigma[W_i/x_i],M_i\sigma'[W_i'/x_i]:\beta, s) & \text{(IH)} \end{array}$$

With i being either 1 or 2, hence done. On the other hand, if $s'\not\sqsubseteq s$, then $\mathsf{protect}(\beta)\not\sqsubseteq s$ by the secrecy condition $s'\sqsubseteq\mathsf{protect}(\beta)$, hence by Proposition 6 (3), we have $\mathtt{mnd}(N\sigma,N\sigma':\beta, s)$ regardless of σ,σ'.

(Pair, Proj) Because $\mathtt{mnd}'(x_1,x_1':\alpha_1, s)\wedge\mathtt{mnd}'(x_2,x_2':\alpha_2, s)\equiv\mathtt{mnd}'(\langle x_1,x_2\rangle,\langle x_1',x_2'\rangle:\alpha_1\times\alpha_2, s)$ (which is immediate from the definition).

(Let) Subsumed by **(Seq)** below.

Note. In **(Lift)** and **(Seq)** below, we assume the mnd-predicate for partial types, $\mathtt{mnd}(e,e':\llcorner\alpha\lrcorner_{s'}, s)$, given as:

$$\begin{array}{l} s'\sqsubseteq s\ \supset\ (e\Downarrow\ \equiv e'\Downarrow)\ \wedge \\ \qquad\forall x^{\alpha},y^{\alpha}.((e=\llcorner x\lrcorner\wedge e'=\llcorner y\lrcorner)\ \supset\ \mathtt{mnd}(x,y:\alpha, s)) \end{array}$$

An older version contains a confusion in this clause, corrected in the versions dated July 26 or later.

(Lift) By $\mathtt{mnd}(x,y:\alpha, s)\supset\mathtt{mnd}(\llcorner x\lrcorner,\llcorner y\lrcorner:\llcorner\alpha\lrcorner_{s'}, s)$.

(Seq) Let $L\stackrel{\text{def}}{=}\mathtt{seq}\ x=N\ \mathtt{in}\ M$ and $\gamma\stackrel{\text{def}}{=}\llcorner\alpha\lrcorner_{s'}$ (note s' replaces s in the typing rule). The secrecy condition becomes $s'\sqsubseteq\mathsf{protect}(\beta)$. First, if $s'\sqsubseteq s$, we infer:

$$\begin{array}{lll} \multicolumn{3}{l}{\mathtt{mnd}'(\sigma,\sigma':\Gamma, s)} \\ \supset & \mathtt{mnd}(N\sigma,N\sigma':\gamma, s) & \text{(IH, Lem. A (3))} \\ \supset & \mathtt{mnd}(N\sigma,N\sigma':\alpha, s)\ \vee\ (N\sigma\Uparrow\wedge N\sigma'\Uparrow) & \text{(Def)} \\ \supset & \mathtt{mnd}(M\sigma[N\sigma/x],M\sigma'[N\sigma'/x]:\alpha, s)\quad\vee & \\ & (L\sigma\Uparrow\wedge L\sigma'\Uparrow) & \text{(IH)} \\ \supset & \mathtt{mnd}(L\sigma,L\sigma':\beta, s) & \text{(Def)} \end{array}$$

If $s'\not\sqsubseteq s$, we have $\mathsf{protect}(\beta)\not\sqsubseteq s$ by the secrecy condition $s'\sqsubseteq\mathsf{protect}(\beta)$, hence by Prop. 6 (3) $\mathtt{mnd}(L\sigma,L\sigma':\beta, s)$ regardless of σ,σ'. We have now exhausted all cases. □

E.3 Global State (1): Preparation

We present a brief summary of the formal constructions for secrecy in global state. Since we should work with both substitutions and stores, the following notation is convenient.

Notation 2 $\xi,\xi',\ldots$ range over substitutions for semi-closed values (recall a value is *semi-closed* if its free names are all typed by reference types). $\sigma,\sigma',\ldots$ range over stores, i.e. maps from reference typed names to semi-closed values. We continue to write $\mathtt{mnd}(\xi_1,\xi_2:\Gamma, s)$ and $\mathtt{mnd}(\sigma_1,\sigma_2:\Gamma, s)$ for the conjunction of component-wise mnd-relationship.

First, basic definitions for extended secrecy types follow.

1. We set $\mathsf{protect}(\mathsf{Refr}_s(\alpha))=\mathsf{protect}(\mathsf{Ref}_s(\alpha))=\mathsf{protect}(\gamma\stackrel{\varphi}{\Rightarrow}\alpha)=\mathsf{protect}(\alpha)$. We also restrict pointed types to the shape $\alpha\stackrel{\varphi}{\Rightarrow}\llcorner\beta\lrcorner_s$ (cf. [4]).

2. $\mathsf{Refr}_s(\alpha)$ and $\mathsf{Ref}_s(\alpha)$ are *structurally secure* iff $s \sqsubseteq \mathsf{protect}(\alpha)$. *We only consider structurally secure reference types from now on.*

3. $\mathsf{protectE}(\rho)$ is given as follows.

$$\mathsf{protectE}(\rho) \;=\; \begin{cases} s & (\rho = \mathsf{Ref}_s(\alpha)) \\ \mathrm{H} & (\text{if else}) \end{cases}$$

Further $\mathsf{protectE}(\Gamma) = \sqcap\{\mathsf{protectE}(\alpha) \mid \alpha \in \mathsf{ran}(\Gamma)\}$.

We also refine the mnd-predicate, for which the only change is in arrow types. We set, with $\gamma = \alpha \overset{\vec{y}}{\Rightarrow} s'\beta$:

$$\mathtt{mnd}(e,e'\!:\!\gamma,s) \overset{\text{def}}{=} \begin{cases} \Phi_{\Downarrow}(e,e'\!:\!\gamma,s) & (\beta \text{ total}) \\ \Phi_{\Downarrow}(e,e'\!:\!\gamma,s) \wedge \Phi_{\Uparrow}(e,e'\!:\!\gamma,s) & (\beta \text{ partial}) \end{cases}$$

where $\Phi_{\Downarrow}(e,e'\!:\!\alpha \overset{\vec{y}}{\Rightarrow} s'\beta, s)$ is given as:

$$\begin{array}{l} \forall xx'zz'\vec{v}\vec{v}'\vec{w}\vec{w}'.\ ((\,\mathtt{NI}(xx'\vec{v}\vec{v}')\ \wedge\ \mathtt{mnd}(x\vec{v},x'\vec{v}',s)\ \wedge \\ \quad \{!\vec{y}=\vec{v}\}\,e\bullet x \searrow u\{u=z \wedge !\vec{y}=\vec{w}\}\ \wedge \\ \quad \{!\vec{y}=\vec{v}'\}\,e'\bullet x' \searrow u\{u=z' \wedge !\vec{y}=\vec{w}'\}\ ) \supset \mathtt{mnd}(z\vec{w},z'\vec{w}'\!:\!\beta\vec{\gamma},s)\,) \end{array}$$

while $\Phi_{\Uparrow}(e,e'\!:\!\alpha \overset{\vec{y}}{\Rightarrow} \beta, s)$, with β partial, is given as:

$$\begin{array}{l} \forall xx'\vec{v}\vec{v}'.\ ((\,\mathtt{NI}(xx'\vec{v}\vec{v}')\ \wedge\ \mathtt{mnd}(x\vec{v},x'\vec{v}',s)\,) \quad \supset \\ \qquad\qquad \{!\vec{y}=\vec{v}\}\,e\bullet x\Uparrow\ \equiv\ \{!\vec{y}=\vec{v}'\}\,e'\bullet x'\Uparrow\,) \end{array}$$

We can now present a logical translation of the secrecy typing. The typing uses two sequents, $\Gamma \vdash_{\mathsf{sec}} M : \alpha$ and $\Gamma \vdash_{\mathsf{sec}} M : \alpha,\varphi$. The former is translated precisely as before. For the latter, let $\Delta = \Gamma \upharpoonright \mathsf{dom}_f(\Gamma)$ and $\Theta = \Gamma \upharpoonright \varphi$. Then, if α is total, $\Gamma \vdash_{\mathsf{sec}} M : \alpha,\varphi$ is translated into:

> *For each s, whenever $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ and $\mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)$, if $\langle M\xi_1,\sigma_1\rangle \Downarrow \langle V_1,\sigma_1'\rangle$ and $\langle M\xi_2,\sigma_2\rangle \Downarrow \langle V_2,\sigma_2'\rangle$ then $\mathtt{mnd}'(V_1,V_2 : \alpha, s)$ and $\mathtt{mnd}'(\sigma_1',\sigma_2' : \Theta, s)$.*

If β is partial, we translate $\Gamma \vdash_{\mathsf{sec}} M : \alpha,\varphi$ into the conjunction of the above statement and the following one:

> *For each s, whenever $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ and $\mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)$, if $\mathsf{protect}(\beta) \sqsubseteq s$, then $\langle M\xi_1,\sigma_1\rangle \Uparrow$ iff $\langle M\xi_2,\sigma_2\rangle \Uparrow$.*

The translation simply increments the one for stateless behaviour with writing effects, saying: *if* the input vary up to s, *then* their effects, in convergence/divergence and the return value/variable content, should only vary up to s.

Remark 1 We can easily observe these translation of secrecy typability is simply a different way of saying the same thing as the mnd-predicate for function types (assuming all free variables are λ-abstracted). Since any program can be transparently closed using a sequence of λ-abstractions without changing its essential meaning, we can in fact do without this translation, working directly with mnd-predicates and closed programs. Importantly this also enables a purely syntactic proofs which use an alternative proof system which we believe to be equivalent to the proof system we presented in the main section. However in this appendix we work with the present shape of translations, because it is more intuitive and because the length of the noninterference proof itself is not too different.

Below we refine/extend Lemmas A and B (Definition A remains the same except for the obvious incorporation of stores). In (5, 6) below, $\beta_1..\beta_n \Rightarrow \alpha$ stands for $\beta_1 \overset{\emptyset}{\Rightarrow} (..\beta_n \overset{\emptyset}{\Rightarrow} \alpha)..)$.

Lemma 12 *Assume Θ (resp. Γ) uses reference types (resp. non-reference types) as its co-domain.*

(1) $\Gamma, x\!:\!\alpha, \Theta \models_{\mathsf{sec}} M : \beta,\varphi$ *iff* $\Gamma,\Theta \models_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \overset{\varphi}{\Rightarrow} \beta$.

(2) $\Theta \models_{\mathsf{sec}} M : \alpha$ *iff* $\Theta \models M :_u \mathtt{NI}(u\!:\!\alpha)$. *If* $\Theta \models_{\mathsf{sec}} M : \alpha,\varphi$, $\Theta \models_{\mathsf{sec}} \sigma :$ *and* $\langle M,\sigma\rangle \Downarrow \langle V,\sigma'\rangle$, *then* $\Theta \models V :_u \mathtt{NI}(u\!:\!\alpha)$.

(3) If $\Gamma,\Theta \models_{\mathsf{sec}} M : \alpha$ *and* $\Gamma,\Theta \models_{\mathsf{sec}} \xi$ *then we have:* $\Theta \models M\xi :_u \mathtt{NI}(u:\alpha)$. *Similarly if* $\Gamma,\Theta \models_{\mathsf{sec}} M : \alpha,\varphi$, $\Gamma,\Theta \models_{\mathsf{sec}} \xi$, $\Theta \models_{\mathsf{sec}} \sigma$ *and* $\langle M\xi,\sigma\rangle \Downarrow \langle V,\sigma'\rangle$, *then we have:* $\Theta \models V :_u \mathtt{NI}(u\!:\!\alpha)$.

(4) $\mathtt{NI}(u\!:\!\alpha)$ *is admissible at u for each α.*

(5) If $\Theta, \vec{y}\!:\!\vec{\beta}, x\!:\!\alpha \models_{\mathsf{sec}} V : \gamma$ *then we have:* $\Theta, \vec{y}\!:\!\vec{\beta}, z\!:\!\vec{\beta} \Rightarrow \alpha \models_{\mathsf{sec}} V[z\bullet\vec{y}/x] : \gamma$.

(6) $\models_{\mathsf{sec}} \mu z.\lambda\vec{y}.V[z\bullet\vec{y}/x] : \vec{\beta} \Rightarrow \alpha\ \supset\ \models_{\mathsf{sec}} \lambda\vec{y}.\mu x.V : \vec{\beta} \Rightarrow \alpha$.

All proofs are easy. (5) is justified via the contextual congruence between two programs, using their process encodings and replication theorems.

Finally we present the secrecy typing rules in Figure 3. (Var), (Op), (Rec), (Promote), (Inj), (Pair), (Proj), (Let) and (Weak) stay as in Figure 1 of the main paper.

E.4 Global State (2): Proof of Noninterference

Theorem 4 $\Gamma \vdash_{\mathsf{sec}} M : \alpha$ *implies* $\Gamma \models_{\mathsf{sec}} M : \alpha$. *Similarly* $\Gamma \vdash_{\mathsf{sec}} M : \alpha,\varphi$ *implies* $\Gamma \models_{\mathsf{sec}} M : \alpha,\varphi$.

PROOF: By rule induction. **(Var, Op, Abs, Rec, Pair, Proj, Inj, Lift)** precisely follow those in Theorem 3, using the corresponding statements in Lemma C. Below metavariables for types etc. are from each secrecy typing rule. Throughout we let $\Delta \overset{\text{def}}{=} \Gamma \upharpoonright \mathsf{dom}_f(\Gamma)$ and $\Theta \overset{\text{def}}{=} \Gamma \upharpoonright \varphi$.

(App) Assume β is partial (which subsumes the total case). $\mathtt{mnd}'(\xi_1,\xi_2\!:\!\Delta,s) \wedge \mathtt{mnd}'(\sigma_1,\sigma_2\!:\!\Theta,s)$

$$[Abs]\ \frac{\Gamma, x:\alpha \vdash_{\mathsf{sec}} M:\beta,\varphi \quad s \sqsubseteq \mathsf{protectE}(\Gamma \upharpoonright \varphi)}{\Gamma \vdash_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \overset{\varphi}{\Rightarrow}_s \beta}$$

$$[App]\ \frac{\Gamma \vdash V:\alpha \overset{\varphi}{\Rightarrow}_s \beta \quad \Gamma \vdash_{\mathsf{sec}} W:\alpha}{\Gamma \vdash_{\mathsf{sec}} VW:\beta,\varphi}$$

$$[If]\ \frac{\Gamma \vdash_{\mathsf{sec}} V:\mathsf{Bool}_s \quad \Gamma \vdash_{\mathsf{sec}} N_{1,2}:\alpha,\varphi \quad s \sqsubseteq \mathsf{protect}(\alpha) \sqcap \mathsf{protectE}(\Gamma \upharpoonright \varphi)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{if}\ V\ \mathtt{then}\ N_1\ \mathtt{else}\ N_2:\alpha}$$

$$[Case]\ \frac{\Gamma \vdash_{\mathsf{sec}} V:\alpha_1 +_s \alpha_2 \quad \Gamma, x_i:\alpha_i \vdash_{\mathsf{sec}} M_i:\beta,\varphi \quad s \sqsubseteq \mathsf{protect}(\beta) \sqcap \mathsf{protectE}(\Gamma \upharpoonright \varphi)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i^{\alpha_i}).M_i\}_{i\in\{1,2\}}:\beta,\varphi}$$

$$[Seq]\ \frac{\Gamma \vdash_{\mathsf{sec}} N:\llcorner\alpha\lrcorner_s,\varphi_1 \quad \Gamma, x:\alpha \vdash_{\mathsf{sec}} M:\beta,\varphi_2 \quad \beta \text{ partial} \quad s \sqsubseteq \mathsf{protect}(\beta) \sqcap \mathsf{protectE}(\Gamma \upharpoonright \varphi_2)}{\Gamma \vdash_{\mathsf{sec}} \mathtt{seq}\ x = N\ \mathtt{in}\ M:\beta,\varphi_1 \cup \varphi_2}$$

$$[Deref]\ \frac{\Gamma \vdash_{\mathsf{sec}} x:\mathsf{Refr}_s(\alpha)}{\Gamma \vdash_{\mathsf{sec}} !x:\alpha,\{x\}} \qquad [Assign]\ \frac{\Gamma \vdash_{\mathsf{sec}} V:\alpha \quad \Gamma \vdash_{\mathsf{sec}} x:\mathsf{Ref}_s(\alpha)}{\Gamma \vdash_{\mathsf{sec}} x:=V:\alpha,\{x\}}$$

$$[Weak\text{-}write]\ \frac{\Gamma, x:\mathsf{Refr}_s(\alpha) \vdash_{\mathsf{sec}} M:\alpha,\varphi}{\Gamma, x:\mathsf{Ref}_s(\alpha) \vdash_{\mathsf{sec}} M:\alpha,\varphi}$$

Figure 3: Secrecy Typing Rules with Global State

$$\begin{array}{lll}
\supset & \mathtt{mnd}(V\xi_1, V\xi_2 : \alpha \overset{\varphi}{\Rightarrow} \beta, s) \ \wedge & \\
& \mathtt{mnd}'(W\xi_1, W\xi_2 : \alpha, s) & \text{(IH, Lem. C (2))} \\
\supset & \wedge_{i=1,2}\langle (VW)\xi_i, \sigma_i\rangle \Uparrow \quad \vee & \\
& (\ \wedge_{i=1,2}\langle (VW)\xi_i, \sigma_i\rangle \Downarrow \langle U_i, \sigma'_i\rangle \ \wedge & \\
& \quad \mathtt{mnd}(U_1, U_2 : \beta, s) \ \wedge & \\
& \quad \mathtt{mnd}(\sigma'_1, \sigma'_2 : \Theta, s)\ ) & \text{(Def).}
\end{array}$$

(If) Subsumed by **(Case)** below.

(Case) Let $N \overset{\text{def}}{=} \mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i^{\alpha_i}).M_i\}_{i\in\{1,2\}}$, $\gamma = \alpha_1 +_{s'} \alpha_2$ and β be partial (subsuming the total case). If $s' \sqsubseteq s$:

$$\begin{array}{lll}
\multicolumn{2}{l}{\mathtt{mnd}'(\xi_1, \xi_2 : \Delta, s) \ \wedge \ \mathtt{mnd}'(\sigma_1, \sigma_2 : \Theta, s)} & \\
\supset & \mathtt{mnd}'(V\sigma_1, V\sigma_2 : \gamma, s) & \text{(IH, Lem. A (3))} \\
\supset & \wedge_{i=1,2}\ V\sigma_i \cong \mathtt{inj}_j(W_i) \ \wedge & \\
& \mathtt{mnd}'(W_1, W_2 : \alpha_j, s) & (s' \sqsubseteq s) \\
\supset & (\wedge_{i=1,2}\ \langle M_j\sigma_i[W_i/x_j], \sigma_i\rangle \Downarrow \langle U_i, \sigma'_i\rangle \ \wedge & \\
& \mathtt{mnd}(U_1, U_2 : \beta, s) \ \wedge \ \mathtt{mnd}(\sigma'_{1j}, \sigma'_{2j} : \Theta, s)) \ \vee & \\
& \wedge_{i=1,2}\ \langle M_j\sigma_i[W_i/x_j], \sigma_i\rangle \Uparrow & \text{(IH)}
\end{array}$$

with j being either 1 or 2. If $s' \not\sqsubseteq s$, we have $\mathsf{protect}(\beta) \not\sqsubseteq s$ and $\mathsf{protectE}(\Gamma \upharpoonright \varphi) \not\sqsubseteq s$ from the secrecy condition, hence neither convergence/divergence nor, when both converge, any state change is s-distinguishable (by structural security).

(Let) Subsumed by **(Seq)** below.

(Seq) Let $L \overset{\text{def}}{=} \mathtt{seq}\ x = N\ \mathtt{in}\ M$ and $\gamma \overset{\text{def}}{=} \llcorner\alpha\lrcorner_{s'}$ (note s' replaces s in the typing rule). The secrecy condition becomes $s' \sqsubseteq \mathsf{protect}(\beta)$. First, if $s' \sqsubseteq s$, we infer:

$$\begin{array}{lll}
\multicolumn{2}{l}{\mathtt{mnd}'(\xi_1, \xi_2 : \Delta, s) \ \wedge \ \mathtt{mnd}'(\sigma_1, \sigma_2 : \Theta, s)} & \\
\supset & \mathtt{mnd}(N\sigma_1, N\sigma_2 : \gamma, s) & \text{(IH, Lem. A (3))} \\
\supset & \mathtt{mnd}(N\sigma_1, N\sigma_2 : \alpha, s) \ \vee \ (N\sigma_1 \Uparrow \wedge N\sigma_2 \Uparrow) & \text{(Def)} \\
\supset & (\ \wedge_{i=1,2}\langle M\xi_i[N\xi_i/x], \sigma_i\rangle \Downarrow \langle V_i, \sigma'_i\rangle \ \wedge & \\
& \ \mathtt{mnd}(V_1, V_2 : \alpha, s) \ \wedge \ \mathtt{mnd}(\sigma'_1, \sigma'_2 : \Delta, s)\ ) \ \vee & \\
& \wedge_{i=1,2} L\sigma_i \Uparrow & \text{(IH)}
\end{array}$$

The case $s' \not\sqsubseteq s$ is identical as **Case** above.

(Deref) Direct from the structural security.

(Assign) $\mathtt{mnd}'(\xi_1, \xi_2 : \Delta, s) \wedge \mathtt{mnd}'(\sigma_1, \sigma_2 : \Theta, s)$

$$\begin{array}{lll}
\supset & \mathtt{mnd}(V\xi_1, V\xi_2 : \alpha, s) & \text{(IH)} \\
\supset & \mathtt{mnd}(\sigma_1[x \mapsto V\xi_1], \sigma_2[x \mapsto V\xi_2] : \alpha, s) &
\end{array}$$

(Weak, Weak-write) Each immediate from the induction hypothesis. We have now exhausted all cases. □

E.5 Local State (1): Preparation

To incorporate local state into secrecy typing, we first refine the arrow type to record the secrecy level, adding:

$$\alpha \overset{\varphi}{\Rightarrow}_s \beta$$

The level s in $\alpha \overset{\varphi}{\Rightarrow}_s \beta$ says when this closure is invoked, it may affect the environment at the level as low as s. We also retain the purely function arrow type $\alpha \Rightarrow \beta$ for a technical convenience, for which the logic can assert using the application operator, $e \bullet e'$.[1]

The contravariant protection levels are now defined in their general form (the definition of $\mathsf{protect}(\alpha)$ stays the same except the refined arrow type, given below).

$$\begin{array}{rcl}
\mathsf{protect}(\alpha \overset{\varphi}{\Rightarrow}_s \beta) & = & \mathsf{protectE}(\alpha) \sqcap \mathsf{protect}(\beta) \\
\mathsf{protectE}(\alpha \overset{\varphi}{\Rightarrow}_s \beta) & = & s \sqcap \mathsf{protect}(\alpha) \sqcap \mathsf{protectE}(\beta) \\
\mathsf{protectE}(\mathsf{Nat}_s) & = & \mathsf{protectE}(\mathsf{Bool}_s) \ = \ \mathrm{H} \\
\mathsf{protectE}(\alpha \times \beta) & = & \mathsf{protectE}(\alpha) \sqcap \mathsf{protectE}(\beta) \\
\mathsf{protectE}(\alpha +_s \beta) & = & \mathsf{protectE}(\alpha) \sqcap \mathsf{protectE}(\beta) \\
\mathsf{protectE}(\llcorner\alpha\lrcorner_s) & = & \mathsf{protectE}(\alpha) \\
\mathsf{protectE}(\mathsf{Ref}_s(\alpha)) & = & s \sqcap \mathsf{protectE}(\alpha) \sqcap \mathsf{protect}(\alpha) \\
\mathsf{protectE}(\mathsf{Refr}_s(\alpha)) & = & \mathsf{protectE}(\alpha)
\end{array}$$

Remark 2 *Regarding $\alpha \overset{\varphi}{\Rightarrow} \beta$ in the system with global state as $\alpha \overset{\varphi}{\Rightarrow}_H \beta$ in the present system, the covariant/contravariant protection levels of each α in the original system coincide with those in the present system.*

We again assume the structural security, which now says:

[1] With global state, $\alpha \Rightarrow \beta$ has a faithful embedding as $\alpha \overset{\emptyset}{\Rightarrow} \beta$. However $\alpha \overset{\emptyset}{\Rightarrow}_s \beta$ in the present system may allow writing at a hidden name.

Definition 5 $\mathsf{Ref}_s(\alpha)$ is *structurally secure* when $s \sqsubseteq \mathsf{protect}(\alpha) \sqcap \mathsf{protectE}(\alpha)$. $\mathsf{Refr}_s(\alpha)$ is *structurally secure* when $s \sqsubseteq \mathsf{protect}(\alpha)$.

The notion conservatively extends the original one (i.e. $\mathsf{Ref}_s(\alpha)$ is structurally secure in the system with global state iff it is so in the system with local state). *From now on, we assume all reference types are structurally secure.*

The secrecy typing rules for existing programs stay precisely the same except the ways to calculate protection levels in $[Case]$, $[If]$ and $[Seq]$ are updated. The "new" is treated by the same rule as the non-secrecy typing.

$$[New]\ \frac{\Gamma \vdash_{\text{sec}} V : \alpha \quad \Gamma, x : \mathsf{Ref}_s(\alpha) \vdash_{\text{sec}} N : \beta, \varphi}{\Gamma \setminus x \vdash_{\text{sec}} \mathtt{new\ var}\ x := V\ \mathtt{in}\ N : \beta \setminus x, \varphi \setminus x}$$

We also use a rule for the purely functional abstraction, which infers $\Gamma \vdash M : \alpha \Rightarrow \beta$ from $\Gamma, x : \alpha \vdash M : \beta$ (note the latter has no effect).

We next augment the predicates for noninterference. Suppose we are given two evaluated programs, M_1 and M_2, named as u_1 and u_2. As discussed in the main section, we first need to open these u_1 and u_2 using opening formulae, then relate u_1 and u_2 as well as a pair (or pairs) of local references.[2] If u_1 and u_2 have arrow types, relating u_1 and u_2 means they act in the same way as far as their local states, global states and input data are related. In the following technical development, the mnd-predicate $\mathtt{mnd}(u_1, u_2 : \alpha, s)$ assumes u_1 and u_2 are already opened, assuming the opening is given beforehand.

By assuming related behaviours are already opened, the only change is in the arrow type. We introduce several notations. Firstly, the predicate:

$$\{C\} e \bullet e' \searrow (\nu \vec{y}) z \{C'\} \tag{41}$$

stands for $\{C\} e \bullet e' \searrow z \{(\nu \vec{y}) z \vec{w}\ \text{in}\ C'\}$ where $\vec{w}$ exhausts all primary names in C'. Secondly, we write

$$\{C\} e \bullet e' \Downarrow \tag{42}$$

for the negation of $\{C\} e \bullet e' \Uparrow$ when the result is a partial type: if not, (42) is set to be T.

We now give the predicate for arrow types, again dividing them into two, one for convergence and another for divergence. For legibility we present the case when there is a single shared (global) effect name, a single local effect name, and a single added local name ($\vec{y}$ in (42) above). By changing each single name to a vector of names, the definition easily extends to the general case. As before, we define two predicates.

1. $\Phi_{\Downarrow}(x_1 : \alpha \overset{y;l_1}{\Rightarrow}_{s'} \beta, x_2 : \alpha \overset{y;l_2}{\Rightarrow}_{s'} \beta, s)$ where β is either total or partial, which says two functions behave in the same way when they terminate.

2. $\Phi_{\Downarrow}(x_1 : \alpha \overset{y;l_1}{\Rightarrow}_{s'} \beta, x_2 : \alpha \overset{y;l_2}{\Rightarrow}_{s'} \beta, s)$ in which β is partial, which says two functions behave in the same way in divergence/convergence.

First we define the former.

$$\begin{array}{l}\forall \vec{u}\vec{v}\vec{w}\vec{t}.\ (\ (\mathtt{mnd}'(u_1 v_1 t_1, u_2 v_2, s) \wedge \bigwedge_{i=1,2}\{!y = u_i \wedge !l_i = t_i\} x_i \bullet v_i \Downarrow) \\ \qquad \supset\ \{!y = u_1 \wedge !l_1 = t_1\} x_1 \bullet v_1 \searrow (\nu z_1) w_1 \\ \qquad\qquad \{\ \{!y = u_2 \wedge !l_2 = t_2\} x_2 \bullet v_2 \searrow (\nu z_2) w_2 \\ \qquad\qquad\quad \{\mathtt{mnd}'(!z_1 !y_1 !l_1 w_1, !z_2 !y_2 !l_2 w_2, s)\}\ \}\ )\end{array}$$

If we add the opening of x_1 and x_2 at the outset (which introduces l_1 and l_2, respectively), and if we further assume z_1 and z_2 are not present, then the above predicate gives what is equivalent to the one given in the main section. We observe:

1. Either y (or its vector generalisation) is total or local, we demand the resulting states, as well as returned values, to be s-related. In addition, e require the new local states to be s-related. This is essential since these local values store the result of information, to influence their external behaviour later.

2. We need nesting of evaluation formulae since we first need to open w_1 and w_2, as well as (possibly) y_1 and y_2, the latter now possibly holding, for example, these returned values.

3. Since types of hidden names are opaque, the shape of the predicate cannot be uniquely determine by a given arrow type. However it suffices that a closed program satisfies any predicate of the shape given below.

We next move to the second predicate, which is for divergence, written $\Phi_{\Downarrow}(x_1 : \alpha \overset{y;l_1}{\Rightarrow}_{s'} \beta, x_2 : \alpha \overset{y;l_2}{\Rightarrow}_{s'} \beta, s)$.

$$\begin{array}{l}\forall \vec{t}\vec{u}\vec{v}.\ (\ \mathtt{mnd}'(t_1 u_1 v_1, t_2 u_2 v_2, s)\ \wedge\ s' \sqsubseteq s \\ \qquad \supset\ (\{!y = u_1 \wedge !l_1 = t_1\} x_1 \bullet v_1 \Uparrow \\ \qquad\qquad \equiv\ \{!y = u_2 \wedge !l_2 = t_2\} x_2 \bullet v_2 \Uparrow)\ )\end{array}$$

which says the convergence behaviours should coincide as far as the information level of the convergence (s' in the above case) is visible from the given secrecy level (s in the above case).

Finally we present the translation of the typability in secrecy typing. The only essential change is the need to prefix each MND-predicates with opening formulae. We introduce a convenient notation. Assuming $M_i \stackrel{\text{def}}{=} \mathtt{new\ var}\ \vec{y}_i := \vec{W}_i\ \mathtt{in}\ V_i$ $(i = 1, 2)$ (closed programs of the same type), we write $\mathtt{mnd}(M_1, M_2 : \alpha, s)$ for $\models M_1, M_2 :_{x_1, x_2} A$ where:

$$A \stackrel{\text{def}}{=} (\nu \vec{y}_1) x_1\ \text{in}\ (\nu \vec{y}_2) x_2\ \text{in}\ \mathtt{mnd}(!\vec{y}_1 x_1, !\vec{y}_2 x_2, s) \tag{43}$$

[2] Here it is assumed that the local references hidden by u_1 and those by u_2 are in precise correspondence with each other, in both number and types. This does not lose generality because of the the closure of semantics of formulae by $\cong$. In particular these local names may not directly correspond to the syntactic structure of programs.

with $!\vec{x} \stackrel{\text{def}}{=} !x_1..!x_n$ when $\vec{x} = x_1..x_n$. We further extend this predicate to a pair of the configurations which are results of converged evaluations, $C_i \stackrel{\text{def}}{=} (\nu\vec{y}_i)(V_i,\sigma_i)$ $(i = 1,2)$, in the obvious way, writing $\mathtt{mnd}(C_1, C_2 : \alpha, \Theta, s)$ where α is the type of the value part while Θ is the typing of the exposed reference names.[3] Now let $\Delta = \Gamma \restriction \mathsf{dom}_f(\Gamma)$. First, the sequent $\Gamma \vdash_{\mathsf{sec}} M : \alpha$ is translated as follows.

> *For each s, if we have* $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ *then* $\mathtt{mnd}'(M\xi_1, M\xi_2 : \alpha, s)$.

Next, the sequent $\Gamma \vdash_{\mathsf{sec}} M : \alpha, \varphi$ with α being total, is translated as follows. Let $\Theta = \Gamma \restriction \varphi$ below.

> *For each s, if we have* $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ *and* $\mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)$*, and, further, if* $\langle M\xi_1,\sigma_1\rangle \Downarrow C_1$ *and* $\langle M\xi_2,\sigma_2\rangle \Downarrow C_2$*, then* $\mathtt{mnd}(C_1, C_2 : \alpha, \Theta, s)$.

If β is partial, we translate $\Gamma \vdash_{\mathsf{sec}} M : \alpha, \varphi$ into the conjunction of the above statement and the following one:

> *For each s, whenever* $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ *and* $\mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)$*, if* $\mathsf{protect}(\beta) \sqsubseteq s$*, then* $\langle M\xi_1,\sigma_1\rangle \Uparrow$ *iff* $\langle M\xi_2,\sigma_2\rangle \Uparrow$.

Note the translation simply increments the one for global state, adding one opening formula at the initial state and another at the final state.

We restate the basic lemma. Below, as before, we write $\alpha_1..\alpha_n \Rightarrow \beta$ for $\alpha_1 \Rightarrow (\alpha_2 \Rightarrow ..(\alpha_n \Rightarrow \beta)..)$. While we believe (6) comes from categorical studies such as [3, 8], it is also a consequence of the equality over the translation of concerned programs into the π-calculus (which essentially only relies on the replication theorems as found in [7]).

Lemma 13 *Assume Θ (resp. Γ) uses only reference types (resp. non-reference types) as its co-domain.*

(1) $\Gamma, x{:}\alpha, \Theta \models_{\mathsf{sec}} M : \beta, \varphi$ *iff* $\Gamma, \Theta \models_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \stackrel{\varphi}{\Rightarrow}_s \beta$. *Similarly,* $\Gamma, x{:}\alpha, \Theta \models_{\mathsf{sec}} M : \beta$ *iff* $\Gamma, \Theta \models_{\mathsf{sec}} \lambda x^{\alpha}.M : \alpha \Rightarrow \beta$.

(2) $\Theta \models_{\mathsf{sec}} M : \alpha$ *iff* $\Theta \models M :_u \mathtt{NI}(u : \alpha)$. *If* $\Theta \models_{\mathsf{sec}} M : \alpha, \varphi$, $\Theta \models_{\mathsf{sec}} \sigma :$ *and* $\langle M,\sigma\rangle \Downarrow C$*, then* $\Theta \models C :_u \mathtt{NI}(u : \alpha)$.

(3) If $\Gamma, \Theta \models_{\mathsf{sec}} M : \alpha$ *and* $\Gamma, \Theta \models_{\mathsf{sec}} \xi$ *then we have:* $\Theta \models M\xi :_u \mathtt{NI}(u : \alpha)$. *Similarly if* $\Gamma, \Theta \models_{\mathsf{sec}} M : \alpha, \varphi$, $\Gamma, \Theta \models_{\mathsf{sec}} \xi$, $\Theta \models_{\mathsf{sec}} \sigma$ *and* $\langle M\xi,\sigma\rangle \Downarrow C$*, then we have:* $\Theta \models C :_u \mathtt{NI}(u : \alpha)$.

(4) $\mathtt{NI}(u : \alpha)$ *is admissible at u for each* α.

(5) If $\Theta, \vec{y}{:}\vec{\beta}, x{:}\alpha \models_{\mathsf{sec}} V : \gamma$ *then we have:* $\Theta, \vec{y}{:}\vec{\beta}, z{:}\vec{\beta} \Rightarrow \alpha \models_{\mathsf{sec}} V[z \bullet \vec{y}/x] : \gamma$.

(6) $\models_{\mathsf{sec}} \mu z.\lambda\vec{y}.V[z \bullet \vec{y}/x] : \vec{\beta} \stackrel{\varphi}{\Rightarrow}_s \alpha \supset \models_{\mathsf{sec}} \lambda\vec{y}.\mu x.V : \vec{\beta} \stackrel{\varphi}{\Rightarrow}_s \alpha$.

[3] Formally, given $(\nu\vec{y})(V, \sigma\cdot\sigma')$ where $\mathsf{dom}(\sigma) = \{\vec{y}\}$ and $\mathsf{dom}(\sigma') = \{\vec{w}\}$, we regard this term as $\mathtt{new\ var}\ \vec{y} := \sigma(\vec{y})\ \mathtt{in}\ (V, \sigma'(\vec{w}))$ where $(V, \sigma'(\vec{w}))$ is an n-ary pairing. This recovers the shape needed for defining the formula in (43).

Proposition 8 *1. Let Θ has only reference types as its codomain,* $\Theta \models_{\mathsf{sec}} M :_{1,2} : \alpha, \varphi$ *and* $\Theta' = \Theta \restriction \varphi$*. Further assume* $\mathsf{protect}(\alpha) \sqcap \mathsf{protectE}(\Theta') \not\sqsubseteq s$*. Then* $\mathtt{mnd}'(\sigma_1, \sigma_2 : \Theta', s)$ *and* $(M_i,\sigma_i) \Downarrow C_i$ $(i = 1,2)$ *imply* $\mathtt{mnd}'(C_1, C_2 : \alpha, \Theta', s)$.

2. *If we have* $\mathtt{mnd}(C_1, C_2 : \alpha, \Theta{\cdot}x : \mathsf{Ref}_s(\beta), s)$*, then we also have* $\mathtt{mnd}((\nu x)C_1, (\nu x)C_2 : \alpha/x, \Theta/x, s)$.

PROOF: For (1), because $\mathsf{protect}(\alpha) \sqcap \mathsf{protectE}(\Theta') \not\sqsubseteq s$ means each of $M_{1,2}$ never writes to a reference visible from s directly or indirectly (including hidden states), we know all references visible from s are never written. Hence, for C_1 and C_2 named as x_1 and x_2, we can open x_1 and x_2 enumerating all their hidden references and s-relate them (by their s-invisibility by programs not writing at s or lower), as well as the return values themselves (for he latter formally by induction on types). (2) is immediate from the form of the mnd-predicate. □

E.6 Local State (2): Proof of Noninterference

Theorem 5 $\Gamma \vdash_{\mathsf{sec}} M : \alpha$ *implies* $\Gamma \models_{\mathsf{sec}} M : \alpha$*. Similarly* $\Gamma \vdash_{\mathsf{sec}} M : \alpha, \varphi$ *implies* $\Gamma \models_{\mathsf{sec}} M : \alpha, \varphi$.

PROOF: By rule induction. All cases are essentially the same as in the case of global state except "hidden secrecy levels" (recorded in arrow types) are taken into consideration. We show the reasoning for two typical cases.

(Case) Let $N \stackrel{\text{def}}{=} \mathtt{case}\ V\ \mathtt{of}\ \{\mathtt{inj}_i(x_i^{\alpha_i}).M_i\}_{i\in\{1,2\}}$ which is typed as β under Γ. Also let $\gamma = \alpha_1 +_{s'} \alpha_2$. For simplicity below we assume β is total (the partial case is dealt with by incorporating divergence in the now familiar way). First, if $s' \sqsubseteq s$:

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s) \wedge \mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)} \\
\supset & \mathtt{mnd}'(V\sigma_1, V\sigma_2 : \gamma, s) & \text{(IH, Lem. A (3))} \\
\supset & \wedge_{i=1,2}\ V\sigma_i \cong \mathtt{inj}_j(W_i)\ \wedge & \\
 & \mathtt{mnd}'(W_1, W_2 : \alpha_j, s) & (s' \sqsubseteq s) \\
\supset & (\wedge_{i=1,2}\ \langle M_j\sigma_i[W_i/x_j], \sigma_i\rangle \Downarrow C_i\ \wedge & \\
 & \mathtt{mnd}(C_1, C_2 : \beta, \Theta, s) & \text{(IH)}
\end{array}
$$

with j being either 1 or 2. On the other hand, if $s' \not\sqsubseteq s$, we have (a) $\mathsf{protect}(\beta) \not\sqsubseteq s$ and (b) $\mathsf{protectE}(\Gamma \restriction \varphi) \not\sqsubseteq s$, by the secrecy condition. From (a) and (b) we can use Proposition 8 (1), hence as required.

(New) Assume $\Gamma \models_{\mathsf{sec}} V : \alpha$ and $\Gamma, x : \mathsf{Ref}_s(\alpha) \models_{\mathsf{sec}} N :: \beta, \varphi$. For simplicity let α be total. We show $\Gamma \setminus x \models_{\mathsf{sec}} \mathtt{new\ var}\ x := V\ \mathtt{in}\ N : \beta \setminus x, \varphi \setminus x$. Let $\Delta = \Gamma \restriction \mathsf{dom}_f(\Gamma)$ and $\Theta = \Gamma \restriction \varphi$. Further let $\mathtt{mnd}'(\xi_1,\xi_2 : \Delta, s)$ and $\mathtt{mnd}'(\sigma_1,\sigma_2 : \Theta, s)$. Now set $M \stackrel{\text{def}}{=} \mathtt{new\ var}\ x := V\ \mathtt{in}\ N$. Then evaluating $(\nu x)(M\xi_i, \sigma_i \cdot x \mapsto V)$ $(i = 1,2)$ is the same thing as evaluating $(M\xi_i, \sigma_i \cdot x \mapsto V)$ $(i = 1,2)$. Since we have $\mathtt{mnd}'(V, V : \alpha, s)$ for each s by the first assumption, if $(M\xi_i, \sigma_i \cdot x \mapsto V) \Downarrow C_i$ $(i = 1,2)$ then we have $\mathtt{mnd}(C_1, C_2 : \alpha, \Theta,)$. By Proposition 8 (2) we are done. □

E.7 Extensions and Refinement

The representation of the secrecy property using logical predicates and associated noninterference proofs can be refined in several directions. First of all, as we noted in Remark 1, §E.3 in this appendix, we believe all proofs including those for local state can be carried out syntactically, using an alternative proof system. While this does not preclude the use of validity in the proofs, this will lead to a further simplification of the non-interference proofs.

As further extensions and refinements, we may consider:

(A) Treatment of convergence.

(B) Extensions to second-order polymorphism and recursive types.

(C) Treatment of secrecy policies with explicitly secrecy levels [2, 6, 10] as well as other secrecy policies [1].

Of these, (A) indicates the choice to disregard convergence/divergence as information emittance. The use of pointed types in our development, however, has already incorporated it, since by raising the level of pointed types to the (unreachable) high level, we can always disregard convergence.

For (B), we believe extensions to second-order polymorphism, including existential and universal types, can be soundly done. This is mainly because we are working with second-order polymorphisms at the level of syntax (a recent method by Vouillon and Mellies may be usable to give a behavioural representation of these types). For recursive types, we cannot use the inductive construction as we have developed so far. Can we logically represent a non-interference property for a program with recursive types finitely? This may be possible for a concrete datum, at least if it does not contain destructive update. Another method is introduction of recursion at the level of formulae, as in modal μ-calculi.

For (C), we believe the policy with local state as discussed in the previous section is in fact obeyed by the type disciplines in [2, 6, 10]. The dependency core calculus [1] has introduced a way to allow global secrecy while violating local secrecy. This can be easily incorporated as a refinement of our predicates, by raising the levels of associated types up to the level of the final protection level.

References

[1] Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *Proc. 26th POPL*. ACM Press, 1999.

[2] François Pottier and Vincent Simonet. Information flow inference for ML. *ACM Trans. Prog. Lang. and Sys.*, 25(1):117–158, January 2003.

[3] Masahito Hasegawa. *Models of Sharing Graphs: A Categorical Semantics of let and letrec*. Distinguished Dissertation Series. Springer-Verlag, 1999.

[4] Kohei Honda and Nobuko Yoshida. A uniform type structure for secure information flow. In *POPL'02*, pages 81–92. ACM Press, 2002. Full version available at www.doc.ic.ac.uk/˜yoshida, Revised in June 2004, 84 pages.

[5] Kohei Honda and Nobuko Yoshida. A compositional logic for polymorphic higher-order functions. In *Proc. PPDP'04*, 2004. To appear.

[6] Andrew C. Myers. Jflow: practical mostly-static information flow control. In *Proc. 26th POPL*, 1999.

[7] B. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *MSCS*, 6(5):409–454, 1996.

[8] Alex Simpson and Gordon Plotkin. Complete Axioms for Categorical Fixed-Ppoint Operators. In *Proc. 15th LiCS*, 2000.

[9] Scott Smith. From Operational to Denotational Semantics. In *Proc. Conference on Mathematical Foundations of Programming Language Semantics*, volume 598 of *LNCS*, 1992.

[10] Lantian Zheng, Stephen Chong, Andrew C. Myers, and Steve Zdancewic. Using replication and partitioning to build secure distributed systems. In *Proc. IEEE Symposium on Security and Privacy*, 2003.