Contents

Glossary

MS Memory Safety. 1, 12

sCCT Strict Cryptographic Constant Time. 1, 13, 14, 22

SMS Spatial Memory Safety. 1, 10–12, 22, 103, 104

SS Speculation Safety. 1, 15

TMS Temporal Memory Safety. 1, 8, 9, 12, 26, 30, 32, 33, 37–43, 51, 85, 87, 100, 103, 104

1 Preliminaries

In comparison to the paper, some notation is different. This document should be considered work-in-progress, we want to make sure both the report and the paper use similar notation. A non-exhaustive description of differences is: compilers are not γ, but $[\![\bullet]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$. Trace relations are written $\approx^{*}_{\delta;\mathsf{X}}$ or $\theta\,(\bullet)$.

Definition 1 (Events)**.** *$E \cup \{\lightning\}$ is the set of atomic propositions hereby called events or actions. The internal action is ε. The action $\lightning$ is program termination.*

We assume that any programming language can be enriched with a self-composition operator in the style of [1]. Furthermore, we also assume the existence of a low-equivalence relation that distinguishes program states only by their public memory. Two traces $\overline{a}_1$,$\overline{a}_2$ are low-equivalent $\overline{a}_1 =_L \overline{a}_2$ iff all their public events coincide.

Definition 2 (Programming Languages)**.** *A programming language is a tuple $\left(\mathcal{P}, \vdash_W, \overset{e}{\hookrightarrow}, \bowtie\right)$ s.t.:*

- $\mathcal{P}$ *: Set - is a set of admissible, partial programs.*
- $\vdash_W$ *: $\mathcal{P}$ - a judgement that holds iff a program is not partial.*
- $\overset{e}{\hookrightarrow}$ *: $\mathcal{W} \to E \cup \{\lightning\} \to \mathcal{W}$ - a step relation, where $\mathcal{W} = \{w \in \mathcal{P} \mid \ \vdash_W w\}$. For $e \in E \cup \{\lightning\}$ and $p, p' \in \mathcal{W}$ we say for $p \overset{e}{\hookrightarrow} p'$ that program p performs a step with action e to program p'. If $e = \varepsilon$, we write $p \hookrightarrow p'$. In case $e = \lightning$, we write $p \Downarrow$.*
- $\bowtie$ *: $\mathcal{P} \to \mathcal{P} \to \mathcal{P}$ - links two partial programs together in some way, resulting in a new partial program.*

Let S, I, and $\mathbf{T}$ be any programming language.

Definition 3 (Notation for Sequences)**.** *For any sequence of events, let $\cdot$ denote the empty sequence and $e :: \overline{t}$ the sequence that starts with e and continues with $\overline{t}$. Hereby, it does not matter whether $\overline{t}$ is finite or infinite, it's merely syntactic sugar to work on sequences of events.*

Definition 4 (Traces)**.** *A trace $\overline{a}$ is an infinite sequence of events that results from the relation $\overset{e}{\hookrightarrow}$. That is, we obtain the trace $\overline{a} = e_0 :: e_1 :: \ldots$ for the execution sequence $p \overset{e_0}{\hookrightarrow} p' \overset{e_1}{\hookrightarrow} \ldots$ and write $p \rightsquigarrow \overline{a}$. The set of all traces is Traces.*

We assume $\lightning$ to occur in traces representing terminating programs such that it occurs infinitely often in a one-by-one sequence.

Definition 5 (Once)**.** *Given an event a and a trace $\overline{a}$, we say a appears exactly once in the trace, written $a \in_! \overline{a}$, iff*

If $n \in \mathbb{N}$ such that $\overline{a}[n] = a$, then for any $m \in \mathbb{N}$ such that $\overline{a}[m] = a$ we have $n = m$.

Definition 6 (Before)**.** *Given two events a_0, a_1 and a trace $\overline{a}$, event a_0 occurs before a_1 in $\overline{a}$, written $a_0 \leq_{\overline{a}} a_1$, iff*

If $n \in \mathbb{N}$ such that $\overline{a}[n] = a_0$, then $\exists m \in \mathbb{N}, \overline{a}[m] = a_1 \wedge n < m$

Definition 7 (Finite Trace Prefixes)**.** *A finite sequence of events m is a finite trace prefix of $\overline{a}$ iff it satisfies the following judgement.*

$$\frac{}{\cdot \leq \overline{a}} \qquad \frac{m \leq \overline{a}}{e :: m \leq e :: \overline{a}}$$

Definition 8 (Behavior)**.** *The behavior of a whole program p is a set of all traces it produces, i.e. $Behav(p) = \{\overline{a} \mid p \rightsquigarrow \overline{a}\}$.*

Definition 9 (Observation)**.** *An observation is a finite set of finite trace prefixes. We say that an observation o is the prefix of a behavior b iff*

$$\forall m \in o. \exists \overline{a} \in b. m \leq t$$

.

Definition 10 (Properties)**.** *A property π is a set of admissible traces. For a program p to satisfy π it must not produce a trace that is not part of π. Thus, p satisfies π iff $Behav(p) \subseteq \pi$ and we write $p \vDash \pi$.*

Definition 11 (Hyperproperties)**.** *A hyperproperty H is a set of admissible sets of traces. Thus, if p satisfies H (also written $p \vDash H$), then $Behav(p) \in H$.*

Lemma 1 (Lifting Properties)**.** *Given a property π, there exists a unique hyperproperty $\lceil \pi \rceil$ that satisfies the exact same policy.*

Proof. We want $\forall p \in \mathcal{P}, p \vDash \pi \equiv p \vDash \lceil \pi \rceil$. Henceforth, given a $p \in \mathcal{P}$, we have $\text{Behav}(p) \subseteq \pi$ iff $\text{Behav}(p) \in \lceil \pi \rceil$. Note that if $\text{Behav}(p) \subseteq \pi$, we have $\text{Behav}(p) \in \{\Pi \mid \Pi = \text{Behav}(p) \subseteq \pi\}$. Thus, $\lceil \pi \rceil$ is the set of all possible program behaviors that are a subset of π. This is exactly the powerset of π and we conclude $\lceil \pi \rceil = \mathscr{P}(\pi)$. □

The lifting of properties to a singleton set does not suffice, since the empty behavior trivially satisfies any property $\emptyset \subseteq \pi = \{\overline{a}\}$, but if we would define $\lceil \overline{a} \rceil = \{\{\overline{a}\}\}$, then $\emptyset \notin \lceil \overline{a} \rceil$.

Lemma 2 (Property Satisfaction Refinement)**.** *For a property π that refines π', i.e. $\pi \subseteq \pi'$, if any $p \in \mathcal{P}$ satisfies $p \vDash \pi'$, then $p \vDash \pi$.*

Proof. Pick any property π' and $p \in \mathcal{P}$ such that $p \vDash \pi$ and assume $\pi \subseteq \pi'$. Simple unfolding reveals $\text{Behav}(p) \subseteq \pi \implies \text{Behav}(p) \subseteq \pi'$. □

For lifted properties, this refinement property also holds on the hyperproperty level. However, it does not work for any hyperproperty [2].

Definition 12 (Robust Property Satisfaction)**.** *A program p robustly satisfies a property π, written $p \vDash_R \pi$, iff $\forall C \in \mathcal{P}, C \bowtie p \vDash \pi$. The same notation is used for robust hyperproperty satisfaction.*

Lemma 3 (Weakening Robust Satisfaction)**.** *Given classes $\mathbb{C}_1, \mathbb{C}_2$ and any program p such that*

(a) $\mathbb{C}_1 \subseteq \mathbb{C}_2$

(b) $p \vDash_R \mathbb{C}_2$

We show

(i) $p \vDash_R \mathbb{C}_1$

Proof. Unfolding Item (i) (Weakening Robust Satisfaction), let $\Pi \in \mathbb{C}_2$ and p be a program, we want to show that $p \vDash_R \Pi$. By Item (a) (Weakening Robust Satisfaction), we also know that $\Pi \in \mathbb{C}_1$. Thus, we can use Item (b) (Weakening Robust Satisfaction) to conclude. ☐

Definition 13 (Classes)**.** *A class of hyperproperties* $\mathbb{C}$ *is a set of hyperproperties. Likewise, a class of properties* $\mathbb{C}$ *is a set of hyperproperties, where every property is lifted to the hyperproperty level. From now on, we use* Π *for elements of any class* $\mathbb{C}$ *in case it does not matter whether it is a lifted property or any hyperproperty.*

Definition 14 (Compilers)**.** *A compiler between languages* S *and* $\mathbf{T}$ *is a partial function* $[\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}}$ *from* $\mathcal{P}$ *to* $\mathcal{P}$*.*

2 Compositionality of Secure Compilers

Definition 15 (Universal Image)**.**

$$\sigma_\sim(\pi) = \{\bar{\mathsf{a}} | \forall \bar{\mathbf{a}}, \bar{\mathsf{a}} \sim \bar{\mathbf{a}} \implies \bar{\mathbf{a}} \in \pi\}$$

Definition 16 (Existential Image)**.**

$$\tau_\sim(\pi) = \{\bar{\mathbf{a}} | \exists \bar{\mathsf{a}}, \bar{\mathsf{a}} \sim \bar{\mathbf{a}} \wedge \bar{\mathsf{a}} \in \pi\}$$

Definition 17 (Robust Trace-Hyperproperty Preservation with Universal Image)**.** *For a given class* $\mathbb{C}$*, a compiler from languages* S *to* $\mathbf{T}$ *robustly preserves* $\mathbb{C}$ *iff*

$$\forall \mathbf{\Pi} \in \mathbb{C}, \forall \mathsf{p} \in \mathcal{P}, \mathsf{p} \vDash_R \sigma_\sim(\mathbf{\Pi}) \implies [\![\mathsf{p}]\!]^{\mathsf{S}\to\mathbf{T}} \vDash_R \mathbf{\Pi}$$

We write $\vdash_{\sigma_\sim} [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}$*.*

Definition 18 (Robust Trace-Hyperproperty Preservation with Existential Image)**.** *For a given class* $\mathbb{C}$*, a compiler from languages* S *to* $\mathbf{T}$ *robustly preserves* $\mathbb{C}$ *iff*

$$\forall \mathsf{\Pi} \in \mathbb{C}, \forall \mathsf{p} \in \mathcal{P}, \mathsf{p} \vDash_R \mathsf{\Pi} \implies [\![\mathsf{p}]\!]^{\mathsf{S}\to\mathbf{T}} \vDash_R \tau_\sim(\mathsf{\Pi})$$

We write $\vdash_{\tau_\sim} [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}$*.*

Definition 19 (Robust Trace-Hyperproperty Preservation)**.** *For a given class* $\mathbb{C}$*, a compiler from languages* S *to* $\mathbf{T}$ *robustly preserves* $\mathbb{C}$ *iff*

$$\forall \Pi \in \mathbb{C}, \forall \mathsf{p} \in \mathcal{P}, \mathsf{p} \vDash_R \Pi \implies [\![\mathsf{p}]\!]^{\mathsf{S}\to\mathbf{T}} \vDash_R \Pi$$

We write $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}$*.*

Definition 20 (Sequential Composition of Compilers)**.** *Given two compilers* $[\![\bullet]\!]^{\mathsf{S}\to I}$ *and* $[\![\bullet]\!]^{I\to\mathbf{T}}$*, their sequential composition is* $[\![\bullet]\!]^{\mathsf{S}\to I\to\mathbf{T}} = [\![[\![\bullet]\!]^{\mathsf{S}\to I}]\!]^{I\to\mathbf{T}}$*.*

Lemma 4 (Weakening RTP)**.** *Given classes* $\mathbb{C}_1, \mathbb{C}_2$ *such that*

(a) $\mathbb{C}_1 \subseteq \mathbb{C}_2$

(b) $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_2$

We show

(i) $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_1$

Proof. Using Definition 19 on the goal, let $\Pi \in \mathbb{C}_1$ and $\mathsf{p} \in \mathcal{P}$ such that $\mathsf{p} \vDash_R \Pi$, so what's left to prove is $[\![\mathsf{p}]\!]^{\mathsf{S}\to\mathbf{T}} \vDash_R \Pi$. Since $\mathbb{C}_1 \subseteq \mathbb{C}_2$ and $\Pi \in \mathbb{C}_1$, we know that $\Pi \in \mathbb{C}_2$. Thus, we can apply the assumption $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_2$ to our goal, leaving us with $\mathsf{p} \vDash_R \Pi$ to show, which was an assumption we made. $\square$

Definition 21 (Well-formedness of $\sim$ for a Class $\mathbb{C}$)**.**

$$\vdash_{wf} \sim : \mathbb{C} := \forall \pi \in \mathbb{C}, \tau_{\sim}(\pi) \in \tau_{\sim}(\mathbb{C})$$

Definition 22 (Well-formedness of $\sim$ for a Class $\mathbb{C}$)**.**

$$\vdash_{wf} \sim : \mathbb{C} := \forall \pi \in \mathbb{C}, \sigma_{\sim}(\pi) \in \sigma_{\sim}(\mathbb{C})$$

Definition 23 (Safety Properties)**.** *The class of safety properties contains the lifting of all properties that can be refuted with a finite trace prefix:*

$$Safety = \{\lceil\pi\rceil \mid \forall \overline{a} \in Traces, t \notin \lceil\pi\rceil \ \textit{ iff } \exists m \geq \overline{a}, \forall \overline{a}' \in Traces, m \leq \overline{a}' \implies \overline{a}' \notin \lceil\pi\rceil\}$$

Definition 24 (Hypersafety Properties)**.** *The class of hypersafety properties contains all hyperpropert that can be refuted with an observation:*

$$HyperSafety = \left\{\Pi \mid \forall b \in 2^{Traces}, b \notin \Pi \textit{ iff } \exists o \geq b, \forall b' \in 2^{Traces}, o \leq b' \implies b' \notin \Pi\right\}$$

Definition 25 (Subset Closed Hyperproperties)**.** *The class of hyperproperties that are closed with respect to the subset relation is*

$$SSC = \{H \mid \forall X \in H, \forall Y \subseteq X, Y \in H\}$$

Lemma 5 (Hypersafety is entailed in SSC)**.** *$HyperSafety \subseteq SSC$.*

Proof. [2] $\square$

Definition 26 (K-Hypersafety)**.** *Exactly the same as Definition 24, but the observations o are restricted to cardinality k. 2-Hypersafety is simply $k = 2$. Definition 27 gives an example instance of a classic 2-hypersafe property.*

Definition 27 (Non-Interference (NI))**.** *We define the class containing the non-interference hyperproperty as:*

$$NI = \{H | \forall \overline{a}_1, \overline{a}_2 \in H. \overline{a}_1 =_L \overline{a}_2 \implies \overline{a}_1 = \overline{a}_2\}$$

Note that $=$ may not be strict equality, but some suitable trace equivalence that checks both public and private actions, instead of just public.

3 Composition of Previous Results

Lemma 6 (Sequential Composition with RTPτ)**.** *Given* $\vdash_{wf} \sim_1 : \mathbb{C}_2$*,* $\vdash_{\tau_{\sim_1}} [\![\bullet]\!]^{\mathsf{S}\to I} : \mathbb{C}_1$*, and* $\vdash_{\tau_{\sim_2}} [\![\bullet]\!]^{I\to\mathbf{T}} : \tau_{\sim_1}(\mathbb{C}_2)$*, then* $\vdash_{\tau_{\sim_1\circ\sim_2}} [\![\bullet]\!]^{\mathsf{S}\to I\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. We need to show $\vdash_{\tau_{\sim_1\circ\sim_2}} [\![\bullet]\!]^{\mathsf{S}\to I\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$. By definition, assume $\pi \in \mathbb{C}_1 \cap \mathbb{C}_2$ and $\mathsf{p} \in \mathcal{P}$ such that $\mathsf{p} \vDash_R \pi$. What is left to show is $[\![\mathsf{p}]\!]^{\mathsf{S}\to I\to\mathbf{T}} \vDash_R \tau_{\sim_1\circ\sim_2}(\pi)$. Note that $\pi \in \mathbb{C}_1$ as well as $\pi \in \mathbb{C}_2$. Also, $[\![\mathsf{p}]\!]^{\mathsf{S}\to I} \in \mathcal{P}$ and $\tau_{\sim_1\circ\sim_2}(\pi) = \tau_{\sim_2}(\tau_{\sim_1}(\pi))$. Since $\sim_1$ is well-formed with respect to $\mathbb{C}_2$, we can apply $\vdash_{\tau_{\sim_2}} [\![\bullet]\!]^{I\to\mathbf{T}} : \tau_{\sim_1}(\mathbb{C}_2)$, changing our goal to $[\![\mathsf{p}]\!]^{\mathsf{S}\to I} \vDash_R \tau_{\sim_1}(\pi)$. Since $\pi \in \mathbb{C}_1$ also holds, we can this time apply $\vdash_{\tau_{\sim_1}} [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_1$. What is left to show is $\mathsf{p} \vDash_R \pi$, which is an assumption of ours. ☐

Lemma 7 (Sequential Composition with RTPσ)**.** *Given* $\vdash_{wf} \sim_2 : \mathbb{C}_1$*,* $\vdash_{\sigma_{\sim_1}} [\![\bullet]\!]^{\mathsf{S}\to I} : \sigma_{\sim_2}(\mathbb{C}_1)$*, and* $\vdash_{\sigma_{\sim_2}} [\![\bullet]\!]^{I\to\mathbf{T}} : \mathbb{C}_2$*, then* $\vdash_{\sigma_{\sim_1\circ\sim_2}} [\![\bullet]\!]^{\mathsf{S}\to I\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. Dual to Lemma 6. ☐

Lemma 8 (Sequential Composition with RTP)**.** *Given* $\vdash [\![\bullet]\!]^{\mathsf{S}\to I} : \mathbb{C}_1$ *and* $\vdash [\![\bullet]\!]^{I\to\mathbf{T}} : \mathbb{C}_2$*, then* $\vdash [\![\bullet]\!]^{\mathsf{S}\to I\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. Simple consequence from either Lemma 6 or Lemma 7 by setting the cross-language trace relations to $=$. ☐

Definition 28 (Upper Composition)**.** *Given two compilers* $[\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}}$ *and* $[\![\bullet]\!]^{I\to\mathbf{T}}$*, their upper composition is*

$$[\![\bullet]\!]^{\mathsf{S}+I\to\mathbf{T}} = \lambda p. \begin{cases} [\![\mathsf{p}]\!]^{\mathsf{S}\to\mathbf{T}} & \textit{if } p \in \mathcal{P} \\ [\![p]\!]^{I\to\mathbf{T}} & \textit{if } p \in \mathcal{P} \end{cases}$$

.

Lemma 9 (Upper Composition with RTP)**.** *Given* $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_1$ *and* $\vdash [\![\bullet]\!]^{I\to\mathbf{T}} : \mathbb{C}_2$*, then* $\vdash [\![\bullet]\!]^{\mathsf{S}+I\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. Analogous argument as in Lemma 8 (Sequential Composition with RTP), but with a case distinction on whether the source program is element of S or I. ☐

Definition 29 (Lower Composition)**.** *Given two compilers* $[\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}}$ *and* $[\![\bullet]\!]^{\mathsf{S}\to I}$*, their lower composition is* $[\![\bullet]\!]^{\mathsf{S}\to I+\mathbf{T}}$*.*

Lemma 10 (Lower Composition with RTP)**.** *Given* $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_1$ *and* $\vdash [\![\bullet]\!]^{\mathsf{S}\to I} : \mathbb{C}_2$*, then* $\vdash [\![\bullet]\!]^{\mathsf{S}\to I+\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. Analogous argument as in Lemma 8 (Sequential Composition with RTP), but with a case distinction on whether the compiled source program is element of I or $\mathbf{T}$. ☐

Lemma 11 (Diamond)**.** *Given* $\vdash [\![\bullet]\!]^{\mathsf{S}\to I+\mathtt{0}} : \mathbb{C}_1$ *and* $\vdash [\![\bullet]\!]^{I+\mathtt{0}\to\mathbf{T}} : \mathbb{C}_2$ *with* $[\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} = \lambda\mathsf{p}.[\![[\![\mathsf{p}]\!]^{\mathsf{S}\to I+\mathtt{0}}]\!]^{I+\mathtt{0}\to\mathbf{T}}$*, then* $\vdash [\![\bullet]\!]^{\mathsf{S}\to\mathbf{T}} : \mathbb{C}_1 \cap \mathbb{C}_2$*.*

Proof. Straightforward using Lemma 8 (Sequential Composition with RTP). ☐

Lemma 12 (Swappable). *Given* $\vdash [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_1$ *and* $\vdash [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_2$*, then* $\vdash [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_2 \cap \mathbb{C}_1$ *and* $\vdash [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_1 \cap \mathbb{C}_2$.

Proof. Both follow from Lemma 8 (Sequential Composition with RTP). □

Lemma 13 (Swappableσ). *Given* $\vdash_{wf} \sim_2 : \mathbb{C}_1$, $\vdash_{wf} \sim_1 : \mathbb{C}_2$, $\vdash_{\sigma_{\sim_1}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_1$, $\vdash_{\sigma_{\sim_1}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \sigma_{\sim_2}(\mathbb{C}_1)$, $\vdash_{\sigma_{\sim_2}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \sigma_{\sim_1}(\mathbb{C}_2)$, *and* $\vdash_{\sigma_{\sim_2}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_2$, *then* $\vdash_{\sigma_{\sim_1 \circ \sim_2}} [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_2 \cap \mathbb{C}_1$ *and* $\vdash_{\sigma_{\sim_2 \circ \sim_1}} [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_1 \cap \mathbb{C}_2$.

Proof. Easy from Lemma 7 (Sequential Composition with RTPσ). □

Lemma 14 (Swappableτ). *Given* $\vdash_{wf} \sim_2 : \mathbb{C}_1$, $\vdash_{wf} \sim_1 : \mathbb{C}_2$, $\vdash_{\tau_{\sim_1}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_1$, $\vdash_{\tau_{\sim_1}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \tau_{\sim_2}(\mathbb{C}_1)$, $\vdash_{\tau_{\sim_2}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \tau_{\sim_1}(\mathbb{C}_2)$, *and* $\vdash_{\tau_{\sim_2}} [\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_2$, *then* $\vdash_{\tau_{\sim_1 \circ \sim_2}} [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)} : \mathbb{C}_2 \cap \mathbb{C}_1$ *and* $\vdash_{\tau_{\sim_2 \circ \sim_1}} [\![[\![\bullet]\!]^{\mathbf{T}\to\mathbf{T}}_{(1)}]\!]^{\mathbf{T}\to\mathbf{T}}_{(2)} : \mathbb{C}_1 \cap \mathbb{C}_2$.

Proof. Easy from Lemma 6 (Sequential Composition with RTPτ). □

4 Case Study

4.1 Specification Language

We introduce a set of actions that allows us to describe properties abstractly instead of a language specific manner.

$$\begin{array}{l}
\textit{Control Tag } t ::= \textit{ctx} \mid \textit{comp} \qquad \textit{Security Tag } \underline{\sigma} ::= 🔒 \mid 🔓 \\
\textit{Pre–Events } a_b^{\text{ms}} ::= \underline{\textit{Alloc } \ell\ n} \mid \underline{\textit{Dealloc } \ell} \mid \underline{\textit{Use } \ell\ n} \mid \underline{\textit{Branch } n} \mid \underline{\textit{Binop } n} \\
\textit{Events } a^{\text{ms}} ::= \underline{\varepsilon} \mid \underline{\lightning} \mid \underline{a_b^{\text{ms}}}; t; \underline{\sigma}
\end{array}$$

Figure 1: Specification Events.

Definition 30 (TMS).

$$\textit{tms} := \left\{ \overline{a_{\text{ms}}} \;\middle|\; \begin{array}{rcl} \underline{\textit{Alloc } \ell\ n} & \leq_{\overline{a_{\text{ms}}}} & \underline{\textit{Dealloc } \ell} \\ \underline{\textit{Use } \ell\ n} & \leq_{\overline{a_{\text{ms}}}} & \underline{\textit{Dealloc } \ell} \\ \underline{\textit{Dealloc } \ell} & \in_! & \overline{a_{\text{ms}}} \\ \underline{\textit{Alloc } \ell\ n} & \in_! & \overline{a_{\text{ms}}} \end{array} \right\}$$

4.1.1 TMS Monitor

In order to just talk about temporal memory safety, we introduce a monitor that works on more abstract monitor-actions, without any other events besides those relevant to temporal memory safety.

$$Abstract\ Store\ T_{\text{TMS}} = \{A : \underline{L} \times t, F : \underline{L} \times t\}$$

$$Abstract\ Events\ \boldsymbol{a} ::= \boldsymbol{\varepsilon} \mid \textbf{Alloc}\ \underline{\ell}\ t \mid \textbf{Dealloc}\ \underline{\ell}\ t \mid \textbf{Use}\ \underline{\ell}\ t \mid \lightning$$

$$\begin{aligned}
T_{\text{TMS}} \subseteq_F T_{\text{TMS}}' &\text{ iff } T_{\text{TMS}}.F \subseteq T_{\text{TMS}}'.F \\
\boldsymbol{a} \in T_{\text{TMS}} &\text{ iff } \boldsymbol{a} \in T_{\text{TMS}}.A \wedge \boldsymbol{a} \notin T_{\text{TMS}}.F \\
\boldsymbol{a} \notin T_{\text{TMS}} &\text{ iff } \boldsymbol{a} \notin T_{\text{TMS}}.A \wedge \boldsymbol{a} \notin T_{\text{TMS}}.F \\
\{(\underline{\ell}; t)\} \cup T_{\text{TMS}} &= \{A : \{(\underline{\ell}; t)\} \cup T_{\text{TMS}}.A, F : T_{\text{TMS}}.F\} \\
T_{\text{TMS}} \setminus \{(\underline{\ell}; t)\} &= \{A : T_{\text{TMS}}.A \setminus \{(\underline{\ell}; t)\}, F : T_{\text{TMS}}.F \cup \{(\underline{\ell}; t)\}\} \\
T_{\text{TMS}} \cup T_{\text{TMS}}' &= \{A : T_{\text{TMS}}.A \cup T_{\text{TMS}}'.A, F : T_{\text{TMS}}.F \cup T_{\text{TMS}}'.F\}
\end{aligned}$$

Figure 2: TMS Monitor.

As before, when doing structural induction over primitive steps (Figure 27) we may encounter the ε, for which $\theta(\varepsilon) = \underline{\varepsilon}$, which needs a „partner" in the abstract events as defined in Figure 2: $\boldsymbol{\varepsilon}$.

$\boxed{\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'}$ „Given abstract state T_{TMS}, step to abstract state T_{TMS}' emitting an abstract event $\boldsymbol{a}$."

$\boxed{\vdash T_{\text{TMS}} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{TMS}}'}$ „Reflexive-Transitive closure of above judgement. $\overline{\boldsymbol{a}}$ is a list of abstract events."

$$\frac{}{\vdash T_{\text{TMS}} \overset{\boldsymbol{\varepsilon}}{\rightsquigarrow} T_{\text{TMS}}}\ \text{(TMS-Uninteresting)} \qquad \frac{}{\vdash T_{\text{TMS}} \overset{\lightning}{\rightsquigarrow} T_{\text{TMS}}}\ \text{(TMS-Abort)} \qquad \frac{(\underline{\ell}; t) \in T_{\text{TMS}}}{\vdash T_{\text{TMS}} \overset{\textbf{Use}\ \underline{\ell}\ t}{\rightsquigarrow} T_{\text{TMS}}}\ \text{(TMS-Use)}$$

$$\frac{T_{\text{TMS}}' = \{(\underline{\ell}; t)\} \cup T_{\text{TMS}} \qquad (\underline{\ell}; t) \notin T_{\text{TMS}}}{\vdash T_{\text{TMS}} \overset{\textbf{Alloc}\ \underline{\ell}\ t}{\rightsquigarrow} T_{\text{TMS}}'}\ \text{(TMS-Alloc)}$$

$$\frac{(\underline{\ell}; t) \in T_{\text{TMS}} \qquad T_{\text{TMS}}' = T_{\text{TMS}} \setminus \{(\underline{\ell}; t)\}}{\vdash T_{\text{TMS}} \overset{\textbf{Dealloc}\ \underline{\ell}\ t}{\rightsquigarrow} T_{\text{TMS}}'}\ \text{(TMS-Dealloc)} \qquad \frac{T_{\text{TMS}}.A = \emptyset}{\vdash T_{\text{TMS}} \overset{[\cdot]}{\rightsquigarrow}{}^{*} T_{\text{TMS}}}\ \text{(TMS-Refl)}$$

$$\frac{\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}' \qquad \vdash T_{\text{TMS}}' \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{TMS}}''}{\vdash T_{\text{TMS}} \overset{\boldsymbol{a} \cdot \overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{TMS}}''}\ \text{(TMS-Trans)}$$

Figure 3: Steps of TMS Monitor.

The monitor-state contains two sets of locations that represent the ones that are active and the ones that have been deallocated, respectively. As seen in Rule TMS-Dealloc, the monitor only steps if the intuitive condition is true: a location can only be deallocated if it is part of the set of allocated locations. Rule TMS-Use ensures that only allocated locations occur in events representing usage, the monitor cannot step if the location has been deallocated before.

As before, we need a way to translate one set of actions to another:

$\boxed{a^{\text{ms}} \cong \boldsymbol{a}}$ „Memory-Safety specific event a^{ms} describes same action as $\boldsymbol{a}$."

$$\frac{}{\underline{Alloc\ \ell\ n}; t; \underline{\sigma} \cong \textbf{Alloc}\ \underline{\ell}\ t}\ \text{(TMS-Alloc-Authentic)} \qquad \frac{}{\underline{Dealloc\ \ell}; t; \underline{\sigma} \cong \textbf{Dealloc}\ \underline{\ell}\ t}\ \text{(TMS-Dealloc-Authentic)}$$

$$\frac{}{\underline{Use\ \ell\ n}; t; \underline{\sigma} \cong \textbf{Use}\ \underline{\ell}\ t}\ \text{(TMS-Use-Authentic)} \qquad \frac{}{\underline{\lightning} \cong \lightning}\ \text{(TMS-Abort-Authentic)} \qquad \frac{}{\underline{\varepsilon} \cong \boldsymbol{\varepsilon}}\ \text{(TMS-None-Authentic)}$$

$$\frac{}{\underline{Branch\ n}; t; \underline{\sigma} \cong \boldsymbol{\varepsilon}}\ \text{(TMS-Branch-Authentic)} \qquad \frac{}{\underline{Binop\ n}; t; \underline{\sigma} \cong \boldsymbol{\varepsilon}}\ \text{(TMS-Binop-Authentic)}$$

$\boxed{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}$ „Memory-Safety specific trace $\overline{a^{\text{ms}}}$ describes same trace as $\overline{\boldsymbol{a}}$."

$$\frac{}{[\cdot] \cong^* [\cdot]}\ \text{(TMS-Refl-Authentic)} \qquad \frac{a^{\text{ms}} \cong \boldsymbol{a} \quad \overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{a^{\text{ms}} \cdot \overline{a^{\text{ms}}} \cong^* \boldsymbol{a} \cdot \overline{\boldsymbol{a}}}\ \text{(TMS-Trans-Authentic)} \qquad \frac{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{\underline{\varepsilon} \cdot \overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}\ \text{(TMS-Trans-Authentic-Lign)}$$

$$\frac{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{\overline{a^{\text{ms}}} \cong^* \boldsymbol{\varepsilon} \cdot \overline{\boldsymbol{a}}}\ \text{(TMS-Trans-Authentic-Rign)}$$

Figure 4: Trace Agreement.

Definition 31 (Trace is temporal memory safe via monitor.)**.** *We say* $\texttt{TMS}(\overline{a^{ms}})$ *iff* $\exists \delta_{tms}\ \overline{\boldsymbol{a}}\ T_{TMS}$ *such that:*

- $\overline{a^{ms}} \cong \overline{\boldsymbol{a}}$
- *and* $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{TMS}$

Lemma 15 ($\texttt{TMS}(\overline{a^{\text{ms}}})$ implies $\overline{a^{\text{ms}}} \in$ tmsafe)**.** *If*

(a) $\texttt{TMS}(\overline{a^{ms}})$

Then

(i) $\overline{a^{ms}} \in \text{tmsafe}$

Proof. Invert Assumption (a):

(H_1) $\overline{a^{\text{ms}}} \cong \overline{\boldsymbol{a}}$

(H_2) $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{\text{TMS}}$

The proof follows with an induction on Assumption (H_2). □

4.1.2 SMS Monitor

$$\textit{Abstract Store}\ T_{\text{SMS}} = \underline{L} \times t \times \mathbb{N}$$
$$\textit{Abstract Events}\ \boldsymbol{a} ::= \boldsymbol{\varepsilon} \mid \textbf{Alloc}\ \underline{\ell}\ t\ \boldsymbol{n} \mid \textbf{Use}\ \underline{\ell}\ t\ \boldsymbol{n} \mid \lightning$$

Figure 5: SMS Monitor.

Definition 32 (SMS)**.**

$$\text{sms} := \left\{ \overline{a_{\text{ms}}} \,\middle|\, \textit{If}\ \underline{Alloc\ \ell\ n} \leq_{\overline{a}_{\text{ms}}} \underline{Use\ \ell\ m},\ \textit{then}\ m < n \right\}$$

$$\boxed{\vdash T_{\text{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{SMS}}'}$$ „Given abstract state T_{SMS}, step to abstract state T_{SMS}' emitting an abstract event $\boldsymbol{a}$."

$$\boxed{\vdash T_{\text{SMS}} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{SMS}}'}$$ „Reflexive-Transitive closure of above judgement. $\overline{\boldsymbol{a}}$ is a list of abstract events."

$$\frac{}{\vdash T_{\text{SMS}} \overset{\boldsymbol{\varepsilon}}{\rightsquigarrow} T_{\text{SMS}}}\;\text{(sms-None)} \qquad \frac{}{\vdash T_{\text{SMS}} \overset{\lightning}{\rightsquigarrow} T_{\text{SMS}}}\;\text{(sms-Abort)}$$

$$\frac{(\underline{\ell}; t; m) \in T_{\text{SMS}} \qquad n < m}{\vdash T_{\text{SMS}} \xrightarrow{\textbf{Use}\ \underline{\ell}\ t\ \boldsymbol{n}} T_{\text{SMS}}}\;\text{(sms-Use)} \qquad \frac{(\underline{\ell}; t; m) \notin \operatorname{dom} T_{\text{SMS}}}{\vdash T_{\text{SMS}} \xrightarrow{\textbf{Alloc}\ \underline{\ell}\ t\ \boldsymbol{n}} T_{\text{SMS}} \cup \{(\underline{\ell}; t; n)\}}\;\text{(sms-Alloc)}$$

$$\frac{}{\vdash T_{\text{SMS}} \overset{[\cdot]}{\rightsquigarrow}{}^{*} T_{\text{SMS}}}\;\text{(sms-Refl)} \qquad \frac{\vdash T_{\text{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{SMS}}' \qquad \vdash T_{\text{SMS}}' \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{SMS}}''}{\vdash T_{\text{SMS}} \overset{\boldsymbol{a}\cdot\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{SMS}}''}\;\text{(sms-Trans)}$$

Figure 6: Steps of SMS Monitor.

$$\boxed{a^{\text{ms}} \cong \boldsymbol{a}}$$ „Memory-Safety specific event a^{ms} describes same action as $\boldsymbol{a}$."

$$\frac{}{\underline{\mathit{Alloc}\ \ell\ n}; t; \underline{\sigma} \cong \textbf{Alloc}\ \underline{\ell}\ t\ \boldsymbol{n}}\;\text{(SMS-Alloc-Authentic)} \qquad \frac{}{\underline{\mathit{Dealloc}\ \ell}; t; \underline{\sigma} \cong \boldsymbol{\varepsilon}}\;\text{(SMS-Dealloc-Authentic)}$$

$$\frac{}{\underline{\mathit{Use}\ \ell\ n}; t; \underline{\sigma} \cong \textbf{Use}\ \underline{\ell}\ t\ \boldsymbol{n}}\;\text{(SMS-Use-Authentic)} \qquad \frac{}{\underline{\varepsilon} \cong \boldsymbol{\varepsilon}}\;\text{(SMS-None-Authentic)}$$

$$\frac{}{\underline{\mathit{Branch}\ n}; t; \underline{\sigma} \cong \boldsymbol{\varepsilon}}\;\text{(SMS-Branch-Authentic)} \qquad \frac{}{\underline{\mathit{Binop}\ n}; t; \underline{\sigma} \cong \boldsymbol{\varepsilon}}\;\text{(SMS-Binop-Authentic)} \qquad \frac{}{\underline{\lightning} \cong \lightning}\;\text{(SMS-Abort-Authentic)}$$

$$\boxed{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}$$ „Memory-Safety specific trace $\overline{a^{\text{ms}}}$ describes same trace as $\overline{\boldsymbol{a}}$."

$$\frac{}{[\cdot] \cong [\cdot]}\;\text{(SMS-Refl-Authentic)} \qquad \frac{a^{\text{ms}} \cong \boldsymbol{a} \qquad \overline{a^{\text{ms}}} \cong \overline{\boldsymbol{a}}}{a^{\text{ms}} \cdot \overline{a^{\text{ms}}} \cong \boldsymbol{a} \cdot \overline{\boldsymbol{a}}}\;\text{(SMS-Trans-Authentic)} \qquad \frac{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\underline{\varepsilon} \cdot \overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}\;\text{(SMS-Trans-Authentic-Lign)}$$

$$\frac{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\overline{a^{\text{ms}}} \cong^{*} \boldsymbol{\varepsilon} \cdot \overline{\boldsymbol{a}}}\;\text{(SMS-Trans-Authentic-Rign)}$$

Figure 7: Trace Agreement.

Definition 33 (Trace is spatial memory safe via monitor.)**.** *We say* $\texttt{SMS}(\overline{a^{ms}})$ *iff* $\exists \overline{\boldsymbol{a}}\ T_{SMS}$ *such that:*

- $\overline{a^{ms}} \cong \overline{\boldsymbol{a}}$
- *and* $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{SMS}$

Lemma 16 ($\texttt{SMS}(\overline{a^{\text{ms}}})$ implies $\overline{a^{\text{ms}}} \in \text{smsafe}$)**.** *If*

(a) $\texttt{SMS}(\overline{a^{ms}})$

Then

(i) $\overline{a^{ms}} \in \text{smsafe}$

Proof. Invert Assumption (a):

(H_1) $\overline{a^{\mathrm{ms}}} \cong \overline{\boldsymbol{a}}$

(H_2) $\vdash \emptyset \xrightsquigarrow{\overline{\boldsymbol{a}}}{}^{*} T_{\mathrm{SMS}}$

The proof follows with an induction on Assumption (H_2). $\square$

4.1.3 MS Monitor

$$\begin{array}{c} \textit{Abstract Store } T_{\mathrm{MS}} = T_{\mathrm{TMS}} \times T_{\mathrm{SMS}} \\ \textit{Abstract Events } \boldsymbol{a} ::= \boldsymbol{a}_{\mathrm{tms}} \times \boldsymbol{a}_{\mathrm{sms}} \end{array}$$

Figure 8: MS Monitor.

Definition 34 (MS)**.** ms := tms ∩ sms

$$\text{(tms-sms-E)}\quad \frac{\vdash T_{\mathrm{SMS}} \xrightsquigarrow{\boldsymbol{a}_{\mathrm{tms}}} T_{\mathrm{SMS}} \qquad \vdash T_{\mathrm{TMS}} \xrightsquigarrow{\boldsymbol{a}_{\mathrm{sms}}} T_{\mathrm{TMS}}}{\vdash (T_{\mathrm{TMS}}; T_{\mathrm{SMS}}) \xrightsquigarrow{\boldsymbol{a}_{tms}; \boldsymbol{a}_{sms}} (T_{\mathrm{TMS}}'; T_{\mathrm{SMS}}')}$$

$$\text{(tms-sms-Refl)}\quad \frac{\vdash T_{\mathrm{TMS}} \xrightsquigarrow{[\cdot]}{}^{*} T_{\mathrm{TMS}} \qquad \vdash T_{\mathrm{SMS}} \xrightsquigarrow{[\cdot]}{}^{*} T_{\mathrm{SMS}}}{\vdash (T_{\mathrm{TMS}}; T_{\mathrm{SMS}}) \xrightsquigarrow{[\cdot]}{}^{*} (T_{\mathrm{TMS}}; T_{\mathrm{SMS}})}$$

$$\text{(tms-sms-Trans)}\quad \frac{\vdash T \xrightsquigarrow{\boldsymbol{a}} T' \qquad \vdash T' \xrightsquigarrow{\overline{\boldsymbol{a}}}{}^{*} T''}{\vdash T \xrightsquigarrow{\boldsymbol{a} \cdot \overline{\boldsymbol{a}}}{}^{*} T''}$$

Figure 9: Steps of combined TMS + SMS Monitor.

$\boxed{\overline{a^{\mathrm{ms}}} \cong^{*} \overline{\boldsymbol{a}}}$ „Memory-Safety specific trace $\overline{a^{\mathrm{ms}}}$ describes same trace as $\overline{\boldsymbol{a}}$."

$$\text{(MS-Refl-Authentic)}\quad \frac{}{[\cdot] \cong [\cdot]} \qquad \text{(MS-Trans-Authentic)}\quad \frac{a^{\mathrm{ms}} \cong \boldsymbol{a}_{\mathrm{tms}} \qquad a^{\mathrm{ms}} \cong \boldsymbol{a}_{\mathrm{sms}} \qquad \overline{a^{\mathrm{ms}}} \cong \overline{\boldsymbol{a}}}{a^{\mathrm{ms}} \cdot \overline{a^{\mathrm{ms}}} \cong (\boldsymbol{a}_{\mathrm{tms}}; \boldsymbol{a}_{\mathrm{sms}}) \cdot \overline{\boldsymbol{a}}}$$

$$\text{(MS-Trans-Authentic-Lign)}\quad \frac{\overline{a^{\mathrm{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\underline{\varepsilon} \cdot \overline{a^{\mathrm{ms}}} \cong^{*} \overline{\boldsymbol{a}}} \qquad \text{(MS-Trans-Authentic-Rign)}\quad \frac{\overline{a^{\mathrm{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\overline{a^{\mathrm{ms}}} \cong^{*} \boldsymbol{\varepsilon} \cdot \overline{\boldsymbol{a}}}$$

Figure 10: Trace Agreement.

Definition 35 (Trace is memory safe via monitor.)**.** *We say* $\mathtt{MS}(\overline{a^{ms}})$ *iff* $\exists \overline{\boldsymbol{a}}\ T_{MS}$ *such that:*

- $\overline{a^{ms}} \cong \overline{\boldsymbol{a}}$
- *and* $\vdash \emptyset \xrightsquigarrow{\overline{\boldsymbol{a}}}{}^{*} T_{MS}$

Lemma 17 ($\mathtt{MS}(\overline{a^{\mathrm{ms}}})$ implies $\overline{a^{\mathrm{ms}}} \in$ msafe)**.** *If* *(a)* $\mathtt{MS}(\overline{a^{ms}})$

Then

(i) $\overline{a^{ms}} \in \text{msafe}$

Proof. Invert Assumption (a):

(H_1) $\overline{a^{\text{ms}}} \cong \overline{\boldsymbol{a}}$

(H_2) $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{MS}}$

The proof follows with Lemmas 15 and 16. ☐

4.1.4 sCCT Monitor

`sCCT`-Problems:

- branching or predicated execution
- comparisons are an issue: can only get the content of special flag register by branching
- timing of loads/stores changes
- mult/div is problematic. mult is safe on modern procs, but not on micro-procs
- bitshifts may translate into loops (unless bitshift by constant)

To remedy most of these real-world problems, which are dependent on the hardware, we introduce another language which has `sCCT`-hardened instructions. That is, we assume there exists an $\hat{\oplus}$ which behaves similarily to $\oplus$, but without leaking timing information. Moreover, we also assume that there are `sCCT`-versions of reading and writing. If the instruction set architecture does not support these, we argue that there is some encoding that can be done in the style of FaCT that the compiler would have to do. For example, to circumvent loading/storing and ensure `sCCT`-versions, the register allocator could ensure that this variable stays in a register. Whenever this is not possible, one could spill and, prior to re-loading the variable, invalidate the caches. Another option is to simply load the whole array sequentially and select the relevant entry using a bitmask operation.

$$\begin{aligned}
\textit{Abstract Store } T_{\text{sCCT}} &= \emptyset\\
\textit{Abstract Security Tag } \boldsymbol{\sigma} &::= 🔒 \mid 🔓\\
\textit{Abstract Pre-Events } \boldsymbol{a}_b &::= \textbf{Any}\\
\textit{Abstract Events } \boldsymbol{a} &::= \boldsymbol{\varepsilon} \mid \boldsymbol{a}_b; \boldsymbol{\sigma} \mid \lightning
\end{aligned}$$

Figure 11: sCCT Monitor.

The `sCCT` monitor serves as an overapproximation for cryptographic constant-time code. Here, any event whatsoever is considered bad, since this leaks information by means of a sidechannel.

Definition 36 (sCCT)**.**

$$\text{scct} := \left\{ \overline{a_{\text{ct}}} \;\middle|\; \overline{a_{\text{ct}}} = [\cdot] \;\; \textit{or} \;\; \exists \overline{a'_{\text{ct}}}, \overline{a_{\text{ct}}} = a^{\text{ct}}_{b}; \text{🔓} \cdot \overline{a'_{\text{ct}}} \land \overline{a'_{\text{ct}}} \in \text{scct} \;\; \right\}$$

$\boxed{\vdash T_{\text{sCCT}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{sCCT}}'}$ „Given abstract state T_{sCCT}, step to abstract state T_{sCCT}' emitting an abstract event $\boldsymbol{a}$."

$\boxed{\vdash T_{\text{sCCT}} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}'}$ „Reflexive-Transitive closure of above judgement. $\overline{\boldsymbol{a}}$ is a list of abstract events."

$$\text{(scct-None)}\;\frac{}{\vdash T_{\text{sCCT}} \overset{\boldsymbol{\varepsilon}}{\rightsquigarrow} T_{\text{sCCT}}} \qquad \text{(scct-Abort)}\;\frac{}{\vdash T_{\text{sCCT}} \overset{\lightning;\text{🔓}}{\rightsquigarrow} T_{\text{sCCT}}}$$

$$\text{(scct-Refl)}\;\frac{}{\vdash T_{\text{sCCT}} \overset{[\cdot]}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}} \qquad \text{(scct-Trans)}\;\frac{\vdash T_{\text{sCCT}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{sCCT}}' \qquad \vdash T_{\text{sCCT}}' \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}''}{\vdash T_{\text{sCCT}} \overset{\boldsymbol{a}\cdot\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}''}$$

Figure 12: Steps of sCCT Monitor.

$\boxed{\underline{\sigma} = \boldsymbol{\sigma}}$ „Equality of Security Tags."

$$\text{(secure-eq)}\;\frac{}{\underline{\text{🔒}} = \text{🔒}} \qquad \text{(insecure-eq)}\;\frac{}{\underline{\text{🔓}} = \text{🔓}}$$

$\boxed{a^{\text{ms}} \cong \boldsymbol{a}}$ „Memory-Safety specific event a^{ms} describes same action as $\boldsymbol{a}$."

$$\text{(sCCT-Public-Authentic)}\;\frac{}{a^{\text{ms}}_{b}; t; \underline{\text{🔓}} \cong \boldsymbol{\varepsilon}} \qquad \text{(sCCT-Use-Authentic)}\;\frac{}{\underline{\mathit{Use}\ \ell\ n}; t; \underline{\text{🔒}} \cong \mathbf{Any}; \text{🔒}} \qquad \text{(sCCT-Abort-Authentic)}\;\frac{}{\underline{\lightning} \cong \lightning}$$

$$\text{(sCCT-None-Authentic)}\;\frac{}{\underline{\varepsilon} \cong \boldsymbol{\varepsilon}} \qquad \text{(sCCT-Branch-Authentic)}\;\frac{}{\underline{\mathit{Branch}\ n}; t; \underline{\text{🔒}} \cong \mathbf{Any}; \text{🔒}}$$

$$\text{(sCCT-Binop-Authentic)}\;\frac{}{\underline{\mathit{Binop}\ n}; t; \underline{\text{🔒}} \cong \mathbf{Any}; \text{🔒}}$$

$\boxed{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}$ „Memory-Safety specific trace $\overline{a^{\text{ms}}}$ describes same trace as $\overline{\boldsymbol{a}}$."

$$\text{(sCCT-Refl-Authentic)}\;\frac{}{[\cdot] \cong^{*} [\cdot]} \qquad \text{(sCCT-Trans-Authentic)}\;\frac{a^{\text{ms}} \cong \boldsymbol{a} \qquad \overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{a^{\text{ms}} \cdot \overline{a^{\text{ms}}} \cong^{*} \boldsymbol{a} \cdot \overline{\boldsymbol{a}}} \qquad \text{(sCCT-Trans-Authentic-Lign)}\;\frac{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\underline{\varepsilon} \cdot \overline{a^{\text{scct}}} \cong^{*} \overline{\boldsymbol{a}}}$$

$$\text{(sCCT-Trans-Authentic-Rign)}\;\frac{\overline{a^{\text{ms}}} \cong^{*} \overline{\boldsymbol{a}}}{\overline{a^{\text{ms}}} \cong^{*} \boldsymbol{\varepsilon} \cdot \overline{\boldsymbol{a}}}$$

Figure 13: Trace Agreement.

Definition 37 (Trace is strictly cryptographic constant time via monitor.)**.** *We say* $\mathtt{sCCT}(\overline{a^{ms}})$ *iff* $\exists \overline{\boldsymbol{a}}\ T_{sCCT}$ *such that:*

- $\overline{a^{ms}} \cong \overline{\boldsymbol{a}}$
- *and* $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{sCCT}$

Lemma 18 ($\mathtt{SCCT}(\overline{a^{\text{ms}}})$ implies $\overline{a^{\text{ms}}} \in \text{scct}$)**.** *If*

(a) $\mathtt{SCCT}(\overline{a^{ms}})$

Then

(i) $\overline{a^{ms}} \in \text{scct}$

Proof. Invert Assumption (a):

(H_1) $\overline{a^{\text{ms}}} \cong \overline{\boldsymbol{a}}$

(H_2) $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}$

The proof follows with an induction on Assumption (H_2). $\square$

4.1.5 SS Monitor

$$
\begin{aligned}
\textit{Abstract Store } T_{\faGhost} &= \emptyset \\
\textit{Abstract SPECTRE Label } \boldsymbol{vX} &::= \textbf{NONE} \mid \textbf{PHT} \\
\textit{Abstract Security Tag } \boldsymbol{\sigma} &::= \faLock_{\boldsymbol{vX}} \mid \faUnlock \\
\textit{Abstract Pre-Events } \boldsymbol{a_b} &::= \textbf{Any} \\
\textit{Abstract Events } \boldsymbol{a} &::= \boldsymbol{\varepsilon} \mid \boldsymbol{a_b}; \boldsymbol{\sigma} \mid \lightning
\end{aligned}
$$

Figure 14: SS Monitor.

The spec monitor serves as an overapproximation for speculative non-interference.

Definition 38 (SS)**.**

$$
\text{spec} := \left\{ \overline{a_{\faGhost}} \;\middle|\; \begin{array}{l} \overline{a_{\faGhost}} = [\cdot] \;\; \textit{or} \; \exists \overline{a'_{\faGhost}}. \\ \left(\overline{a_{\faGhost}} = a_b^{\faGhost}; \faUnlock \cdot \overline{a'_{\faGhost}} \,\textit{or}\; \overline{a_{\faGhost}} = a_b^{\faGhost}; \faLock_{NONE} \cdot \overline{a'_{\faGhost}} \right) \\ \textit{and} \;\; \overline{a'_{\faGhost}} \in \text{spec} \end{array} \right\}
$$

$\boxed{\vdash T_{\faGhost} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\faGhost}'}$ „Given abstract state $T_{\faGhost}$, step to abstract state $T_{\faGhost}'$ emitting an abstract event $\boldsymbol{a}$."

$\boxed{\vdash T_{\faGhost} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\faGhost}'}$ „Reflexive-Transitive closure of above judgement. $\overline{\boldsymbol{a}}$ is a list of abstract events."

$$
\text{(spec-None)}\;\frac{}{\vdash T_{\faGhost} \overset{\boldsymbol{\varepsilon}}{\rightsquigarrow} T_{\faGhost}}
\qquad
\text{(spec-Leak-None)}\;\frac{}{\vdash T_{\faGhost} \overset{\boldsymbol{a_b}; \faLock_{\textbf{NONE}}}{\rightsquigarrow} T_{\faGhost}}
\qquad
\text{(spec-Abort)}\;\frac{}{\vdash T_{\faGhost} \overset{\lightning; \faUnlock}{\rightsquigarrow} T_{\faGhost}}
$$

$$
\text{(spec-Refl)}\;\frac{}{\vdash T_{\faGhost} \overset{[\cdot]}{\rightsquigarrow}{}^{*} T_{\faGhost}}
\qquad
\text{(spec-Trans)}\;\frac{\vdash T_{\faGhost} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\faGhost}' \qquad \vdash T_{\faGhost}' \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\faGhost}''}{\vdash T_{\faGhost} \overset{\boldsymbol{a} \cdot \overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\faGhost}''}
$$

Figure 15: Steps of SS Monitor.

$\boxed{\underline{\sigma} = \boldsymbol{\sigma}}$ „Equality of Security Tags."

$$\frac{}{\underline{\faLock} = \faLock_{\boldsymbol{vX}}}\ \text{(spec-secure-eq)} \qquad \frac{}{\underline{\faUnlock} = \faUnlock}\ \text{(spec-insecure-eq)}$$

$\boxed{a^{\text{ms}} \cong \boldsymbol{a}}$ „Memory-Safety specific event a^{ms} describes same action as $\boldsymbol{a}$."

$$\frac{}{a_b^{\text{ms}}; t; \underline{\faUnlock} \cong \boldsymbol{\varepsilon}}\ \text{(spec-Public-Authentic)} \qquad \frac{}{\underline{Use\ \ell\ n}; t; \underline{\faLock} \cong \textbf{Any}; \faLock}\ \text{(spec-Use-Authentic)} \qquad \frac{}{\underline{\lightning} \cong \lightning}\ \text{(spec-Abort-Authentic)}$$

$$\frac{}{\underline{\varepsilon} \cong \boldsymbol{\varepsilon}}\ \text{(spec-None-Authentic)} \qquad \frac{}{\underline{Branch\ n}; t; \underline{\faLock} \cong \textbf{Any}; \faLock}\ \text{(spec-Branch-Authentic)}$$

$$\frac{}{\underline{Binop\ n}; t; \underline{\faLock} \cong \textbf{Any}; \faLock}\ \text{(spec-Binop-Authentic)}$$

$\boxed{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}$ „Memory-Safety specific trace $\overline{a^{\text{ms}}}$ describes same trace as $\overline{\boldsymbol{a}}$."

$$\frac{}{[\cdot] \cong^* [\cdot]}\ \text{(spec-Refl-Authentic)} \qquad \frac{a^{\text{ms}} \cong \boldsymbol{a} \qquad \overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{a^{\text{ms}} \cdot \overline{a^{\text{ms}}} \cong^* \boldsymbol{a} \cdot \overline{\boldsymbol{a}}}\ \text{(spec-Trans-Authentic)} \qquad \frac{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{\underline{\varepsilon} \cdot \overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}\ \text{(spec-Trans-Authentic-Lign)}$$

$$\frac{\overline{a^{\text{ms}}} \cong^* \overline{\boldsymbol{a}}}{\overline{a^{\text{ms}}} \cong^* \boldsymbol{\varepsilon} \cdot \overline{\boldsymbol{a}}}\ \text{(spec-Trans-Authentic-Rign)}$$

Figure 16: Trace Agreement.

Definition 39 (Trace is strictly cryptographic constant time via monitor.)**.** *We say* $\texttt{SPEC}(\overline{a^{ms}})$ *iff* $\exists \overline{\boldsymbol{a}}\ T_{\text{\faBell}}$ *such that:*

- $\overline{a^{ms}} \cong \overline{\boldsymbol{a}}$
- *and* $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{\text{\faBell}}$

Lemma 19 ($\texttt{SPEC}(\overline{a^{\text{ms}}})$ implies $\overline{a^{\text{ms}}} \in$ spec)**.** *If*

(a) $\texttt{spec}(\overline{a^{ms}})$

Then

(i) $\overline{a^{ms}} \in$ spec

Proof. Invert Assumption (a):

(H_1) $\overline{a^{\text{ms}}} \cong \overline{\boldsymbol{a}}$

(H_2) $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{\text{\faBell}}$

The proof follows with an induction on Assumption (H_2). $\square$

4.2 Source Language

4.2.1 Syntax

$$
\begin{array}{l}
\textit{Final Result } \mathsf{f} ::= \mathsf{v} \mid \mathsf{x} \qquad \textit{May be a Result } \mathsf{f}_{\not\downarrow} ::= \mathsf{f} \mid \mathsf{stuck} \\
\textit{Expressions } \mathsf{e} ::= \mathsf{f}_{\not\downarrow} \mid \mathsf{e_1} \oplus \mathsf{e_2} \mid \mathsf{x[e]} \mid \mathsf{let\ x = e_1\ in\ e_2} \mid \mathsf{x[e_1]} \leftarrow \mathsf{e_2} \\
\qquad \mid \mathsf{let\ x = new\ e_1\ in\ e_2} \mid \mathsf{delete\ x} \mid \mathsf{return\ e} \mid \mathsf{call\ foo\ e} \\
\qquad \mid \mathsf{ifz\ e_1\ then\ e_2\ else\ e_3} \quad \text{where } \oplus \in \{+,-,\times,<\} \\
\textit{Functions } \mathsf{F} ::= \mathsf{let\ foo\ x} : \tau_\lambda := \mathsf{e} \\
\textit{Expr. Types } \tau_\mathsf{e} ::= \mathbb{N} \mid \mathsf{ref_q}\ \mathbb{N} \quad \textit{Qualifier } \mathsf{q} ::= 1 \mid {}^1\!/_2 \quad \textit{Values } \mathsf{v} ::= \mathsf{n} \in \mathbb{N} \\
\textit{Ectx Types } \tau_\lambda ::= \tau_\mathsf{e} \to \tau_\mathsf{e} \quad \textit{Types } \tau ::= \tau_\mathsf{e} \mid \tau_\lambda \mid \tau_\mathsf{e} \to \bot \quad \textit{References } \ell \in \mathbb{N} \\
\textit{Eval.Ctx. } \mathsf{K} ::= [\cdot] \mid \mathsf{K} \oplus \mathsf{e} \mid \mathsf{v} \oplus \mathsf{K} \mid \mathsf{x[K]} \mid \mathsf{let\ x = K\ in\ e} \\
\qquad \mid \mathsf{x[K]} \leftarrow \mathsf{e} \mid \mathsf{x[v]} \leftarrow \mathsf{K} \mid \mathsf{let\ x = new\ K\ in\ e} \\
\qquad \mid \mathsf{ifz\ K\ then\ e_1\ else\ e_2} \mid \mathsf{call\ foo\ K} \mid \mathsf{return\ K} \\
\textit{Variables } \mathsf{x} \mid \mathsf{y} \mid \mathsf{foo} \mid \ldots \quad \textit{Poison } \rho ::= \square \mid ☣ \\
\textit{Sandbox Tag } \mathsf{t} ::= \mathsf{ctx} \mid \mathsf{comp} \\
\textit{Typing.Env. } \Gamma ::= [\cdot] \mid \Gamma, \mathsf{x} : \tau \quad \textit{Store } \Delta ::= [\cdot] \mid \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta \\
\textit{Communication } \mathbf{c} ::= ? \mid ! \mid \varnothing \quad \textit{Heaps } \mathsf{H} ::= [\cdot] \mid \mathsf{H} :: \mathsf{n} \\
\textit{Cont. Stack } \overline{\mathsf{K}} ::= [\cdot] \mid (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}} \quad \textit{Library } \Xi ::= [\cdot] \mid \mathsf{F}, \Xi \\
\textit{Relevant } \xi ::= [\cdot] \mid \mathsf{foo}, \xi \quad \textit{State } \Omega ::= \Phi; \mathsf{t}; \Psi \\
\textit{Flow State } \Phi ::= \xi; \Xi; \overline{\mathsf{K}} \quad \textit{Memory State } \Psi ::= \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta \\
\textit{Programs } \mathsf{prog}\ \Xi_\mathsf{ctx}\ \Xi_\mathsf{comp} \quad \textit{Substitutions } \gamma ::= [\mathsf{v}/\mathsf{x}], \gamma \mid [\cdot]
\end{array}
$$

Figure 17: Syntax of $\mathrm{L_{tms}}$

Most of the syntax is more or less standard. Qualifiers q can be attached to pointers to signal ownership. Hereby, the qualifier 1 means "fully owned", meaning we may and also must delete the pointer at some point, while we cannot do the same for ${}^1\!/_2$, which forbids us to delete. Poison ρ marks locations in the execution-context Ω as „to-be-deleted” ($\square$) or „deleted” (☣). ξ contains a list of identifiers referencing what functions are considered a component. Anything not present in ξ is considered to be a context. We describe the act of calling or returning from component to context or vice versa as „crossing the boundary”. The heap is split into two parts, such that any component listed in ξ will allocate in $\mathsf{H^{comp}}$ and anything else in $\mathsf{H^{ctx}}$. Objects pointed at by a location ℓ are marshalled when they are passed across the boundary. Without the heap sandboxing, contexts can arbitrarily rewrite a component's memory, which incurs compiler correctness issues. We chose to distinguish between successful final results (f) and potentially crashed results ($\mathsf{f}_{\not\downarrow}$). Note that a final result may also be an identifier, which we keep around as abstract representation for locations of pointers. However, we still distinct them from ordinary values (v), since they don't behave like normal values. For example, we cannot make canonical typing lemmas for them without the additional information contained in a non-empty typing-context. When discussing secure compilation, there is a notion of „context” and „component”, where the latter is usually the part one cares about. We use the term „context-switching” with its usual meaning, i.e. change in control flow to some other procedure. For **c**, the representation $?$ signalizes a context switch from context to component, $!$ from component to context, and $\varnothing$ signals an internal change, either inside the context or inside the component. The state Ω carries information on what the current context is and the continuation stack $\overline{\mathsf{K}}$ is used in the semantics to mark events with above policy accordingly. Continuations K in the stack are annotated with the name of the function foo the continuation originates from. Programs $\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}}$ contain two lists of top-level definitions Ξ_{ctx} and Ξ_{comp}. Ξ_{ctx} takes the role as attacker code. We reason explicitly about substitutions in the dynamic semantics.

4.2.2 Static Semantics

$\boxed{\vdash \tau\ \mathrm{int}}$ „τ is an interface type."

$$\frac{}{\vdash \mathbb{N}\ \mathrm{int}}\ (\mathsf{int}\text{-}\mathbb{N})$$

Figure 18: Interface types of $\mathrm{L_{tms}}$.

We introduce interface types to explicitly disallow passing owned pointers.

$\boxed{\Gamma \equiv \Gamma_1 \circ \Gamma_2}$ „Environment Γ can be split into Γ_1 and Γ_2."

$$\frac{}{[\cdot] \equiv [\cdot] \circ [\cdot]}\ (\mathsf{splitEmpty}) \qquad \frac{}{\Gamma \equiv [\cdot] \circ \Gamma}\ (\mathsf{splitEmptyL}) \qquad \frac{}{\Gamma \equiv \Gamma \circ [\cdot]}\ (\mathsf{splitEmptyR})$$

$$\frac{\Gamma \equiv \Gamma_1 \circ \Gamma_2}{\mathsf{x} : \mathbb{N}, \Gamma \equiv \mathsf{x} : \mathbb{N}, \Gamma_1 \circ \mathsf{x} : \mathbb{N}, \Gamma_2}\ (\mathbb{N}\mathsf{split})$$

$$\frac{\Gamma \equiv \Gamma_1 \circ \Gamma_2}{\mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N}, \Gamma \equiv \mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N}, \Gamma_1 \circ \mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N}, \Gamma_2}\ (\mathsf{weakPtrSplit})$$

$$\frac{\Gamma \equiv \Gamma_1 \circ \Gamma_2}{\mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma \equiv \mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma_1 \circ \Gamma_2}\ (\mathsf{ptrLSplit})$$

$$\frac{\Gamma \equiv \Gamma_1 \circ \Gamma_2}{\mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma \equiv \mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N}, \Gamma_1 \circ \mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma_2}\ (\mathsf{ptrRSplit})$$

$$\frac{\vdash \tau_{\mathsf{e}}^{(1)}\ \mathrm{int} \qquad \vdash \tau_{\mathsf{e}}^{(2)}\ \mathrm{int} \qquad \Gamma \equiv \Gamma_1 \circ \Gamma_2}{\mathsf{foo} : \tau_{\mathsf{e}}^{(1)} \to \tau_{\mathsf{e}}^{(2)}, \Gamma \equiv \mathsf{foo} : \tau_{\mathsf{e}}^{(1)} \to \tau_{\mathsf{e}}^{(2)}, \Gamma_1 \circ \mathsf{foo} : \tau_{\mathsf{e}}^{(1)} \to \tau_{\mathsf{e}}^{(2)}, \Gamma_2}\ (\mathsf{arrowSplit})$$

Figure 19: Context Splitting of $\mathrm{L_{tms}}$ typing contexts.

The splitting of contexts takes care to propagate owned-pointers towards the end. This way, no non-owned pointer occurs in the context after an owned one, where both have the same identifier. Note that non-owned pointers and values may be freely duplicated and that we can generate non-owned pointers if we have ownership.

Definition 40 (NoOwnedPtr)**.** *We write* $\mathit{NoOwnedPtr}\ \Gamma$ *iff for any* x, τ*, if* $\mathsf{x} : \tau \in \Gamma$*, then* $\tau \neq \mathsf{ref}_1\ \mathbb{N}$.

$\boxed{\Gamma \vdash \mathsf{e} : \tau}$ „Under environment Γ the expression e has type τ."

$$\frac{\text{NoOwnedPtr } \Gamma_1, \mathsf{x} : \tau, \Gamma_2}{\Gamma_1, \mathsf{x} : \tau, \Gamma_2 \vdash \mathsf{x} : \tau}\;(t-\mathsf{var}) \qquad \frac{\text{NoOwnedPtr } \Gamma}{\Gamma \vdash \mathsf{n} : \mathbb{N}}\;(t-\mathbb{N})$$

$$\frac{\Gamma_1 \vdash \mathsf{e}_1 : \mathbb{N} \quad \Gamma_2 \vdash \mathsf{e}_2 : \mathbb{N}}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{e}_1 \oplus \mathsf{e}_2 : \mathbb{N}}\;(t-\oplus) \qquad \frac{\Gamma_2 \vdash \mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N} \quad \Gamma_1 \vdash \mathsf{e} : \mathbb{N}}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{x}[\mathsf{e}] : \mathbb{N}}\;(t-\mathsf{get})$$

$$\frac{\Gamma_3 \vdash \mathsf{x} : \mathsf{ref}_{1/2}\ \mathbb{N} \quad \Gamma_1 \vdash \mathsf{e}_1 : \mathbb{N} \quad \Gamma_2 \vdash \mathsf{e}_2 : \mathbb{N}}{(\Gamma_1 \circ \Gamma_2) \circ \Gamma_3 \vdash \mathsf{x}[\mathsf{e}_1] \leftarrow \mathsf{e}_2 : \mathbb{N}}\;(t-\mathsf{set})$$

$$\frac{\Gamma_1 \vdash \mathsf{e}_1 : \tau_{\mathsf{e}}^{(1)} \quad \mathsf{x} : \tau_{\mathsf{e}}^{(1)}, \Gamma_2 \vdash \mathsf{e}_2 : \tau^{(2)}}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2 : \tau^{(2)}}\;(t-\mathsf{let})$$

$$\frac{\Gamma_1 \vdash \mathsf{e}_1 : \mathbb{N} \quad \mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma_2 \vdash \mathsf{e}_2 : \mathbb{N}}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{let}\ \mathsf{x} = \mathsf{new}\ \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2 : \mathbb{N}}\;(t-\mathsf{new})$$

$$\frac{\text{NoOwnedPtr } \Gamma_1 \quad \text{NoOwnedPtr } \Gamma_2}{\Gamma_1, \mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma_2 \vdash \mathsf{delete}\ \mathsf{x} : \mathbb{N}}\;(t-\mathsf{delete})$$

$$\frac{\begin{array}{cc} \vdash \tau_{\mathsf{e}}^{(1)}\ \text{int} & \vdash \tau_{\mathsf{e}}^{(2)}\ \text{int} \\ \Gamma \vdash \mathsf{foo} : \tau_{\mathsf{e}}^{(1)} \to \tau_{\mathsf{e}}^{(2)} & \Gamma \vdash \mathsf{e} : \tau_{\mathsf{e}}^{(1)} \end{array}}{\Gamma \vdash \mathsf{call}\ \mathsf{foo}\ \mathsf{e} : \tau_{\mathsf{e}}^{(2)}}\;(t-\mathsf{call}) \qquad \frac{\tau_{\mathsf{e}}\ \text{int} \quad \Gamma \vdash \mathsf{e} : \tau_{\mathsf{e}}}{\Gamma \vdash \mathsf{return}\ \mathsf{e} : \tau_{\mathsf{e}} \to \bot}\;(t-\mathsf{return})$$

$$\frac{\Gamma_1 \vdash \mathsf{e}_1 : \mathbb{N} \quad \Gamma_2 \vdash \mathsf{e}_2 : \tau \quad \Gamma_2 \vdash \mathsf{e}_3 : \tau}{\Gamma_1 \circ \Gamma_2 \vdash \mathsf{ifz}\ \mathsf{e}_1\ \mathsf{then}\ \mathsf{e}_2\ \mathsf{else}\ \mathsf{e}_3 : \tau}\;(t-\mathsf{ifz}) \qquad \frac{}{\Gamma \vdash \mathsf{abort}() : \tau}\;(t-\mathsf{abort})$$

Figure 20: Checking of $\mathrm{L}_{\mathrm{tms}}$ expressions.

The context splitting takes care that for e.g. Rule $t-\oplus$ it cannot happen that we do something like $(\mathsf{delete}\ \mathsf{x}) + \mathsf{x}[0]$. For Rules $t-\mathsf{var}$ and $t-\mathbb{N}$ we require that the contexts do not contain any owned pointer. Intuitively, an owned pointer is useless upto getting a non-owned version from context splitting, because we cannot do anything with it besides deleting. This is inspired by linear logic.

$\boxed{\Gamma \vdash K : \tau^{(1)} \to \tau^{(2)}}$ „Under environment Γ the evaluation context K requires an object of type $\tau^{(1)}$ and yields one of type $\tau^{(2)}$."

$$\frac{}{\Gamma \vdash [\cdot] : \tau \to \tau}\ (\text{et-}[\cdot]) \qquad \frac{\Gamma \vdash v : \mathbb{N} \quad \Gamma \vdash K : \tau \to \mathbb{N}}{\Gamma \vdash v \oplus K : \tau \to \mathbb{N}}\ (\text{et-}\oplus\text{R})$$

$$\frac{\Gamma \vdash K : \tau \to \mathbb{N} \quad \Gamma \vdash e : \mathbb{N}}{\Gamma \vdash K \oplus e : \tau \to \mathbb{N}}\ (\text{et-}\oplus\text{L}) \qquad \frac{\Gamma \vdash x : \mathsf{ref}_{1/2}\ \mathbb{N} \quad \Gamma \vdash K : \tau \to \mathbb{N}}{\Gamma \vdash x[K] : \tau \to \mathbb{N}}\ (\text{et-get})$$

$$\frac{\Gamma \vdash v : \mathbb{N} \quad \Gamma \vdash K : \tau \to \mathbb{N}}{\Gamma \vdash x[v] \leftarrow K : \tau \to \mathbb{N}}\ (\text{et-setR}) \qquad \frac{\Gamma \vdash K : \tau \to \mathbb{N} \quad \Gamma \vdash e : \mathbb{N}}{\Gamma \vdash x[K] \leftarrow e : \tau \to \mathbb{N}}\ (\text{et-setL})$$

$$\frac{\Gamma \vdash K : \tau^{(1)} \to \tau'_e \quad x : \tau'_e, \Gamma \vdash e : \tau^{(2)}}{\Gamma \vdash \mathsf{let}\ x = K\ \mathsf{in}\ e : \tau^{(1)} \to \tau^{(2)}}\ (\text{et-let})$$

$$\frac{\Gamma \vdash K : \tau^{(1)} \to \mathbb{N} \quad x : \mathsf{ref}_1\ \mathbb{N}, \Gamma \vdash e : \tau^{(2)}}{\Gamma \vdash \mathsf{let}\ x = \mathsf{new}\ K\ \mathsf{in}\ e : \tau^{(1)} \to \tau^{(2)}}\ (\text{et-new})$$

$$\frac{\Gamma \vdash K : \tau^{(1)} \to \mathbb{N} \quad \Gamma \vdash e_2 : \tau^{(2)} \quad \Gamma \vdash e_3 : \tau^{(2)}}{\Gamma \vdash \mathsf{ifz}\ K\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : \tau^{(1)} \to \tau^{(2)}}\ (\text{et-ifz})$$

$$\frac{\Gamma \vdash K : \tau^{(1)} \to \tau' \quad \Gamma \vdash \mathsf{foo} : \tau' \to \tau^{(2)}}{\Gamma \vdash \mathsf{call}\ \mathsf{foo}\ K : \tau^{(1)} \to \tau^{(2)}}\ (\text{et-call})$$

$$\frac{\Gamma \vdash K : \tau^{(1)} \to \tau^{(2)}}{\Gamma \vdash \mathsf{return}\ K : \tau^{(1)} \to (\tau^{(2)} \to \bot)}\ (\text{et-ret})$$

Figure 21: Checking of L_{tms} evaluation contexts.

$\boxed{\Xi \downarrow = \Gamma}$ „Project L_{tms} library to static typing context."

$$\frac{}{[\cdot] \downarrow = [\cdot]}\ (\Xi\text{-proj-}[\cdot]) \qquad \frac{\Xi \downarrow = \Gamma}{(\mathsf{let}\ \mathsf{foo}\ x : \tau_\lambda := e), \Xi \downarrow = (\mathsf{foo} \mapsto \tau_\lambda), \Gamma}\ (\Xi\text{-proj-cons})$$

$\boxed{\mathrm{dom}\ \ \Xi = \mathsf{foo}, \ldots, \mathsf{bar}}$ „Collect function names."

$$\frac{}{\mathrm{dom}\ \ [\cdot] = [\cdot]}\ (\Xi\text{-dom-}[\cdot]) \qquad \frac{\mathrm{dom}\ \ \Xi = D}{\mathrm{dom}\ (\mathsf{let}\ \mathsf{foo}\ x : \tau_\lambda := e), \Xi = \mathsf{foo}, D}\ (\Xi\text{-dom-cons})$$

Figure 22: Extracting type annotations and function names.

$$\boxed{\Xi \equiv \Xi_1 \cup \Xi_2} \quad \text{„Merging } \mathrm{L_{tms}} \text{ libraries.''}$$

$$\frac{\Xi = \Xi_1}{\Xi \equiv \Xi_1 \cup [\cdot]}\ (\text{lib-merge-empty}) \qquad \frac{\Xi \equiv \Xi_1 \cup \Xi_2}{\mathsf{F}, \Xi \equiv \Xi_1 \cup \mathsf{F}, \Xi_2}\ (\text{lib-merge-cons})$$

$$\boxed{\Xi_1 \bowtie \Xi_2 = \Xi} \quad \text{„Syntactically linking } \mathrm{L_{tms}} \text{ libraries.''}$$

$$\frac{\mathrm{dom}\, \Xi_1 \cap \mathrm{dom}\, \Xi_2 = \emptyset \qquad \Xi \equiv \Xi_1 \cup \Xi_2}{\Xi_1 \bowtie \Xi_2 = \Xi}\ (\text{syntactic-plugging})$$

Figure 23: $\mathrm{L_{tms}}$ plugging of libraries.

$$\boxed{\Gamma \vdash \Xi \ \text{ok}} \quad \text{„} \mathrm{L_{tms}} \text{ library typechecks.''}$$

$$\frac{}{\Gamma \vdash [\cdot] \ \text{ok}}\ (\text{t-}\Xi\text{-empty})$$

$$\frac{\tau_e^{(1)} \ \text{int} \qquad \tau_e^{(2)} \ \text{int} \qquad \Gamma \vdash \Xi \ \text{ok} \qquad \mathsf{x} : \tau_e^{(1)}, \Gamma \vdash \mathsf{e} : \ \tau_e^{(2)} \to \bot}{\Gamma \vdash (\mathsf{let\ foo\ x} : \tau_e^{(1)} \to \tau_e^{(2)} := \mathsf{e}), \Xi \ \text{ok}}\ (\text{t-}\Xi\text{-cons})$$

$$\boxed{\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \Xi, \xi} \quad \text{„} \mathrm{L_{tms}} \text{ program } \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \text{ typechecks. } \Xi \text{ is the result of linking } \Xi_{\mathsf{ctx}} \text{ and } \Xi_{\mathsf{comp}}.\ \xi \text{ is dom } \Xi_{\mathsf{comp}}.\text{''}$$

$$\frac{\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}} \qquad \Xi \downarrow = \Gamma_0 \qquad \mathsf{main} \in \mathrm{dom}\, \Gamma_0 \qquad \Gamma_0 \vdash \Xi \ \text{ok} \qquad \Gamma_0 \vdash \mathsf{call\ main\ 0} : \ \mathbb{N} \to \mathbb{N}}{\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \Xi, \mathrm{dom}\ \Xi_{\mathsf{comp}}}\ (t-\mathsf{prog})$$

$$\boxed{\vdash \Omega \triangleright \mathsf{e} : \ \tau} \quad \text{„} \mathrm{L_{tms}} \text{ runtime program } \Omega \triangleright \mathsf{e} \text{ typechecks.''}$$

$$\frac{\Xi \vdash \Delta : \ \Gamma \qquad \Gamma \vdash \mathsf{e} : \ \tau}{\vdash \xi; \Xi; \overline{\mathsf{K}}; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \triangleright \mathsf{e} : \ \tau}\ (t-\mathsf{prog}-\mathsf{runtime})$$

Figure 24: Checking of $\mathrm{L_{tms}}$ contexts, components, programs, and whole programs.

Perhaps most interesting is Rule $t-\mathsf{prog}-\mathsf{runtime}$ which generates a suitable typing context from the execution context. Figure 25 shows how this works exactly.

$$\boxed{\Xi \vdash \Delta : \ \Gamma} \quad \text{„} \mathrm{L_{tms}} \text{ location map } \Delta \text{ yields static typing environment } \Gamma.\text{''}$$

$$\frac{\Xi \downarrow = \Gamma}{\Xi \vdash [\cdot] : \ \Gamma}\ (T\mathsf{empty}\Delta) \qquad \frac{\Xi \vdash \Delta : \ \Gamma}{\Xi \vdash \mathsf{x} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta : \ \mathsf{x} : \mathsf{ref}_1\ \mathbb{N}, \Gamma}\ (T\mathsf{ref}_1\ \mathbb{N})$$

$$\frac{\Xi \vdash \Delta : \ \Gamma}{\Xi \vdash \mathsf{x} \mapsto (\ell; \mathsf{t}; ☣; \mathsf{n}), \Delta : \ \Gamma}\ (T\mathsf{ref}_1\ \mathbb{N}\mathsf{poison})$$

Figure 25: $\mathrm{L_{tms}}$ store typing.

Here, we want to populate the typing context with all *valid* pointers, which are those that are not poisoned. Pointers of type $\mathsf{ref}_{1/2}\ \mathbb{N}$ are implicitly generated during elaboration, see Figure 19.

4.2.3 Dynamic Semantics

The language trivially satisfies sCCT. It does not satisfy SMS, as seen in Section 4.2.4 (Translation to Specification Events).

$$
\begin{aligned}
\textit{Base Events}\ \mathsf{a_b} ::=\ & \mathsf{Alloc}\ \ell\ \mathsf{v} \mid \mathsf{Dealloc}\ \ell \mid \mathsf{Get}\ \ell\ \mathsf{v} \mid \mathsf{Set}\ \ell\ \mathsf{v}\ \mathsf{v}' \\
& \mid \mathsf{Call}\ \mathbf{c}\ \mathsf{foo}\ \mathsf{v} \mid \mathsf{Ret}\ \mathbf{c}\ \mathsf{v} \mid \mathsf{Start} \mid \mathsf{End}\ \mathsf{v} \\
\textit{Events}\ \mathsf{a} ::=\ & \varepsilon \mid \mathsf{a_b}; \mathsf{t} \mid \lightning
\end{aligned}
$$

Figure 26: Events of $\mathrm{L_{tms}}$.

$\boxed{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}} \Omega' \triangleright \mathsf{e}'}$ „Expression e applied evaluates under configuration Ω to e' and new configuration Ω', emitting event a."

$$(e-\oplus)\quad \frac{\mathsf{n_1} \oplus \mathsf{n_2} = \mathsf{n_3}}{\Omega \triangleright \mathsf{n_1} \oplus \mathsf{n_2} \xrightarrow{\varepsilon} \Omega \triangleright \mathsf{n_3}}$$

$$(e-\mathsf{get}-\in)\quad \frac{\Psi = \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 \qquad \ell + \mathsf{n} \in \mathrm{dom}\ \Psi.\mathsf{H^t}}{\Phi; \mathsf{t}'; \Psi \triangleright \mathsf{x}[\mathsf{n}] \xrightarrow{\mathsf{Get}\ \ell\ \mathsf{n;t}} \Phi; \mathsf{t}'; \Psi \triangleright \mathsf{H^t}(\ell + \mathsf{n})}$$

$$(e-\mathsf{get}-\notin)\quad \frac{\Psi.\Delta = \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 \qquad \ell + \mathsf{n} \notin \mathrm{dom}\ \Psi.\mathsf{H^t}}{\Phi; \mathsf{t}'; \Psi \triangleright \mathsf{x}[\mathsf{n}] \xrightarrow{\mathsf{Get}\ \ell\ \mathsf{n;t}} \Phi; \mathsf{t}'; \Psi \triangleright 1729}$$

$$(e-\mathsf{set}-\in)\quad \frac{\begin{array}{cc}\Psi.\Delta = \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 & \ell + \mathsf{n} \in \mathrm{dom}\ \Psi.\mathsf{H^t} \\ \mathsf{H^t_1} = \Psi.\mathsf{H^t}(\ell + \mathsf{n} \mapsto \mathsf{v}) & \Psi' = \Psi[\mathsf{H^t} \leftarrow \mathsf{H^t_1}]\end{array}}{\Phi; \mathsf{t_0}; \Psi \triangleright \mathsf{x}[\mathsf{n}] \leftarrow \mathsf{v} \xrightarrow{\mathsf{Set}\ \ell\ \mathsf{n}\ \mathsf{v;t_0}} \Phi; \mathsf{t_0}; \Psi' \triangleright \mathsf{v}}$$

$$(e-\mathsf{set}-\notin)\quad \frac{\Psi.\Delta = \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 \qquad \ell + \mathsf{n} \notin \mathrm{dom}\ \Psi.\mathsf{H^t}}{\Phi; \mathsf{t_0}; \Psi \triangleright \mathsf{x}[\mathsf{n}] \leftarrow \mathsf{v} \xrightarrow{\mathsf{Set}\ \ell\ \mathsf{n}\ \mathsf{v;t_0}} \Phi; \mathsf{t_0}; \Psi' \triangleright \mathsf{v}}$$

$$(e-\mathsf{let}-\mathsf{f})\quad \frac{}{\Omega \triangleright \mathsf{let}\ \mathsf{x} = \mathsf{f}\ \mathsf{in}\ \mathsf{e} \xrightarrow{\varepsilon} \Omega \triangleright \mathsf{e}[\mathsf{f}/\mathsf{x}]}$$

$$(e-\mathsf{delete})\quad \frac{\begin{array}{c}\Psi = \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta_2 \\ \Psi = \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2\end{array}}{\Phi; \mathsf{t}; \Psi \triangleright \mathsf{delete}\ \mathsf{x} \xrightarrow{\mathsf{Dealloc}\ \ell\mathsf{;t}} \Phi; \mathsf{t}; \Psi' \triangleright 0}$$

$$(e-\mathsf{new})\quad \frac{\begin{array}{c}\Psi.\Delta \vdash \ell\ \textit{fresh} \qquad \Psi.\Delta \vdash \mathsf{z}\ \textit{fresh} \qquad \mathsf{H^t_1} = \Psi.\mathsf{H^t} \ll \mathsf{n} \\ \Psi' = \Psi[\mathsf{H^t} \leftarrow \mathsf{H^t_1}][\Delta \leftarrow \mathsf{z} \mapsto (\ell, \mathsf{t}, \square; \mathsf{n}), \Psi.\Delta]\end{array}}{\Phi; \mathsf{t}; \Psi \triangleright \mathsf{let}\ \mathsf{x} = \mathsf{new}\ \mathsf{n}\ \mathsf{in}\ \mathsf{e} \xrightarrow{\mathsf{Alloc}\ \ell\ \mathsf{n;t}} \Phi; \mathsf{t}; \Psi' \triangleright \mathsf{e}[\mathsf{z}/\mathsf{x}]}$$

$$(e-\mathsf{ifz}-\mathsf{true})\quad \frac{}{\Omega \triangleright \mathsf{ifz}\ 0\ \mathsf{then}\ \mathsf{e_1}\ \mathsf{else}\ \mathsf{e_2} \xrightarrow{\varepsilon} \Omega \triangleright \mathsf{e_1}}$$

$$(e-\mathsf{ifz}-\mathsf{false})\quad \frac{}{\Omega \triangleright \mathsf{ifz}\ \mathsf{S}(\mathsf{n})\ \mathsf{then}\ \mathsf{e_1}\ \mathsf{else}\ \mathsf{e_2} \xrightarrow{\varepsilon} \Omega \triangleright \mathsf{e_2}} \qquad (e-\mathsf{abort})\quad \frac{}{\Omega \triangleright \mathsf{abort}() \xrightarrow{\lightning} \lightning \triangleright \mathsf{stuck}}$$

Figure 27: Primitive Evaluation of $\mathrm{L_{tms}}$ expressions.

Evaluation is mostly straightforward, the only interesting cases involve the pointers. Specifically, Rule $e-\mathsf{delete}$ demonstrates why we need the poison-tag on the locations: Regardless of the current tag, emit a $\mathsf{Dealloc}\ \ell$ event and mark ℓ as poisoned (☣). The intuitive solution, removing the mapping on deletion, doesn't allow to run programs that delete twice. However, we want to model memory effects and show that such situations never happen, even though they *could*. Given a poisoned location, we can still do everything with it: reading, writing, or deletion. When generating the static context from the execution context, Rule $T\mathsf{ref}_1\ \mathbb{N}$poison allows us to disregard deleted locations which will help us reason that the given execution could not have been happening if the program was well-typed to begin with. For sake of readablity, we will omit $\gamma = [\cdot]$ when writing down evaluation steps. That is, whenever we have $\Omega \triangleright \mathsf{e}\,[\cdot] \xrightarrow{\mathsf{a}} \Omega' \triangleright \mathsf{e}'\,[\cdot]$, it's written $\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}} \Omega' \triangleright \mathsf{e}'$. Moreover, γ as postfix binds anything before it up to $\triangleright$, so $\mathsf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2\gamma$ is $(\mathsf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2)\gamma$.

$$\boxed{\Omega \triangleright K[e] \xrightarrow{a}_{ctx} \Omega' \triangleright K'[e']}$$ „Given an evaluation context K and an expression e it evaluates under configuration Ω to e' and new configuration Ω' in context K, emitting event a."

$$\frac{\Omega \triangleright e \xrightarrow{a} \Omega' \triangleright e'}{\Omega \triangleright K[e] \xrightarrow{a}_{ctx} \Omega' \triangleright K[e']}\ (e-\mathsf{ctx}) \qquad \frac{\Omega \triangleright e \xrightarrow{\lightning} \Omega' \triangleright \mathsf{stuck}}{\Omega \triangleright K[e] \xrightarrow{\lightning}_{ctx} \Omega' \triangleright \mathsf{stuck}}\ (e-\mathsf{ctx}-\mathsf{stuck})$$

$$\frac{\Omega = \xi; \Xi; [\cdot]\,; \mathsf{comp}; \Psi \quad \Xi = \Xi_1, (\mathsf{let\ main\ x} : \tau_\lambda := e), \Xi_2 \quad \Omega' = \xi; \Xi; \mathsf{ctx}; (K; \mathsf{main}), [\cdot]\,; \Psi}{\Omega \triangleright K[\mathsf{call\ main\ v}] \xrightarrow{\mathsf{Start;comp}}_{ctx} \Omega' \triangleright e[x/v]}\ (e-\mathsf{ctx}-\mathsf{call}-\mathsf{main})$$

$$\frac{\Omega = \xi; \Xi; \overline{K}; t; \Psi \quad \Xi = \Xi_1, (\mathsf{let\ foo\ x} : \tau_\lambda := e), \Xi_2 \quad \mathsf{foo} \in_{\neg t} \xi \quad \rho\,(t) = \mathbf{c} \quad \Omega' = \xi; \Xi; (K; \mathsf{foo}), \overline{K}; \neg t; \Psi}{\Omega \triangleright K[\mathsf{call\ foo\ v}] \xrightarrow{\mathsf{Call}\ \mathbf{c}\ \mathsf{foo\ v;t}}_{ctx} \Omega' \triangleright e[x/v]}\ (e-\mathsf{ctx}-\mathsf{call}-\mathsf{notsame})$$

$$\frac{\Omega = \xi; \Xi; \overline{K}; t; \Psi \quad \Xi = \Xi_1, (\mathsf{let\ foo\ x} : \tau_\lambda := e), \Xi_2 \quad \mathsf{foo} \in_{t} \xi \quad \Omega' = \xi; \Xi; (K; \mathsf{foo}), \overline{K}; t; \Psi}{\Omega \triangleright K[\mathsf{call\ foo\ v}] \xrightarrow{\mathsf{Call}\ \varnothing\ \mathsf{foo\ v;t}}_{ctx} \Omega' \triangleright e[x/v]}\ (e-\mathsf{ctx}-\mathsf{call}-\mathsf{same})$$

$$\frac{}{\xi; \Xi; [\cdot]^{\mathsf{main}}, [\cdot]\,; \mathsf{ctx}; \Psi \triangleright K'[\mathsf{return\ v}] \xrightarrow{\mathsf{End\ v;t}}_{ctx} \xi; \Xi; [\cdot]\,; \mathsf{comp}; \Psi \triangleright \mathsf{v}}\ (e-\mathsf{ctx}-\mathsf{return}-\mathsf{main})$$

$$\frac{\mathsf{foo} \in_{\neg t} \xi \quad \rho\,(t) = \mathbf{c}}{\xi; \Xi; (K; \mathsf{foo}), \overline{K}; t; \Psi \triangleright K'[\mathsf{return\ v}] \xrightarrow{\mathsf{Ret}\ \mathbf{c}\ \mathsf{v;t}}_{ctx} \xi; \Xi; \overline{K}; \neg t; \Psi \triangleright K[v]}\ (e-\mathsf{ctx}-\mathsf{return}-\mathsf{notsame})$$

$$\frac{\mathsf{foo} \in_{t} \xi}{\xi; \Xi; (K; \mathsf{foo}), \overline{K}; t; \Psi \triangleright K'[\mathsf{return\ v}] \xrightarrow{\mathsf{Ret}\ \varnothing\ \mathsf{v;t}}_{ctx} \xi; \Xi; \overline{K}; t; \Psi \triangleright K[v]}\ (e-\mathsf{ctx}-\mathsf{return}-\mathsf{same})$$

$$\boxed{\rho(t) = \mathbf{c}}$$ „Returns either ? or ! depending on t."

$$\frac{}{\rho(\mathsf{ctx}) = \,?}\ (\mathsf{comm\text{-}ctxtocomp}) \qquad \frac{}{\rho(\mathsf{comp}) = \,!}\ (\mathsf{comm\text{-}comptoctx})$$

$$\boxed{\neg t = t'}$$ „Negation of t."

$$\frac{}{\neg\mathsf{ctx} = \mathsf{comp}}\ (\mathsf{neg\text{-}ctx}) \qquad \frac{}{\neg\mathsf{comp} = \mathsf{ctx}}\ (\mathsf{neg\text{-}comp})$$

Figure 28: Contextual Evaluation of L_{tms} expressions.

$\boxed{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}'}$ „Expression e evaluates under configuration Ω to e′ and new configuration Ω′, emitting list of events ā."

$$\frac{}{\Omega \triangleright \mathsf{f} \xrightarrow{[\cdot]}{}^{*}_{\mathsf{ctx}} \Omega \triangleright \mathsf{f}}\ (es-\mathsf{refl})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}' \qquad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}'' \qquad \mathsf{a} \neq \varepsilon}{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}\cdot\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (es-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}' \qquad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (es-\mathsf{trans{-}unimportant})$$

$\boxed{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{n}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}'}$ „Expression e evaluates in n steps under configuration Ω to e′ and new configuration Ω′, emitting list of events ā."

$$\frac{}{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}{}^{0}_{\mathsf{ctx}} \Omega \triangleright \mathsf{e}}\ (en-\mathsf{refl})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}' \qquad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{n}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}'' \qquad \mathsf{a} \neq \varepsilon}{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}\cdot\bar{\mathsf{a}}}{}^{(n+1)}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (en-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}_{\mathsf{ctx}} \Omega' \triangleright \mathsf{e}' \qquad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{n}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{(n+1)}_{\mathsf{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (en-\mathsf{trans{-}unimportant})$$

Figure 29: Trace prefix generation given a $\mathrm{L}_{\mathrm{tms}}$ program using the reflexive-transitive closure.

$\boxed{\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \overset{\bar{\mathsf{a}}}{\Longrightarrow} \Omega \triangleright \mathsf{f}}$ „Run $\mathrm{L}_{\mathrm{tms}}$ program prog Ξ_{ctx} Ξ_{comp}, giving dynamic state Ω and emitting trace ā."

$$\frac{\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \xi, \Xi \qquad \mathsf{main} \notin \xi \qquad \xi; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathsf{ctx}} \Omega \triangleright \mathsf{f}}{\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \overset{\bar{\mathsf{a}}}{\Longrightarrow} \Omega \triangleright \mathsf{f}}\ (e-\mathsf{wprog})$$

$$\frac{\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \xi, \Xi \qquad \mathsf{main} \notin \xi \qquad \xi; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\bar{\mathsf{a}}\cdot\lightning\,;\mathsf{t}}{}^{*}_{\mathsf{ctx}} \Omega \triangleright \mathsf{f}_{\lightning}}{\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \xRightarrow{\bar{\mathsf{a}}\cdot\lightning\,;\mathsf{t}} \Omega \triangleright \mathsf{f}_{\lightning}}\ (e-\mathsf{wprog}-\lightning)$$

Figure 30: Running a whole $\mathrm{L}_{\mathrm{tms}}$ program.

4.2.4 Translation to Specification Events

We need a way to translate from concrete actions, as emitted by a program's execution, to abstract actions. The translation is, however, standard. The empty action $\underline{\varepsilon}$ is necessary. When proving lemmas by structural induction on the primitive steps (Figure 27), we get a concrete event ε that needs to be related to some a^{ms}. The natural candidate is $\underline{\varepsilon}$. We also use it to project the call/return events onto it.

$\boxed{\delta_{MS}(\ell) = \underline{\ell}}$ „A map from $\mathsf{L_{tms}}$ memory locations ℓ to specification locations $\underline{\ell}$."

$\boxed{\theta_{\delta_{MS}}(\mathsf{a}) = \overline{a^{\text{ms}}}}$ „Project an $\mathsf{L_{tms}}$ event to specification events."

$$\frac{\mathsf{a_b} \neq \lightning}{\theta_{\delta_{MS}}(\mathsf{a_b};\mathsf{ctx}) = \underline{\varepsilon}}\ (\text{filter-context}) \qquad \frac{}{\theta_{\delta_{MS}}(\mathsf{Start};\mathsf{comp}) = \underline{\varepsilon}}\ (\text{filter-comp-start})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathsf{n}}{\theta_{\delta_{MS}}(\mathsf{Alloc}\ \ell\ \mathsf{n};\mathsf{comp}) = \underline{Alloc\ \ell\ n}}\ (\text{filter-comp-alloc})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell}}{\theta_{\delta_{MS}}(\mathsf{Dealloc}\ \ell;\mathsf{comp}) = \underline{Dealloc\ \ell}}\ (\text{filter-comp-dealloc})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathsf{n}}{\theta_{\delta_{MS}}(\mathsf{Get}\ \ell\ \mathsf{n};\mathsf{comp}) = \underline{Use\ \ell\ n}}\ (\text{filter-comp-get}) \qquad \frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathsf{n}}{\theta_{\delta_{MS}}(\mathsf{Set}\ \ell\ \mathsf{n}\ \mathsf{v};\mathsf{comp}) = \underline{Use\ \ell\ n}}\ (\text{filter-comp-set})$$

$$\frac{}{\theta_{\delta_{MS}}(\mathsf{Call}\ \mathbf{c}\ \mathsf{foo}\ \mathsf{v};\mathsf{comp}) = \underline{\varepsilon}}\ (\text{filter-comp-call}) \qquad \frac{}{\theta_{\delta_{MS}}(\mathsf{Ret}\ \mathbf{c}\ \mathsf{v};\mathsf{comp}) = \underline{\varepsilon}}\ (\text{filter-comp-ret})$$

$$\frac{}{\theta_{\delta_{MS}}(\lightning;\mathsf{t}) = \underline{\lightning}}\ (\text{filter-abort})$$

$\boxed{\theta^*_{\delta_{MS}}(\overline{\mathsf{a}}) = \overline{a^{\text{ms}}}}$ „Project an $\mathsf{L_{tms}}$ trace to specification traces."

$$\frac{}{\theta^*_{\delta_{MS}}([\cdot]) = \underline{[\cdot]}}\ (\text{filter-empty}) \qquad \frac{\theta_{\delta_{MS}}(\mathsf{a}) = \underline{a} \qquad \theta^*_{\delta_{MS}}(\overline{\mathsf{a}}) = \overline{a^{\text{ms}}} \qquad \underline{a} \neq \underline{\varepsilon}}{\theta^*_{\delta_{MS}}(\mathsf{a}\cdot\overline{\mathsf{a}}) = \underline{a}\cdot\overline{a^{\text{ms}}}}\ (\text{filter-cons-relevant})$$

$$\frac{\theta_{\delta_{MS}}(\mathsf{a}) = \underline{\varepsilon} \qquad \theta^*_{\delta_{MS}}(\overline{\mathsf{a}}) = \overline{a^{\text{ms}}}}{\theta^*_{\delta_{MS}}(\mathsf{a}\cdot\overline{\mathsf{a}}) = \overline{a^{\text{ms}}}}\ (\text{filter-cons-relevant})$$

Figure 31: Projection of $\mathsf{L_{tms}}$ events to specification events.

$\boxed{\delta_{MS}(\ell) = \underline{\ell}}$ „Map $\mathsf{L_{tms}}$ locations ℓ to abstract locations $\underline{\ell}$."

$\boxed{T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta}$ „Abstract memory state T_{TMS} describes the concrete state Δ."

$$\frac{}{\emptyset \simeq_{\delta_{MS}} [\cdot]}\ (\text{Empty-Agree}) \qquad \frac{}{\emptyset \simeq_{\delta_{MS}} \lightning}\ (\text{Abort-Agree})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{\ell} \notin T_{\text{TMS}} \qquad T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta}{\{\underline{\ell}\} \cup T_{\text{TMS}} \simeq_{\delta_{MS}} \mathsf{x} \mapsto (\ell;\mathsf{t};\square;\mathsf{n}),\Delta}\ (\text{Cons-Agree})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta}{T_{\text{TMS}} \simeq_{\delta_{MS}} \mathsf{x} \mapsto (\ell;\mathsf{t};☣;\mathsf{n}),\Delta}\ (\text{Poison-Agree})$$

Figure 32: Store Agreement.

Statement of ?? 2 ($\mathsf{L_{tms}}$ is TMS via Monitor).If

(a) $\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \xRightarrow{\overline{\mathsf{a}}} \Omega \triangleright \mathsf{f}_{\lightning}$

Then

(i) $\theta^{*}_{\delta_{ms}}(\overline{\mathsf{a}}) \in \text{tmsafe}$

.

Theorem 1 ($\mathtt{L}_{\mathrm{tms}}$ is not spatially memory safe). *There is a well-typed $\mathtt{L}_{tms}$ component that does not robustly satisfy Definition 32: $\Xi \vDash_R smsafe$*

Proof. We pick:

$\Xi_{\mathsf{comp}} = \mathsf{let\ foo\ x : \mathbb{N} \to \mathbb{N} := let\ z = new\ x\ in\ let\ w = z[1337]\ in\ let\ \_ = delete\ z\ in\ w}, [\cdot]$

and $\Xi_{\mathsf{ctx}} = \mathsf{let\ main\ z : \mathbb{N} \to \mathbb{N} := call\ foo\ 42}, [\cdot]$. Let $\Xi \equiv \Xi_{\mathsf{comp}} \cup \Xi_{\mathsf{ctx}}$. We show $\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \Xi, \{\mathsf{foo}\}$. First note that $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$, since $\mathrm{dom}\ \Xi_{\mathsf{ctx}} \cap \mathrm{dom}\ \Xi_{\mathsf{comp}} = \emptyset$ and $\Xi \equiv \Xi_{\mathsf{ctx}} \cup \Xi_{\mathsf{comp}}$ by definition. Now, $\mathrm{dom}\ \Xi_{\mathsf{comp}} = \{\mathsf{foo}\}$ and $\Xi \downarrow = \mathsf{foo} \mapsto \mathbb{N} \to \mathbb{N}, \mathsf{main} \mapsto \mathbb{N} \to \mathbb{N}, [\cdot] = \Gamma_0$ by Rules Ξ-proj-$[\cdot]$ and Ξ-proj-cons. Consequently, $\mathsf{main} \in \mathrm{dom}\,\Gamma_0 = \{\mathsf{foo}, \mathsf{main}\}$. Note that Rule int-$\mathbb{N}$ gives $\mathbb{N}$ int. Using Rules t-Ξ-empty and t-Ξ-cons, what is left to show are the following:

1. $\mathsf{x} : \mathbb{N}, \Gamma_0 \vdash \mathsf{let\ z = new\ x\ in\ let\ w = z[1337]\ in\ delete\ z; w} : \mathbb{N}$

2. $\mathsf{z} : \mathbb{N}, \Gamma_0 \vdash \mathsf{call\ foo\ 42} : \mathbb{N}$

Let $\mathsf{e_w} = \mathsf{let\ w = z[1337]\ in\ delete\ z; w}$, $\Gamma = \mathsf{z} : \mathsf{ref_1}\ \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot]$, and $\Gamma' = \mathsf{z} : \mathsf{ref_{1/2}}\ \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot]$ due to space constraints.

$$\dfrac{\dfrac{\mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}}{\mathsf{x} : \mathbb{N}, [\cdot] \vdash \mathsf{x} : \mathbb{N}}\text{Rule } t - \mathsf{var} \qquad \dfrac{\Gamma' \vdash \mathsf{z[1337]} : \mathbb{N} \qquad \mathsf{w} : \mathbb{N}, \Gamma \vdash \mathsf{delete\ z; w} : \mathbb{N}}{\Gamma' \circ \Gamma \vdash \mathsf{e_w} : \mathbb{N}}\text{Rule } t - \mathsf{let}}{\mathsf{x} : \mathbb{N}, [\cdot] \circ \mathsf{x} : \mathbb{N}, [\cdot] \vdash \mathsf{let\ z = new\ x\ in\ let\ w = z[1337]\ in\ delete\ z; w} : \mathbb{N}}\text{Rule } t - \mathsf{new}$$

Observe that:

$$\dfrac{\dfrac{\mathsf{ref_{1/2}}\ \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N} \qquad \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}}{\Gamma' \vdash \mathsf{z} : \mathsf{ref_{1/2}}\ \mathbb{N}}\text{Rule } t - \mathsf{var} \qquad \dfrac{\mathsf{ref_{1/2}}\ \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N} \qquad \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}}{\Gamma' \vdash \mathsf{x} : \mathbb{N}}\text{Rule } t - \mathsf{var}}{\Gamma' \circ \Gamma' \vdash \mathsf{z[1337]} : \mathbb{N}}\text{Rule } t\text{–get}$$

And:

$$\dfrac{\dfrac{\mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N} \qquad \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}}{\mathsf{w} : \mathbb{N}, \mathsf{z} : \mathsf{ref_1}\ \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot] \vdash \mathsf{delete\ z} : \mathbb{N}}\text{Rule } t - \mathsf{delete} \qquad \dfrac{\mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N} \qquad \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}}{\mathsf{w} : \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot] \vdash \mathsf{w} : \mathbb{N}}\text{Rule } t - \mathsf{var}}{\mathsf{w} : \mathbb{N}, \mathsf{z} : \mathsf{ref_1}\ \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot] \circ \mathsf{w} : \mathbb{N}, \mathsf{x} : \mathbb{N}, [\cdot] \vdash \mathsf{delete\ z; w} : \mathbb{N}}\text{Rule } t - \mathsf{let}$$

Thus, it typechecks.

Now consider the other case:

$$\dfrac{\dfrac{}{\mathsf{z} : \mathbb{N}, \Gamma_0 \vdash \mathsf{foo} : \mathbb{N} \to \mathbb{N}}\text{Rule } t - \mathsf{var} \qquad \dfrac{}{\mathsf{z} : \mathbb{N}, \Gamma_0 \vdash 42 : \mathbb{N}}\text{Rule } t - \mathbb{N} \qquad \dfrac{}{\mathbb{N}\ \text{int}}\text{Rule int-}\mathbb{N}}{\mathsf{z} : \mathbb{N}, \Gamma_0 \vdash \mathsf{call\ foo\ 42} : \mathbb{N}}\text{Rule } t - \mathsf{call}$$

Note that $\mathsf{NoOwnedPtr\ z : \mathbb{N}, \Gamma_0}$ holds by $\mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}$ and $\mathbb{N} \to \mathbb{N} \neq \mathsf{ref_1}\ \mathbb{N}$
Running the whole program via Rule $e-$wprog yields trace prefix
$\overline{\mathsf{a}} = \mathsf{Start; comp{\cdot}Call\ !foo\ 42; ctx{\cdot}Alloc}\ \ell\ \mathsf{42; comp{\cdot}Get}\ \ell\ \mathsf{1337; comp{\cdot}Dealloc}\ \ell\mathsf{; comp{\cdot}Ret\ ?1729; comp \cdot End\ 1729; ctx}$. We omit the precise derivation for brevity, but note that Lemma 41 (Top-Level Progress) gives us the necessary execution.

Let $\delta_{ms} = \{\ell \mapsto \underline{\ell}\}$, by Figure 31, $\theta^*_{\delta_{ms}}(\overline{\mathsf{a}}) = \underline{Alloc\ \ell\ 42} \cdot \underline{Use\ \ell\ 1337} \cdot \underline{Dealloc\ \ell}$.

For Definition 32, note that $\underline{Alloc\ \ell\ 42} \in \theta^*_{\delta_{ms}}(\overline{\mathsf{a}})$ and $\underline{Use\ \ell\ 1337} \in \theta^*_{\delta_{ms}}(\overline{\mathsf{a}})$, but $1337 \not< 42$, hence $\Xi_{\mathsf{comp}} \not\models_R smsafe$. ☐

4.2.5 Auxiliary Definitions and Lemmas

Lemma 20 ($\to^n_{ctx}$ and $\to^*_{ctx}$ yield $\to^*_{ctx}$). *If*

(a) $\Omega \triangleright \mathsf{e} \xrightarrow{\overline{\mathsf{a}}}{}^n_{ctx} \Omega' \triangleright \mathsf{e'}$

(b) $\Omega' \triangleright \mathsf{e'} \xrightarrow{\overline{\mathsf{a'}}}{}^*_{ctx} \Omega'' \triangleright \mathsf{f}_{\lightning}$

Then

(i) $\Omega \triangleright \mathsf{e} \xrightarrow{\overline{\mathsf{a}} \cdot \overline{\mathsf{a'}}}{}^*_{ctx} \Omega'' \triangleright \mathsf{f}_{\lightning}$

Proof. Induction on Assumption (a). ☐

Lemma 21 ($\to^*_{ctx}$ splits into $\to^n_{ctx}$ and $\to^*_{ctx}$). *If*

(a) $\Omega \triangleright \mathsf{e} \xrightarrow{\overline{\mathsf{a_0}} \cdot \overline{\mathsf{a_1}}}{}^*_{ctx} \Omega' \triangleright \mathsf{f}_{\lightning}$

Then $\exists \Omega_0\ \mathsf{e_0}\ n$,

(i) $\Omega \triangleright \mathsf{e} \xrightarrow{\overline{\mathsf{a_0}}}{}^n_{ctx} \Omega_0 \triangleright \mathsf{e_0}$

(ii) $\Omega_0 \triangleright \mathsf{e_0} \xrightarrow{\overline{\mathsf{a_1}}}{}^*_{ctx} \Omega' \triangleright \mathsf{f}_{\lightning}$

Proof. Induction on Assumption (a). ☐

Lemma 22 (Static Typing implies Runtime Typing (Toplevel)). *If*

(a) $\Xi \downarrow = \Gamma_0$

(b) $\Gamma_0 \vdash \mathsf{call\ main\ 0} : \mathbb{N}$

Then

(i) $\vdash \xi; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]^{\mathsf{ctx}}\,; [\cdot]^{\mathsf{comp}}\,; [\cdot] \triangleright \mathsf{call\ main\ 0} : \mathbb{N}$

Proof. Use Rules int-$\mathbb{N}$, $t-\mathsf{var}$, $t-\mathbb{N}$, $t-\mathsf{call}$ and $t-$prog$-$runtime. ☐

Lemma 23 (Typed Linking Recomposition). *If*

(a) $\Xi = \Xi_1 \bowtie \Xi_2$

(b) $\Gamma \vdash \Xi_1$ ok

(c) $\Gamma \vdash \Xi_2$ ok

Then

(i) $\Gamma \vdash \Xi\,\text{ok}$

Proof. Easy. $\square$

Lemma 24 (Typing Decomposition). *If*

(a) $\Gamma \vdash \mathsf{K}[\mathsf{e}] : \tau$

Then $\exists \tau_{\mathsf{e}}$,

(i) $\Gamma \vdash \mathsf{K} : \tau_{\mathsf{e}} \to \tau$

(ii) $\Gamma \vdash \mathsf{e} : \tau_{\mathsf{e}}$

Proof. Induction on K. $\square$

Lemma 25 (Typing Composition). *If*

(a) $\Gamma \vdash \mathsf{K} : \tau_{\mathsf{e}} \to \tau$

(b) $\Gamma \vdash \mathsf{e} : \tau_{\mathsf{e}}$

Then

(i) $\Gamma \vdash \mathsf{K}[\mathsf{e}] : \tau$

Proof. Induction on Assumption (a). $\square$

Lemma 26 (Runtime Typing Decomposition). *If*

(a) $\vdash \Omega \triangleright \mathsf{K}[\mathsf{e}] : \tau$

Then $\exists \tau_{\mathsf{e}}$,

(i) $\vdash \Omega \triangleright \mathsf{K} : \tau_{\mathsf{e}} \to \tau$

(ii) $\vdash \Omega \triangleright \mathsf{e} : \tau_{\mathsf{e}}$

Proof. Use Lemma 24. $\square$

Lemma 27 (Runtime Typing Composition). *If*

(a) $\vdash \Omega \triangleright \mathsf{K} : \tau_{\mathsf{e}} \to \tau$

(b) $\vdash \Omega \triangleright \mathsf{e} : \tau_{\mathsf{e}}$

Then

(i) $\vdash \Omega \triangleright \mathsf{K}[\mathsf{e}] : \tau$

Proof. Use Lemma 25. $\square$

Lemma 28 (Store Agree Weaken). *If*

(a) $T_{TMS} \simeq_{\delta_{MS}} \Delta$

(b) $\delta_{MS} \subseteq \delta'_{MS}$

Then

(i) $T_{TMS} \simeq_{\delta'_{MS}} \Delta$

Proof. Induction on $T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta$. $\square$

Lemma 29 (Filter Weaken). *If*

(a) $\theta^*_{\delta_{ms}}(\overline{a}) \neq \underline{[\cdot]}$

(b) $\delta_{ms} \subseteq \delta'_{ms}$

Then

(i) $\theta^*_{\delta_{ms}}(\overline{a}) = \theta^*_{\delta'_{ms}}(\overline{a})$

Lemma 30 (TMS-Δ-split). *If*

(a) $T_{TMS} \simeq_{\delta_{MS}} \Delta_1, x \mapsto (\ell; t; \rho; n), \Delta_2$

Then $\exists T_{TMS1}\ T_{TMS2}\ \underline{\ell}$,

(i) $T_{TMS1} \simeq_{\delta_{MS}} \Delta_1$

(ii) $\{\underline{\ell}\} \cup \emptyset \simeq_{\delta_{MS}} x \mapsto (\ell; t; \rho; n), [\cdot]$

(iii) $T_{TMS2} \simeq_{\delta_{MS}} \Delta_2$

(iv) $T_{TMS} = T_{TMS1} \cup \{\underline{\ell}\} \cup T_{TMS2}$

Proof. Induction on Assumption (a). $\square$

Lemma 31 (Trace-based Transitivity Authentic). *If*

(a) $\overline{a_1^{ms}} \cong^* \overline{\boldsymbol{a}}_1$

(b) $\overline{a_2^{ms}} \cong^* \overline{\boldsymbol{a}}_2$

Then

(i) $\overline{a_1^{ms}} \cdot \overline{a_2^{ms}} \cong^* \overline{\boldsymbol{a}}_1 \cdot \overline{\boldsymbol{a}}_2$

Proof. Easy using Rule TMS-Trans-Authentic. $\square$

Lemma 32 (Monitor Step Subset). *If*

(a) $\vdash T_{TMS} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{TMS}'$

Then

(i) $T_{TMS} \subseteq_F T_{TMS}'$

Proof. Easy induction on $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$. $\square$

Lemma 33 (Monitor Steps Subset). *If*

(a) $\vdash T_{TMS} \overset{\boldsymbol{a}}{\rightsquigarrow}{}^* T_{TMS}'$

Then

(i) $T_{TMS} \subseteq_F T_{TMS}'$

Proof. Easy induction on $\vdash T_{\text{TMS}} \overset{a}{\rightsquigarrow}{}^{*} T_{\text{TMS}}{}'$ using Lemma 32 (Monitor Step Subset). ☐

Lemma 34 (α-conv. Typing)**.** *If*

(a) $\Gamma \vdash \mathsf{e} : \tau$

Then

(i) $\Gamma[\mathsf{z} : \tau' / \mathsf{x} : \tau'] \vdash \mathsf{e}[\mathsf{z}/\mathsf{x}] : \tau$

.

Proof. Induction on Assumption (a). ☐

Lemma 35 (Substitution)**.** *If*

(a) $\mathsf{x} : \tau', \Gamma_1 \vdash \mathsf{e} : \tau$

(b) $\Gamma_2 \vdash \mathsf{v} : \tau'$

(c) $\Gamma_3 = \Gamma_1 \circ \Gamma_2$

Then

(i) $\Gamma_3 \vdash \mathsf{e}[\mathsf{v}/\mathsf{x}] : \tau$

.

Proof. By induction on Assumption (a). ☐

Lemma 36 (Base Preservation)**.** *If*

(a) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(b) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\mathsf{a}} \Omega' \triangleright \mathsf{e}'\gamma'$

(c) $\Omega' \neq \lightning$

Then

(i) $\vdash \Omega' \triangleright \mathsf{e}'\gamma' : \tau$

.

Proof. Induction on Assumption (b). ☐

Lemma 37 (Ctx Preservation)**.** *If*

(a) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(b) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\mathsf{a}}_{ctx} \Omega' \triangleright \mathsf{e}'\gamma'$

Then

(i) $\vdash \Omega' \triangleright \mathsf{e}'\gamma' : \tau$

.

Proof. Induction on Assumption (b). ☐

Lemma 38 (N-Steps Preservation). *If*

(a) $\vdash \Omega \triangleright e\gamma : \tau$

(b) $\Omega \triangleright e\gamma \xrightarrow{\bar{a}}{}^{n}_{ctx} \Omega' \triangleright e'\gamma'$

Then

(i) $\vdash \Omega' \triangleright e'\gamma' : \tau$

.

Proof. Induction on Assumption (b). ☐

Lemma 39 (Steps Preservation). *If*

(a) $\vdash \Omega \triangleright e\gamma : \tau$

(b) $\Omega \triangleright e\gamma \xrightarrow{\bar{a}}{}^{*}_{ctx} \Omega' \triangleright e'\gamma'$

Then

(i) $\vdash \Omega' \triangleright e'\gamma' : \tau$

.

Proof. Induction on Assumption (b). ☐

Lemma 40 (Progress). *If*

(a) $\vdash \Omega \triangleright e\gamma : \tau$

Then $\exists \Omega'\ f_{\lightning}\ \bar{a}$,

(i) $\Omega \triangleright e \xrightarrow{\bar{a}}{}^{*}_{ctx} \Omega' \triangleright f_{\lightning}$

.

Proof. Nested induction on Assumption (a). ☐

Lemma 41 (Top-Level Progress). *If*

(a) $\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \Xi, \xi$

Then $\exists \Omega\ f_{\lightning}\ \bar{a}$,

(i) $\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \xRightarrow{\bar{a}} \Omega \triangleright f_{\lightning}$

(ii) $\Omega.\Xi = \Xi$

(iii) $\Omega.\xi = \xi$

.

Proof. Immediate consequence of Lemma 40. ☐

Lemma 42 (Base TMS via Monitor). *If*

(a) $\vdash \Omega \triangleright e\gamma : \tau$

(b) $\Omega \triangleright e\gamma \xrightarrow{a} \Omega' \triangleright e'\gamma'$

(c) $T_{TMS} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{MS}\ T_{TMS}'$,

(i) $\delta_{MS} \subseteq \delta'_{MS}$

(ii) $\theta_{\delta'_{MS}}(\mathsf{a}) \cong \boldsymbol{a}$

(iii) $\vdash T_{TMS} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{TMS}'$

(iv) $T_{TMS}' \simeq_{\delta'_{MS}} \Omega'.\Delta$

.

Proof. First, we note

- $\Omega = \Phi; \mathsf{t}; \Psi$
- $\Phi = \xi; \Xi; \overline{\mathsf{K}}$
- $\Psi = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta$

Induction on $\Omega \triangleright e\gamma \xrightarrow{a} \Omega' \triangleright e'\gamma'$.

Case e – delete**:** If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\vdash \Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta_2 \triangleright \mathsf{delete}\ \mathsf{x} : \tau$

(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta_2$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\underline{Dealloc\ }\delta'_{\text{MS}}(\ell) \cong \boldsymbol{a}$

(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$

(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; ☣; \mathsf{n}), \Delta_2$

Apply Lemma 30 (TMS-Δ-split) on Assumption (H_3):

(F_1) $T_{\text{TMS}1} \simeq_{\delta_{MS}} \Delta_1$

(F_2) $\{\underline{\ell}\} \cup \emptyset \simeq_{\delta_{MS}} \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), [\cdot]$

(F_3) $T_{\text{TMS}2} \simeq_{\delta_{MS}} \Delta_2$

(F_4) $T_{\text{TMS}} = T_{\text{TMS}1} \cup \{\underline{\ell}\} \cup T_{\text{TMS}2}$

Invert Assumption (F_2) to conclude $\delta_{\text{MS}}(\ell) = \underline{\ell}$. Choose $\boldsymbol{a} = \textbf{Dealloc}\ \underline{\ell}$ and $\delta'_{\text{MS}} = \delta_{\text{MS}}$.

Goal (i), $\delta_{\text{MS}} \subseteq \delta_{\text{MS}}$, follows trivially.

For $\underline{Dealloc\ }\delta'_{\text{MS}}(\ell) \cong \textbf{Dealloc}\ \underline{\ell}$ (Goal (ii)), apply Rule TMS-Dealloc-Authentic.

The inversion yields two cases:

Case Cons-Agree**:** If

(F_1) $\vdash \Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta_2 \triangleright \mathsf{delete}\ \mathsf{x} : \ \tau$

(F_2) $T_{\text{TMS1}} \simeq_{\delta_{MS}} \Delta_1$

(F_3) $\delta_{\text{MS}}(\ell) = \underline{\ell}$

(F_4) $T_{\text{TMS2}} \simeq_{\delta_{MS}} \Delta_2$

(F_5) $T_{\text{TMS}} = T_{\text{TMS1}} \cup \{\underline{\ell}\} \cup T_{\text{TMS2}}$

then $\exists {T_{\text{TMS}}}'$,

(i) $\vdash T_{\text{TMS}} \overset{\textbf{Dealloc}\ \underline{\ell}}{\rightsquigarrow} {T_{\text{TMS}}}'$

(ii) ${T_{\text{TMS}}}' \simeq_{\delta'_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2$

Choose ${T_{\text{TMS}}}' = T_{\text{TMS1}} \cup T_{\text{TMS2}}$.

Goal (i), $\vdash T_{\text{TMS}} \overset{\textbf{Dealloc}\ \underline{\ell}}{\rightsquigarrow} {T_{\text{TMS}}}'$, follows immediately by Rule TMS-Dealloc.

Goal (ii), ${T_{\text{TMS}}}' \simeq_{\delta'_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2$, follows immediately by Rule Poison-Agree.

Case Poison-Agree**:** If

(F_1) $\vdash \Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2 \triangleright \mathsf{delete}\ \mathsf{x} : \ \tau$

(F_2) $T_{\text{TMS1}} \simeq_{\delta_{MS}} \Delta_1$

(F_3) $\delta_{\text{MS}}(\ell) = \underline{\ell}$

(F_4) $T_{\text{TMS2}} \simeq_{\delta_{MS}} \Delta_2$

(F_5) $T_{\text{TMS}} = T_{\text{TMS1}} \cup T_{\text{TMS2}}$

then $\exists {T_{\text{TMS}}}'$,

(i) $\vdash T_{\text{TMS}} \overset{\textbf{Dealloc}\ \underline{\ell}}{\rightsquigarrow} {T_{\text{TMS}}}'$

(ii) ${T_{\text{TMS}}}' \simeq_{\delta'_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2$

.

Invert Assumption (F_1) to get

(F_7) $\Xi \vdash \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2 : \ \Gamma$

(F_8) $\Gamma \vdash \mathsf{delete}\ \mathsf{x} : \ \mathbb{N}$

Due to Assumption (F_7), we know $\mathsf{x} : \mathsf{ref}_1\ \mathbb{N} \notin \Gamma$.

But, that contradicts Assumption (F_8).

Case $e - \oplus$**:** If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\vdash \Omega \triangleright \mathsf{n}_1 + \mathsf{n}_2 : \ \tau$

(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ {T_{\text{TMS}}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\underline{\varepsilon} \cong \boldsymbol{a}$

(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} {T_{\text{TMS}}}'$

(iv) ${T_{\text{TMS}}}' \simeq_{\delta'_{\text{MS}}} \Omega$

Choose $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$.

All goals are completely trivial.

Case $e - \mathsf{let} - \mathsf{f}$**:** If

(H_1) $\gamma = [\cdot]$
(H_2) $\gamma' = [\mathsf{f}/\mathsf{x}], [\cdot]$
(H_3) $\vdash \Omega \triangleright \mathsf{let\ x = f\ in\ e'} : \tau$
(H_4) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{\varepsilon} \cong \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Omega$

Choose $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$.

All goals are completely trivial.

Case $e - \mathsf{ifz}$–true**:** If

(H_1) $\gamma = \gamma' = [\cdot]$
(H_2) $\vdash \Omega \triangleright \mathsf{ifz\ 0\ then\ e_1\ else\ e_2} : \tau$
(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{\varepsilon} \cong \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Omega$

Choose $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$.

All goals are completely trivial.

Case $e - \mathsf{ifz}$–false**:** If

(H_1) $\gamma = \gamma' = [\cdot]$
(H_2) $\vdash \Omega \triangleright \mathsf{ifz\ S(n)\ then\ e_1\ else\ e_2} : \tau$
(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{\varepsilon} \cong \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Omega$

Choose $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$.

All goals are completely trivial.

Case e – abort**:** If

(H_1) $\gamma = \gamma' = [\cdot]$
(H_2) $\vdash \Omega \triangleright \mathsf{abort}() : \tau$
(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{\lightning} \cong \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \lightning$

Choose $\boldsymbol{a} = \lightning$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = \emptyset$.

All goals are completely trivial.

Case e – new**:** If

(H_1) $\gamma = [\cdot]$
(H_2) $\gamma' = [\mathsf{z}/\mathsf{x}], [\cdot]$
(H_3) $\vdash \Omega : \tau$
(H_4) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega$
(H_5) $\Delta \vdash \ell$ *fresh*
(H_6) $\Delta \vdash \mathsf{z}$ *fresh*
(H_7) $\mathsf{H}^{\mathsf{t}'} = \mathsf{H}^{\mathsf{t}} \ll \mathsf{n}$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{Alloc\ \delta'_{\text{MS}}(\ell)\ n} \cong \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \mathsf{z} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta$

Let $\underline{\ell}$ be such that $\operatorname{dom} \delta_{\text{MS}} \vdash \underline{\ell}$ *fresh*. Choose $\boldsymbol{a} = \boldsymbol{Alloc}\ \underline{\ell}$, $\delta'_{\text{MS}} = \delta_{\text{MS}} \cup \{\ell \mapsto \underline{\ell}\}$, and $T_{\text{TMS}}' = \{\underline{\ell}\} \cup T_{\text{TMS}}$. The goals become:

(i) $\delta_{\text{MS}} \subseteq \delta_{\text{MS}} \cup \{\ell \mapsto \underline{\ell}\}$
(ii) $\underline{Alloc\ \ell\ n} \cong \boldsymbol{Alloc}\ \underline{\ell}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{Alloc}\ \underline{\ell}}{\rightsquigarrow} \{A : T_{\text{TMS}}.A \cup \{\underline{\ell}\}, F : T_{\text{TMS}}.F\}$
(iv) $\{\underline{\ell}\} \cup T_{\text{TMS}} \simeq_{\delta_{MS} \cup \{\ell \mapsto \underline{\ell}\}} \mathsf{z} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta$

Goal (i) is trivial.

Goal (ii) follows immediately by Rule TMS-Alloc-Authentic.

Goal (iii) follows by Rule TMS-Alloc.

Goal (iv) follows by Rule Cons-Agree.

Case $e - \mathsf{get} - \in$**:** The proof is analogous to the next, up to the nested case analysis.

Case $e - \mathsf{get} - \notin$**:** If

(H_1) $\gamma = \gamma' = [\cdot]$
(H_2) $\vdash \Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 \triangleright \mathsf{x}[\mathsf{n}] : \tau$
(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$
(H_4) $\ell \in \operatorname{dom} \mathsf{H}^{\mathsf{t}} \implies \mathsf{v} = \mathsf{H}^{\mathsf{t}}(\ell + \mathsf{n})$
(H_5) $\ell \notin \operatorname{dom} \mathsf{H}^{\mathsf{t}} \implies \mathsf{v} = 1729$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$
(ii) $\underline{Use\ \delta'_{\text{MS}}(\ell)\ n} \cong_{\delta'_{\text{MS}}} \boldsymbol{a}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$
(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

Apply Lemma 30 (TMS-Δ-split) on Assumption (H_3):

(F_1) $T_{\text{TMS1}} \simeq_{\delta_{MS}} \Delta_1$
(F_2) $\{\underline{\ell}\} \cup \emptyset \simeq_{\delta_{MS}} \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), [\cdot]$
(F_3) $T_{\text{TMS2}} \simeq_{\delta_{MS}} \Delta_2$
(F_4) $T_{\text{TMS}} = T_{\text{TMS1}} \cup \{\underline{\ell}\} \cup T_{\text{TMS2}}$

Invert Assumption (F_2) to conclude $\delta_{\text{MS}}(\ell) = \underline{\ell}$. Choose $\boldsymbol{a} = \boldsymbol{Use}\ \underline{\ell}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$. The goals become:

(i) $\delta_{\text{MS}} \subseteq \delta_{\text{MS}}$
(ii) $\underline{Use\ \ell\ n} \cong \boldsymbol{Use}\ \underline{\ell}$
(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{Use}\ \underline{\ell}}{\rightsquigarrow} T_{\text{TMS}}$
(iv) $T_{\text{TMS}} \simeq_{\delta_{\text{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

All goals follow easily.

Case $e - \mathsf{set} - \in$**:** The proof is analogous to the next, up to the nested case analysis.

Case $e - \mathsf{set} - \notin$**:** If

(H_1) $\gamma = \gamma' = [\cdot]$
(H_2) $\vdash \Phi; \mathsf{t}; \mathsf{H}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2 \triangleright \mathsf{x}[\mathsf{n}] : \tau$
(H_3) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$
(H_4) $\ell + \mathsf{n} \in \operatorname{dom} \mathsf{H}^{\mathsf{t}} \implies \mathsf{H}^{\mathsf{t}'} = \mathsf{H}^{\mathsf{t}}(\ell + \mathsf{n} \mapsto \mathsf{v})$
(H_5) $\ell + \mathsf{n} \notin \operatorname{dom} \mathsf{H}^{\mathsf{t}} \implies \mathsf{H}^{\mathsf{t}'} = \mathsf{H}^{\mathsf{t}}$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\underline{Use\ \ell\ n} \cong \boldsymbol{a}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{TMS}}'$

(iv) $T_{\mathrm{TMS}}' \simeq_{\delta'_{\mathrm{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

Apply Lemma 30 (TMS-Δ-split) on Assumption (H_3):

(F_1) $T_{\mathrm{TMS}1} \simeq_{\delta_{MS}} \Delta_1$

(F_2) $\{\underline{\ell}\} \cup \emptyset \simeq_{\delta_{MS}} \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), [\cdot]$

(F_3) $T_{\mathrm{TMS}2} \simeq_{\delta_{MS}} \Delta_2$

(F_4) $T_{\mathrm{TMS}} = T_{\mathrm{TMS}1} \cup \{\underline{\ell}\} \cup T_{\mathrm{TMS}2}$

Invert Assumption (F_2) to conclude $\delta_{\mathrm{MS}}(\ell) = \underline{\ell}$. Choose $\boldsymbol{a} = \boldsymbol{Use}\ \underline{\ell}$, $\delta'_{\mathrm{MS}} = \delta_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}' = T_{\mathrm{TMS}}$. The goals become:

(i) $\delta_{\mathrm{MS}} \subseteq \delta_{\mathrm{MS}}$

(ii) $\underline{Use\ \ell\ n} \cong_{\delta'_{\mathrm{MS}}} \boldsymbol{Use}\ \underline{\ell}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{Use}\ \underline{\ell}}{\rightsquigarrow} T_{\mathrm{TMS}}$

(iv) $T_{\mathrm{TMS}} \simeq_{\delta'_{\mathrm{MS}}} \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

All goals follow easily.

□

Lemma 43 (Ctx TMS via Monitor)**.** *If*

(a) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(b) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\mathsf{a}}_{ctx} \Omega' \triangleright \mathsf{e}'\gamma'$

(c) $T_{TMS} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{MS}\ T_{TMS}'$,

(i) $\delta_{MS} \subseteq \delta'_{MS}$

(ii) $\theta_{\delta'_{MS}}(\mathsf{a}) \cong \boldsymbol{a}$

(iii) $\vdash T_{TMS} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{TMS}'$

(iv) $T_{TMS}' \simeq_{\delta'_{MS}} \Omega'.\Delta$

.

Proof. Induction on Assumption (b).

Case $e-$ctx**:** Analogous to next case.

Case $e-$ctx$-$stuck**:** If

(H_1) $\mathsf{e}\gamma = \mathsf{K}[\mathsf{e}_0]$

(H_2) $\mathsf{e}'\gamma' = \mathsf{K}[\mathsf{e}'_0]$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $\Omega \triangleright \mathsf{e}_0 \xrightarrow{\mathsf{a}} \Omega' \triangleright \mathsf{e}_0'$

(H_5) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\theta_{\delta'_{\text{MS}}}(\mathsf{a}) \cong \boldsymbol{a}$

(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$

(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Omega'.\Delta$

Rewrite Assumption (H_3) using Assumption (H_1):

(H_6) $\vdash \Omega \triangleright \mathsf{K}[\mathsf{e}_0] : \tau$

Apply Lemma 26 (Runtime Typing Decomposition) on Assumption (H_6):

(H_7) $\vdash \Omega \triangleright \mathsf{K} : \tau_{\mathsf{e}} \to \tau$

(H_8) $\vdash \Omega \triangleright \mathsf{e}_0 : \tau_{\mathsf{e}}$

Apply Lemma 42 (Base TMS via Monitor) on Assumptions (H_4), (H_5) and (H_8), giving us exactly the assumptions necessary to prove the goals.

Case $e - \text{ctx} - \mathsf{call} - \text{main}$**:** If

(H_1) $\mathsf{e}\gamma = \mathsf{K}[\mathsf{call\ main\ 0}]$

(H_2) $\gamma' = [0/\mathsf{x}]$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\theta_{\delta'_{\text{MS}}}(\mathsf{Start; ctx}) \cong \boldsymbol{a}$

(iii) $\vdash T_{\text{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{TMS}}'$

(iv) $T_{\text{TMS}}' \simeq_{\delta'_{\text{MS}}} \Omega'.\Delta$

Note that $\theta_{\delta_{\text{MS}}}(\mathsf{Start; ctx}) = \underline{\varepsilon}$ for any δ_{MS} whatsoever. So, instantiate $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\text{MS}} = \delta_{\text{MS}}$, and $T_{\text{TMS}}' = T_{\text{TMS}}$. All goals follow easily.

Case $e - \text{ctx} - \mathsf{call} - \text{same}$**:** Mostly similar to next case.

Case $e - \text{ctx} - \mathsf{call} - \text{notsame}$**:** If

(H_1) $\mathsf{e}\gamma = \mathsf{K}[\mathsf{call\ foo\ v}]$

(H_2) $\gamma' = [\mathsf{v}/\mathsf{x}]$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $T_{\text{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{\text{MS}}\ T_{\text{TMS}}'$,

(i) $\delta_{\text{MS}} \subseteq \delta'_{\text{MS}}$

(ii) $\theta_{\delta'_{\mathrm{MS}}}$ (Call **c** foo v; t) $\cong \boldsymbol{a}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{TMS}}{}'$

(iv) $T_{\mathrm{TMS}}{}' \simeq_{\delta'_{\mathrm{MS}}} \Omega'.\Delta$

Note that $\theta_{\delta_{\mathrm{MS}}}$ (Call **c** foo v; t) $= \underline{\varepsilon}$ for any δ_{MS} and t whatsoever. So, instantiate $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\mathrm{MS}} = \delta_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}{}' = T_{\mathrm{TMS}}$. All goals follow easily.

Case $e -$ ctx $-$ return$-$main**:** If

(H_1) $\mathsf{e}\gamma = \mathsf{K}[\mathsf{return\ v}]$

(H_2) $\mathsf{e}'\gamma' = \mathsf{v}$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $T_{\mathrm{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{\mathrm{MS}}\ T_{\mathrm{TMS}}{}'$,

(i) $\delta_{\mathrm{MS}} \subseteq \delta'_{\mathrm{MS}}$

(ii) $\theta_{\delta'_{\mathrm{MS}}}$ (End v; ctx) $\cong \boldsymbol{a}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{TMS}}{}'$

(iv) $T_{\mathrm{TMS}}{}' \simeq_{\delta'_{\mathrm{MS}}} \Omega'.\Delta$

Note that $\theta_{\delta_{\mathrm{MS}}}$ (End v; ctx) $= \underline{\varepsilon}$ for any δ_{MS} whatsoever. So, instantiate $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\mathrm{MS}} = \delta_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}{}' = T_{\mathrm{TMS}}$. All goals follow easily.

Case $e -$ ctx $-$ return $-$ same**:** Mostly similar to next case.

Case $e -$ ctx $-$ return $-$ notsame**:** If

(H_1) $\mathsf{e}\gamma = \mathsf{K}'[\mathsf{return\ v}]$

(H_2) $\mathsf{e}\gamma' = \mathsf{K}[\mathsf{v}]$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $T_{\mathrm{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \boldsymbol{a}\ \delta'_{\mathrm{MS}}\ T_{\mathrm{TMS}}{}'$,

(i) $\delta_{\mathrm{MS}} \subseteq \delta'_{\mathrm{MS}}$

(ii) $\theta_{\delta'_{\mathrm{MS}}}$ (Ret **c** v; t) $\cong \boldsymbol{a}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{TMS}}{}'$

(iv) $T_{\mathrm{TMS}}{}' \simeq_{\delta'_{\mathrm{MS}}} \Omega'.\Delta$

Note that $\theta_{\delta_{\mathrm{MS}}}$ (Ret **c** v; t) $= \underline{\varepsilon}$ for any δ_{MS} and t whatsoever. So, instantiate $\boldsymbol{a} = \boldsymbol{\varepsilon}$, $\delta'_{\mathrm{MS}} = \delta_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}{}' = T_{\mathrm{TMS}}$. All goals follow easily.

□

Lemma 44 (Steps TMS via Monitor)**.** *If*

(a) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(b) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\bar{a}}{}^{*}_{ctx} \Omega' \triangleright \mathsf{e}'\gamma'$

(c) $T_{TMS} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \overline{\boldsymbol{a}}\ \delta'_{MS}\ T_{TMS}'$,

(i) $\delta_{MS} \subseteq \delta'_{MS}$

(ii) $\theta_{\delta'_{MS}}(\bar{\mathsf{a}}) \cong^* \overline{\boldsymbol{a}}$

(iii) $\vdash T_{TMS} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{TMS}'$

(iv) $T_{TMS}' \simeq_{\delta'_{MS}} \Omega'.\Delta$

.

Proof. Induction on Assumption (b).

$\Omega' = \Omega$, $\mathsf{e}' = \mathsf{e}$, and $\bar{\mathsf{a}} = [\cdot]$: This case is easy, since nothing did change. Instantiate $\overline{\boldsymbol{a}} = [\cdot]$, $\delta'_{\mathrm{MS}} = \delta_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}' = T_{\mathrm{TMS}}$. Then, Goals (i) to (iii) follow immediately by reflexivity and Goal (iv) by Assumption (c).

Induction step: The inductive hypothesis is as follows. If

(A_1) $\vdash \Omega_{\mathsf{H}} \triangleright \mathsf{e}_{\mathsf{H}}\gamma_{\mathsf{H}} : \tau$

(A_2) $T_{\mathrm{TMS}\,IH} \simeq_{\delta^0_{\mathrm{MS}}} \Omega_{\mathsf{H}}.\Delta$

then $\exists \overline{\boldsymbol{a}}_{IH}\ \delta^{IH}_{\mathrm{MS}}\ T_{\mathrm{TMS}}'_{IH}$,

(IH_1) $\delta^0_{\mathrm{MS}} \subseteq \delta^{IH}_{\mathrm{MS}}$

(IH_2) $\theta_{\delta^{IH}_{\mathrm{MS}}}(\bar{\mathsf{a}}) \cong \overline{\boldsymbol{a}}_{IH}$

(IH_3) $\vdash T_{\mathrm{TMS}\,IH} \overset{\overline{\boldsymbol{a}}_{IH}}{\rightsquigarrow}{}^* T_{\mathrm{TMS}}'_{IH}$

(IH_4) $T_{\mathrm{TMS}}'_{IH} \simeq_{\delta^{IH}_{\mathrm{MS}}} \Omega'.\Delta$

Note that above, anything not bound by $\exists$ is universally quantified, besides τ and Ω'. While our remaining context (excluding the inductive hypothesis) is: If

(H_1) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\mathsf{a}_0}_{\mathrm{ctx}} \Omega_0 \triangleright \mathsf{e}_0\gamma_0$

(H_2) $\Omega_0 \triangleright \mathsf{e}_0\gamma_0 \xrightarrow{\bar{\mathsf{a}}}{}^*_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}'\gamma'$

(H_3) $\vdash \Omega \triangleright \mathsf{e}\gamma : \tau$

(H_4) $T_{\mathrm{TMS}} \simeq_{\delta_{MS}} \Omega.\Delta$

Then $\exists \overline{\boldsymbol{a}}\ \delta'_{\mathrm{MS}}\ T_{\mathrm{TMS}}'$,

(i) $\delta_{\mathrm{MS}} \subseteq \delta'_{\mathrm{MS}}$

(ii) $\theta_{\delta'_{\mathrm{MS}}}(\mathsf{a}_0 \cdot \bar{\mathsf{a}}) \cong \overline{\boldsymbol{a}}$

(iii) $\vdash T_{\mathrm{TMS}} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^* T_{\mathrm{TMS}}'$

(iv) $T_{\mathrm{TMS}}' \simeq_{\delta'_{\mathrm{MS}}} \Omega'.\Delta$

Use Lemma 43 (Ctx TMS via Monitor) with Assumptions (H_1), (H_3) and (H_4), obtaining:

(F_1) $\delta_{\mathrm{MS}} \subseteq \delta^0_{\mathrm{MS}}$

(F_2) $\theta_{\delta^0_{\mathrm{MS}}}(\mathsf{a}_0) \cong \boldsymbol{a}_0$

(F_3) $\vdash T_{\mathrm{TMS}} \overset{\boldsymbol{a}_0}{\rightsquigarrow} T_{\mathrm{TMS}0}$

(F_4) $T_{\mathrm{TMS}0} \simeq_{\delta^0_{\mathrm{MS}}} \Omega_0.\Delta$

Discharge the inductive hypothesis with Assumption (F_4) and the result of applying Lemma 37 (Ctx Preservation) using Assumptions (H_1) and (H_3):

(G_1) $\delta^0_{\mathrm{MS}} \subseteq \delta^{IH}_{\mathrm{MS}}$

(G_2) $\theta_{\delta^{IH}_{\mathrm{MS}}}(\overline{\mathsf{a}}) \cong \overline{\boldsymbol{a}}_{IH}$

(G_3) $\vdash T_{\mathrm{TMS}0} \overset{\overline{\boldsymbol{a}}_{IH}}{\rightsquigarrow}{}^* T_{\mathrm{TMS}}{}^{IH}$

(G_4) $T_{\mathrm{TMS}}{}^{IH} \simeq_{\delta^{IH}} \Omega'.\Delta$

We solve our goals by instantiating $\overline{\boldsymbol{a}} = \boldsymbol{a} \cdot \overline{\boldsymbol{a}}_{IH}$, $\delta'_{\mathrm{MS}} = \delta^{IH}_{\mathrm{MS}}$, and $T_{\mathrm{TMS}}{}' = T_{\mathrm{TMS}}{}^{IH}$. Goal (i) follows by transitivity using Assumptions (F_1) and (G_1).

Similarily for Goal (ii) using Assumptions (F_2) and (G_2) and Lemma 29 (Filter Weaken).

Goal (iii) also by transitivity using Assumptions (F_3) and (G_3).

Goal (iv), again, by transitivity using Assumptions (F_4) and (G_4), making use of Lemma 28 (Store Agree Weaken).

$\square$

Theorem 2 ($\mathsf{L}_{\mathrm{tms}}$ is TMS via Monitor)**.** *If*

(a) $\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \overset{\overline{\mathsf{a}}}{\Longrightarrow} \Omega \triangleright \mathsf{f}_{\lightning}$

Then

(i) $\theta^*_{\delta_{ms}}(\overline{\mathsf{a}}) \in \mathrm{tmsafe}$

.

Proof. Apply Lemma 15 ($\mathtt{TMS}(\overline{a^{\mathrm{ms}}})$ implies $\overline{a^{\mathrm{ms}}} \in \mathrm{tmsafe}$) on the goal, what is left to show is:

(i) $\mathtt{TMS}(\theta_{\delta_{ms}}(\overline{\mathsf{a}}))$

Inverting Assumption (a) and omitting the spurious cases, we get

(H_1) $\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} : \xi, \Xi$

(H_2) $\xi; \Xi; \mathsf{comp}; [\cdot]\,;[\cdot]\,;[\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\overline{\mathsf{a}}}{}^*_{\mathrm{ctx}} \Omega \triangleright \mathsf{f}_{\lightning}$

Invert Assumption (H_1):

(H_3) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

(H_4) $\Xi \downarrow = \Gamma_0$

(H_5) $\mathsf{main} \in \Gamma_0$

(H_6) $\Gamma_0 \vdash \Xi\,\text{ok}$

(H_7) $\Gamma_0 \vdash \mathsf{call\ main\ 0} : \mathbb{N}$

Apply Lemma 22 (Static Typing implies Runtime Typing (Toplevel)) on Assumptions (H_4) and (H_7):

(H_8) $\vdash \xi; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]^{\mathsf{ctx}}\,; [\cdot]^{\mathsf{comp}}\,; [\cdot] \triangleright \mathsf{call\ main\ 0} : \mathbb{N}$

Note that by definition, we have:

(H_9) $\emptyset \simeq_{\delta_{ms}} [\cdot]$

With Assumptions (H_2), (H_8) and (H_9) apply Lemma 44 (Steps TMS via Monitor):

(H_{10}) $\delta_{\text{ms}} \subseteq \delta'_{\text{ms}}$

(H_{11}) $\theta_{\delta'_{\text{MS}}}(\overline{\mathsf{a}}) \cong^{*} \overline{\boldsymbol{a}}$

(H_{12}) $\vdash \emptyset \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{TMS}}$

(H_{13}) $T_{\text{TMS}} \simeq_{\delta'_{\text{MS}}} \Omega.\Delta$

Immediately, we instantiate our goal with $\delta'_{\text{ms}}, \overline{\boldsymbol{a}}$, and T_{TMS}. Make use of Lemma 29 (Filter Weaken) on the first part, and Lemma 28 (Store Agree Weaken) on the second part, noting that Assumption (H_{10}). Now, Assumptions (H_{11}) and (H_{12}) are precisely what is required in Definition 31 (Trace is temporal memory safe via monitor). $\square$

4.3 Target Language

4.3.1 Syntax

$$
\begin{array}{rl}
\textit{Final Result } \mathbf{f} ::= \mathbf{v} \mid \mathbf{x} & \textit{May be a Result } \mathbf{f}_{\not\downarrow} ::= \mathbf{f} \mid \mathbf{stuck} \\
\textit{Expressions } \mathbf{e} ::= & \mathbf{f}_{\not\downarrow} \mid \mathbf{e_1} \oplus \mathbf{e_2} \mid \mathbf{x}[\mathbf{e}] \mid \mathbf{let\ x = e_1\ in\ e_2} \mid \mathbf{x}[\mathbf{e_1}] \leftarrow \mathbf{e_2} \\
& \mid \mathbf{let\ x = new\ e_1\ in\ e_2} \mid \mathbf{delete\ x} \mid \mathbf{return\ e} \mid \mathbf{call\ foo\ e} \\
& \mid \mathbf{ifz\ e_1\ then\ e_2\ else\ e_3} \mid \mathbf{abort()} \mid \mathbf{x\ is\ ☣} \\
& \mid \langle \mathbf{e_1}; \mathbf{e_2} \rangle \mid \pi_1\ \mathbf{e} \mid \pi_2\ \mathbf{e} \mid \mathbf{e\ has\ } \tau \quad \text{where } \oplus \in \{+, -, \times, <\} \\
\textit{Functions } \mathbf{F} ::= \mathbf{let\ foo\ x := e} & \textit{Types } \tau ::= \mathbb{N} \mid \mathbb{N} \times \mathbb{N} \\
\textit{Values } \mathbf{v} ::= \langle \mathbf{n_1}, \mathbf{n_2} \rangle \mid \mathbf{n} \in \mathbb{N} & \textit{References } \ell \in \mathbb{N} \\
\textit{Eval.Ctx. } \mathbf{K} ::= & [\cdot] \mid \mathbf{K} \oplus \mathbf{e} \mid \mathbf{v} \oplus \mathbf{E} \mid \mathbf{x}[\mathbf{K}] \mid \mathbf{let\ x = K\ in\ e} \\
& \mid \mathbf{x}[\mathbf{K}] \leftarrow \mathbf{e} \mid \mathbf{x}[\mathbf{v}] \leftarrow \mathbf{K} \mid \mathbf{let\ x = new\ K\ in\ e} \\
& \mid \langle \mathbf{K}; \mathbf{e} \rangle \mid \langle \mathbf{n}; \mathbf{K} \rangle \mid \mathbf{ifz\ K\ then\ e_1\ else\ e_2} \mid \pi_1\ \mathbf{K} \mid \pi_2\ \mathbf{K} \\
& \mid \mathbf{K\ has\ } \tau \mid \mathbf{call\ foo\ K} \mid \mathbf{return\ K} \\
\textit{Variables } \mathbf{x} \mid \mathbf{y} \mid \mathbf{foo} \mid \ldots & \textit{Poison } \rho ::= \square \mid ☣ \\
\textit{Sandbox Tag } \mathbf{t} ::= \mathbf{ctx} \mid \mathbf{comp} & \\
\textit{Store } \boldsymbol{\Delta} ::= [\cdot] \mid \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}), \boldsymbol{\Delta} & \textit{Communication } \mathbf{c} ::= \ ? \mid\ ! \mid \varnothing \\
\textit{Cont. Stack } \overline{\mathbf{K}} ::= [\cdot] \mid (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}} & \textit{Library } \boldsymbol{\Xi} ::= [\cdot] \mid \mathbf{F}, \boldsymbol{\Xi} \\
\textit{Relevant } \xi ::= [\cdot] \mid \mathbf{foo}, \xi & \textit{Heaps } \mathbf{H} ::= [\cdot] \mid \mathbf{H} :: \mathbf{n} \quad \textit{State } \boldsymbol{\Omega} ::= \boldsymbol{\Phi}; \mathbf{t}; \boldsymbol{\Psi} \\
\textit{Flow State } \boldsymbol{\Phi} ::= \xi; \boldsymbol{\Xi}; \overline{\mathbf{K}} & \textit{Memory State } \boldsymbol{\Psi} ::= \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \boldsymbol{\Delta} \\
\textit{Programs } \mathbf{prog}\ \boldsymbol{\Xi}_{\mathbf{ctx}}\ \boldsymbol{\Xi}_{\mathbf{comp}} & \textit{Substitutions } \gamma ::= [\mathbf{v}/\mathbf{x}], \gamma \mid [\cdot]
\end{array}
$$

Figure 33: Syntax of $\mathrm{L_{ms}}$

The target language is very similar to the source language presented in the previous section. However, it does contain simple, non-nested pairs and dynamic type checks.

4.3.2 Dynamic Semantics

$$\boxed{\mathrm{dom}\ \Xi = \mathbf{foo}, \ldots, \mathbf{bar}}\ \text{„Collect function names."}$$

$$\frac{}{\mathrm{dom}\ [\cdot] = [\cdot]}\ (\Xi\text{-dom-}[\cdot]) \qquad \frac{\mathrm{dom}\ \Xi = \mathbf{D}}{\mathrm{dom}\ (\mathbf{let\ foo\ x} : \tau_\lambda := \mathbf{e}), \Xi = \mathbf{foo}, \mathbf{D}}\ (\Xi\text{-dom-cons})$$

$$\boxed{\Xi \equiv \Xi_1 \cup \Xi_2}\ \text{„Merging } \mathrm{L_{ms}} \text{ libraries."}$$

$$\frac{\Xi = \Xi_1}{\Xi \equiv \Xi_1 \cup [\cdot]}\ (\text{lib-merge-empty}) \qquad \frac{\Xi \equiv \Xi_1 \cup \Xi_2}{\mathbf{F}, \Xi \equiv \Xi_1 \cup \mathbf{F}, \Xi_2}\ (\text{lib-merge-cons})$$

$$\boxed{\Xi_1 \bowtie \Xi_2 = \Xi}\ \text{„Syntactically linking } \mathrm{L_{ms}} \text{ libraries."}$$

$$\frac{\mathrm{dom}\,\Xi_1 \cap \mathrm{dom}\,\Xi_2 = \emptyset \qquad \Xi \equiv \Xi_1 \cup \Xi_2}{\Xi_1 \bowtie \Xi_2 = \Xi}\ (\text{syntactic-plugging})$$

Figure 34: $\mathrm{L_{ms}}$ plugging of libraries and collecting of function names.

$$\begin{aligned} \textit{Base Events}\ \mathbf{a_b} ::= {} & \mathbf{Alloc}\ \ell\ \mathbf{v} \mid \mathbf{Dealloc}\ \ell \mid \mathbf{Get}\ \ell\ \mathbf{v} \mid \mathbf{Set}\ \ell\ \mathbf{v}\ \mathbf{v'} \mid \lightning \\ & \mid \mathbf{Call\ c\ foo\ v} \mid \mathbf{Ret\ c\ v} \mid \mathbf{Start} \mid \mathbf{End\ v} \\ \textit{Events}\ \mathbf{a} ::= {} & \varepsilon \mid \mathbf{a_b}; \mathbf{t} \end{aligned}$$

Figure 35: Events of $\mathrm{L_{ms}}$.

$$\boxed{\Omega \triangleright e \xrightarrow{a} \Omega' \triangleright e'}$$ „Expression e evaluates under configuration Ω to e' and new configuration Ω', emitting event a."

$$\frac{n_1 \oplus n_2 = n_3}{\Omega \triangleright n_1 \oplus n_2 \xrightarrow{\varepsilon} \Omega \triangleright n_3}\ (e-\oplus)$$

$$\frac{\Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; t; \rho; m), \Delta_2 \qquad \ell + n \in \text{dom } H^t}{\Phi; t'; \Psi \triangleright x[n] \xrightarrow{\text{Get } \ell \text{ n;t}} \Phi; t'; \Psi \triangleright H^t(\ell + n)}\ (e-\text{get}-\in)$$

$$\frac{\Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; t; \rho; m), \Delta_2 \qquad \ell \notin \text{dom } H^t}{\Phi; t'; \Psi \triangleright x[n] \xrightarrow{\text{Get } \ell \text{ n;t}} \Phi; t'; \Psi \triangleright 1729}\ (e-\text{get}-\notin)$$

$$\frac{\begin{array}{c}\Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; ctx; \rho; n), \Delta_2 \qquad H^{ctx'} = H^{ctx}(\ell + n \mapsto v) \\ \Psi' = H^{ctx'}; H^{comp}; \Delta_1, x \mapsto (\ell; ctx; \rho; n), \Delta_2\end{array}}{\Phi; ctx; \Psi \triangleright x[n] \leftarrow v \xrightarrow{\text{Set } \ell \text{ n v;ctx}} \Phi; ctx; \Psi' \triangleright v}\ (e-\text{set}-ctx)$$

$$\frac{\begin{array}{c}\Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; comp; \rho; n), \Delta_2 \qquad H^{comp'} = H^{comp}(\ell + n \mapsto v) \\ \Psi' = H^{ctx}; H^{comp'}; \Delta_1, x \mapsto (\ell; comp; \rho; n), \Delta_2\end{array}}{\Phi; comp; \Psi \triangleright x[n] \leftarrow v \xrightarrow{\text{Set } \ell \text{ n v;comp}} \Phi; comp; \Psi' \triangleright v}\ (e-\text{set}-comp)$$

$$\frac{}{\Omega \triangleright \text{let } x = f \text{ in } e \xrightarrow{\varepsilon} \Omega \triangleright e[f/x]}\ (e-\text{let}-f)$$

$$\frac{\begin{array}{c}\Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; t; \square; n), \Delta_2 \\ \Psi = H^{ctx}; H^{comp}; \Delta_1, x \mapsto (\ell; t; ☣; n), \Delta_2\end{array}}{\Phi; t; \Psi \triangleright \text{delete } x \xrightarrow{\text{Dealloc } \ell\text{;t}} \Phi; t; \Psi' \triangleright 0}\ (e-\text{delete})$$

$$\frac{\begin{array}{c}\Delta \vdash \ell \textit{ fresh} \qquad \Delta \vdash z \textit{ fresh} \qquad H^{ctx'} = H^{ctx} \ll n \\ \Psi = H^{ctx'}; H^{comp}; z \mapsto (\ell; ctx; \square; n), \Delta\end{array}}{\Phi; ctx; H^{ctx}; H^{comp}; \Delta \triangleright \text{let } x = \text{new } n \text{ in } e \xrightarrow{\text{Alloc } \ell \text{ n;ctx}} \Phi; ctx; \Psi \triangleright e[z/x]}\ (e-\text{new}-ctx)$$

$$\frac{\begin{array}{c}\Delta \vdash \ell \textit{ fresh} \qquad \Delta \vdash z \textit{ fresh} \qquad H^{comp'} = H^{comp} \ll n \\ \Psi = H^{ctx}; H^{comp'}; z \mapsto (\ell; ctx; \square; n), \Delta\end{array}}{\Phi; comp; H^{ctx}; H^{comp}; \Delta \triangleright \text{let } x = \text{new } n \text{ in } e \xrightarrow{\text{Alloc } \ell \text{ n;comp}} \Phi; comp; \Psi \triangleright e[z/x]}\ (e-\text{new}-comp)$$

$$\frac{}{\Omega \triangleright \text{ifz } 0 \text{ then } e_1 \text{ else } e_2 \xrightarrow{\varepsilon} \Omega \triangleright e_1}\ (e-\text{ifz}-\text{true})$$

$$\frac{n \neq 0}{\Omega \triangleright \text{ifz } n \text{ then } e_1 \text{ else } e_2 \xrightarrow{\varepsilon} \Omega \triangleright e_2}\ (e-\text{ifz}-\text{false}) \qquad \frac{}{\Omega \triangleright \text{abort}() \xrightarrow{\lightning} \Omega \triangleright \text{stuck}}\ (e-\text{abort})$$

Figure 36: Evaluation of L_{ms} expressions.

$$\frac{}{\boldsymbol{\Omega} \triangleright \pi_1 \langle \mathbf{n_1}; \mathbf{n_2} \rangle \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{n_1}} \;(e-\pi_1) \qquad \frac{}{\boldsymbol{\Omega} \triangleright \pi_2 \langle \mathbf{n_1}; \mathbf{n_2} \rangle \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{n_2}} \;(e-\pi_2)$$

$$\frac{\boldsymbol{\Psi} = \mathbf{H}; \boldsymbol{\Delta} \qquad \boldsymbol{\Delta} = \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; ☣; \mathbf{n}), \boldsymbol{\Delta_2}}{\boldsymbol{\Phi}; \mathbf{t}'; \boldsymbol{\Psi} \triangleright \mathbf{x} \textbf{ is } ☣ \xrightarrow{\varepsilon} \boldsymbol{\Phi}; \mathbf{t}'; \boldsymbol{\Psi} \triangleright \mathbf{0}} \;(e-\mathbf{x}\textbf{ is }☣-\mathsf{yes})$$

$$\frac{\boldsymbol{\Psi} = \mathbf{H}; \boldsymbol{\Delta} \qquad \boldsymbol{\Delta} = \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \square; \mathbf{n}), \boldsymbol{\Delta_2}}{\boldsymbol{\Phi}; \mathbf{t}'; \boldsymbol{\Psi} \triangleright \mathbf{x} \textbf{ is } ☣ \xrightarrow{\varepsilon} \boldsymbol{\Phi}; \mathbf{t}'; \boldsymbol{\Psi} \triangleright \mathbf{1}} \;(e-\mathbf{x}\textbf{ is }☣-\mathsf{no})$$

$$\frac{}{\boldsymbol{\Omega} \triangleright \mathbf{n} \textbf{ has } \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{0}} \;(e-\mathbf{n}\textbf{ has }\mathbb{N}) \qquad \frac{}{\boldsymbol{\Omega} \triangleright \langle \mathbf{n_1}, \mathbf{n_2} \rangle \textbf{ has } \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{1}} \;(e-pair-\textbf{has }\mathbb{N})$$

$$\frac{}{\boldsymbol{\Omega} \triangleright \mathbf{x} \textbf{ has } \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{1}} \;(e-\mathbf{x}\textbf{ has }\mathbb{N}) \qquad \frac{}{\boldsymbol{\Omega} \triangleright \mathbf{n} \textbf{ has } \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{1}} \;(e-\mathbf{n}\textbf{ has }\mathbb{N}\times\mathbb{N})$$

$$\frac{}{\boldsymbol{\Omega} \triangleright \langle \mathbf{n_1}, \mathbf{n_2} \rangle \textbf{ has } \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{0}} \;(e-pair-\textbf{has }\mathbb{N}\times\mathbb{N}) \qquad \frac{}{\boldsymbol{\Omega} \triangleright \mathbf{x} \textbf{ has } \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{1}} \;(e-\mathbf{x}\textbf{ has }\mathbb{N}\times\mathbb{N})$$

Figure 37: Evaluation of $\mathrm{L_{ms}}$ expressions, continued.

γ acts as postfix and binds anything before it up to $\triangleright$, so $\textbf{let } \mathbf{x} = \mathbf{e_1} \textbf{ in } \mathbf{e_2}\gamma$ is $(\textbf{let } \mathbf{x} = \mathbf{e_1} \textbf{ in } \mathbf{e_2})\gamma$.

$\boxed{\Omega \triangleright \mathbf{K}[\mathbf{e}] \xrightarrow{\mathbf{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{K}[\mathbf{e}']}$ „Given an evaluation context $\mathbf{K}$ and an expression $\mathbf{e}$, it evaluates under configuration Ω to $\mathbf{e}'$ and new configuration Ω' in context $\mathbf{K}$, emitting event $\mathbf{a}$."

$$\text{(}e-\mathsf{ctx}\text{)}\ \frac{\Omega \triangleright \mathbf{e} \xrightarrow{\mathbf{a}} \Omega' \triangleright \mathbf{e}'}{\Omega \triangleright \mathbf{K}[\mathbf{e}] \xrightarrow{\mathbf{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{K}[\mathbf{e}']} \qquad \text{(}e-\mathsf{ctx}-\mathbf{stuck}\text{)}\ \frac{\Omega \triangleright \mathbf{e} \xrightarrow{\lightning} \Omega' \triangleright \mathbf{stuck}}{\Omega \triangleright \mathbf{K}[\mathbf{e}] \xrightarrow{\lightning}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{stuck}}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{call}-\mathsf{main}\text{)}\ \frac{\begin{array}{c}\Omega = \xi; \Xi; [\cdot]\,; \mathbf{comp}; \Psi \qquad \Xi = \Xi_1, (\mathbf{let\ main\ x} : \tau_\lambda := \mathbf{e}), \Xi_2 \\ \Omega' = \xi; \Xi; \mathbf{ctx}; \mathbf{K}^{\mathbf{main}}, [\cdot]\,; \Psi\end{array}}{\Omega \triangleright \mathbf{K}[\mathbf{call\ main\ v}] \xrightarrow{\mathbf{Start;comp}}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{e}[\mathbf{x}/\mathbf{v}]}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{call}-\mathsf{notsame}\text{)}\ \frac{\begin{array}{c}\Omega = \xi; \Xi; \overline{\mathbf{K}}; \mathbf{t}; \Psi \qquad \Xi = \Xi_1, (\mathbf{let\ foo\ x} : \tau_\lambda := \mathbf{e}), \Xi_2 \\ \mathbf{foo} \in_{\neg \mathbf{t}} \xi \qquad \rho\,(\neg \mathbf{t}) = \mathbf{c} \qquad \Omega' = \xi; \Xi; \mathbf{K}^{\mathbf{foo}}, \overline{\mathbf{K}}; \neg \mathbf{t}; \Psi\end{array}}{\Omega \triangleright \mathbf{K}[\mathbf{call\ foo\ v}] \xrightarrow{\mathbf{Call\ c\ foo\ v;t}}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{e}[\mathbf{x}/\mathbf{v}]}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{call}-\mathsf{same}\text{)}\ \frac{\begin{array}{c}\Omega = \xi; \Xi; \overline{\mathbf{K}}; \mathbf{t}; \Psi \qquad \Xi = \Xi_1, (\mathbf{let\ foo\ x} : \tau_\lambda := \mathbf{e}), \Xi_2 \\ \mathbf{foo} \in_{\mathbf{t}} \xi \qquad \Omega' = \xi; \Xi; \mathbf{K}^{\mathbf{foo}}, \overline{\mathbf{K}}; \mathbf{t}; \Psi\end{array}}{\Omega \triangleright \mathbf{K}[\mathbf{call\ foo\ v}] \xrightarrow{\mathbf{Call}\ \varnothing\ \mathbf{foo\ v;t}}_{\mathsf{ctx}} \Omega' \triangleright \mathbf{e}[\mathbf{x}/\mathbf{v}]}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{return}-\mathsf{main}\text{)}\ \frac{}{\xi; \Xi; [\cdot]^{\mathbf{main}}, [\cdot]\,; \mathbf{ctx}; \Psi \triangleright \mathbf{K}'[\mathbf{return\ v}] \xrightarrow{\mathbf{End\ v;t}}_{\mathsf{ctx}} \xi; \Xi; [\cdot]\,; \mathbf{comp}; \Psi \triangleright \mathbf{v}}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{return}-\mathsf{notsame}\text{)}\ \frac{\mathbf{foo} \in_{\neg \mathbf{t}} \xi \qquad \rho\,(\neg \mathbf{t}) = \mathbf{c}; \mathbf{t}}{\xi; \Xi; \mathbf{K}^{\mathbf{foo}}, \overline{\mathbf{K}}; \mathbf{t}; \Psi \triangleright \mathbf{K}'[\mathbf{return\ v}] \xrightarrow{\mathbf{Ret\ c\ v;t}}_{\mathsf{ctx}} \xi; \Xi; \overline{\mathbf{K}}; \neg \mathbf{t}; \Psi \triangleright \mathbf{K}[\mathbf{v}]}$$

$$\text{(}e-\mathsf{ctx}-\mathbf{return}-\mathsf{same}\text{)}\ \frac{\mathbf{foo} \in_{\mathbf{t}} \xi}{\xi; \Xi; \mathbf{K}^{\mathbf{foo}}, \overline{\mathbf{K}}; \mathbf{t}; \Psi \triangleright \mathbf{K}'[\mathbf{return\ v}] \xrightarrow{\mathbf{Ret}\ \varnothing\ \mathbf{v;t}}_{\mathsf{ctx}} \xi; \Xi; \overline{\mathbf{K}}; \mathbf{t}; \Psi \triangleright \mathbf{K}[\mathbf{v}]}$$

$\boxed{\rho(\mathbf{t}) = \mathbf{c}}$ „Returns either ? or ! depending on $\mathbf{t}$."

$$\text{(comm-ctxtocomp)}\ \frac{}{\rho(\mathbf{ctx}) = ?} \qquad \text{(comm-comptoctx)}\ \frac{}{\rho(\mathbf{comp}) = !}$$

$\boxed{\neg \mathbf{t} = \mathbf{t}'}$ „Negation of $\mathbf{t}$."

$$\text{(neg-ctx)}\ \frac{}{\neg \mathbf{ctx} = \mathbf{comp}} \qquad \text{(neg-comp)}\ \frac{}{\neg \mathbf{comp} = \mathbf{ctx}}$$

Figure 38: Contextual Evaluation of L_{ms} expressions.

$\boxed{\Omega \triangleright e \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \Omega' \triangleright e'}$ „Expression e evaluates under configuration Ω to e' and new configuration Ω', emitting list of events $\overline{a}$."

$$\frac{}{\Omega \triangleright f_{\lightning} \xrightarrow{[\cdot]}{}^{*}_{\mathrm{ctx}} \Omega \triangleright f_{\lightning}}\;(et-\mathsf{refl})$$

$$\frac{\Omega \triangleright e \xrightarrow{a}_{\mathrm{ctx}} \Omega' \triangleright e' \qquad \Omega' \triangleright e' \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright e'' \qquad a \neq \varepsilon}{\Omega \triangleright e \xrightarrow{a \cdot \overline{a}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright e''}\;(et-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright e \xrightarrow{\varepsilon}_{\mathrm{ctx}} \Omega' \triangleright e' \qquad \Omega' \triangleright e' \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright e''}{\Omega \triangleright e \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright e''}\;(et-\mathsf{trans{-}unimportant})$$

$\boxed{\Omega \triangleright e \xrightarrow{\overline{a}}{}^{n}_{\mathrm{ctx}} \Omega' \triangleright e'}$ „Expression e evaluates in n steps under configuration Ω to e' and new configuration Ω', emitting list of events $\overline{a}$."

$$\frac{}{\Omega \triangleright e \xrightarrow{\varepsilon}_{\mathrm{ctx}}{}^{0} \Omega \triangleright e}\;(en-\mathsf{refl})$$

$$\frac{\Omega \triangleright e \xrightarrow{a}_{\mathrm{ctx}} \Omega' \triangleright e' \qquad \Omega' \triangleright e' \xrightarrow{\overline{a}}_{\mathrm{ctx}}{}^{n} \Omega'' \triangleright e'' \qquad a \neq \varepsilon}{\Omega \triangleright e \xrightarrow{a \cdot \overline{a}}_{\mathrm{ctx}}{}^{(n+1)} \Omega'' \triangleright e''}\;(en-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright e \xrightarrow{\varepsilon}_{\mathrm{ctx}} \Omega' \triangleright e' \qquad \Omega' \triangleright e' \xrightarrow{\overline{a}}_{\mathrm{ctx}}{}^{n} \Omega'' \triangleright e''}{\Omega \triangleright e \xrightarrow{\overline{a}}_{\mathrm{ctx}}{}^{(n+1)} \Omega'' \triangleright e''}\;(en-\mathsf{trans{-}unimportant})$$

Figure 39: Trace prefix generation given a L_{ms} program using the reflexive-transitive closure.

$\boxed{\mathbf{prog}\ \Xi_{ctx}\ \Xi_{comp} \overset{\overline{a}}{\Longrightarrow} \Omega \triangleright f}$ „Run L_{ms} program $\mathbf{prog}\ \Xi_{ctx}\ \Xi_{comp}$, giving dynamic state Ω and emitting trace $\overline{a}$."

$$\frac{\begin{array}{c}\Xi = \Xi_{ctx} \bowtie \Xi_{comp} \qquad \mathbf{main} \notin \xi \qquad \xi = \mathrm{dom}\ \Xi_{comp} \\ \xi; \Xi; [\cdot]\,; t; [\cdot]\,; [\cdot] \triangleright \mathbf{call\ main\ 0} \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \Omega \triangleright f_{\lightning}\end{array}}{\mathbf{prog}\ \Xi_{ctx}\ \Xi_{comp} \overset{\overline{a}}{\Longrightarrow} \Omega \triangleright f_{\lightning}}\;(e-\mathsf{wprog})$$

$$\frac{\begin{array}{c}\Xi = \Xi_{ctx} \bowtie \Xi_{comp} \qquad \mathbf{main} \notin \xi \qquad \xi = \mathrm{dom}\ \Xi_{comp} \\ \xi; \Xi; [\cdot]\,; t; [\cdot]\,; [\cdot] \triangleright \mathbf{call\ main\ 0} \xrightarrow{\overline{a} \cdot \lightning ; t}{}^{*}_{\mathrm{ctx}} \Omega \triangleright f_{\lightning}\end{array}}{\mathbf{prog}\ \Xi_{ctx}\ \Xi_{comp} \overset{\overline{a} \cdot \lightning ; t}{\Longrightarrow} \Omega \triangleright f_{\lightning}}\;(e-\mathsf{wprog}-\lightning)$$

Figure 40: Running a whole L_{ms} program.

4.3.3 Proofs and Auxiliary Lemmas

Lemma 45 (Determinism of Step)**.** *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a_1} \Omega' \triangleright e_1\gamma'$

(b) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\mathbf{a_2}} \boldsymbol{\Omega}'' \triangleright \mathbf{e_2}\gamma''$

Then

(i) $\mathbf{e_1} = \mathbf{e_2}$

(ii) $\gamma' = \gamma''$

(iii) $\boldsymbol{\Omega}' = \boldsymbol{\Omega}''$

(iv) $\mathbf{a_1} = \mathbf{a_2}$

Proof. By induction on Assumption (a). □

Lemma 46 (Determinism of Ctx-Step)**.** *If*

(a) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\mathbf{a_2}}_{ctx} \boldsymbol{\Omega}' \triangleright \mathbf{e_1}\gamma'$

(b) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\mathbf{a_1}}_{ctx} \boldsymbol{\Omega}'' \triangleright \mathbf{e_2}\gamma''$

Then

(i) $\mathbf{e_1} = \mathbf{e_2}$

(ii) $\gamma' = \gamma''$

(iii) $\boldsymbol{\Omega}' = \boldsymbol{\Omega}''$

(iv) $\mathbf{a_1} = \mathbf{a_2}$

Proof. By induction on Assumption (a) making use of Lemma 45 (Determinism of Step). □

Lemma 47 (Determinism of n-Steps)**.** *If*

(a) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\overline{\mathbf{a_1}}}{}^{n}_{ctx} \boldsymbol{\Omega}' \triangleright \mathbf{e_1}\gamma'$

(b) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\overline{\mathbf{a_2}}}{}^{n}_{ctx} \boldsymbol{\Omega}'' \triangleright \mathbf{e_2}\gamma''$

Then

(i) $\mathbf{e_1} = \mathbf{e_2}$

(ii) $\gamma' = \gamma''$

(iii) $\boldsymbol{\Omega}' = \boldsymbol{\Omega}''$

(iv) $\overline{\mathbf{a_1}} = \overline{\mathbf{a_2}}$

Proof. By induction on Assumption (a) making use of Lemma 46 (Determinism of Ctx-Step). □

Lemma 48 (Determinism of Steps)**.** *If*

(a) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\overline{\mathbf{a_1}}}{}^{*}_{ctx} \boldsymbol{\Omega}' \triangleright \mathbf{f}^{\mathbf{1}}_{\lightning}\gamma'$

(b) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\overline{\mathbf{a_2}}}{}^{*}_{ctx} \boldsymbol{\Omega}'' \triangleright \mathbf{f}^{\mathbf{2}}_{\lightning}\gamma''$

Then

(i) $\mathbf{f}^1_{\lightning} = \mathbf{f}^2_{\lightning}$

(ii) $\gamma' = \gamma''$

(iii) $\mathbf{\Omega}' = \mathbf{\Omega}''$

(iv) $\overline{\mathbf{a_1}} = \overline{\mathbf{a_2}}$

Proof. By induction on Assumption (a) making use of Lemma 46 (Determinism of Ctx-Step). □

4.4 Robust TMS Preserving Compiler

4.4.1 Compiler

$\boxed{[\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{e}}$ „Compile $\mathrm{L_{tms}}$ expression e to $\mathrm{L_{ms}}$ expression $\mathbf{e}$.

$\boxed{[\![[\mathsf{v}/\mathsf{x}]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = [\mathbf{v}/\mathbf{x}]}$ „Compile $\mathrm{L_{tms}}$ substitution to $\mathrm{L_{ms}}$ substitution."

$\boxed{[\![\xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \xi}$ „Compile $\mathrm{L_{tms}}$ component library to $\mathrm{L_{ms}}$ component library."

$$
\begin{aligned}
[\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{f}_{\lightning}\\
[\![\mathsf{call\ foo\ e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{call}\ [\![\mathsf{foo}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{return\ e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{return}\ [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{e_1} \oplus \mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \oplus [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{x}[\mathsf{e}]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\,[[\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}]\\
[\![\mathsf{let\ x = e_1\ in\ e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{let}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = [\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{in}\ [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{x}[\mathsf{e_1}] \leftarrow \mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\,[[\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}] \leftarrow [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{let\ x = new\ e_1\ in\ e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{let}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{new}\ [\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{in}\ [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{delete\ x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{delete}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
[\![\mathsf{ifz\ e_1\ then\ e_2\ else\ e_3}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{ifz}\ [\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
&\quad\ \mathbf{then}\ [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\\
&\quad\ \mathbf{else}\ [\![\mathsf{e_3}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}
\end{aligned}
$$

$$
[\![[\mathsf{v}/\mathsf{x}]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = [\,[\![\mathsf{v}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} / [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\,]
$$

$$
\begin{aligned}
[\![[\cdot]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\cdot]\\
[\![\mathsf{foo}, \xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{foo}, [\![\xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}
\end{aligned}
$$

Figure 41: Compiler from $\mathrm{L_{tms}}$ to $\mathrm{L_{ms}}$.

$$\boxed{[\![\mathsf{F}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{F}} \quad \text{„Compile } \mathrm{L_{tms}} \text{ procedures to } \mathrm{L_{ms}} \text{ procedures."}$$

$$\begin{aligned}
[\![\mathsf{let\ foo\ x} : \mathbb{N} \to \tau_{\mathsf{e}} := \mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \quad = \quad & \mathbf{let}\ [\![\mathsf{foo}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} := \\
& \mathbf{ifz}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{has}\ \mathbb{N}\ \mathbf{then} \\
& \quad [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
& \mathbf{else} \\
& \quad \mathbf{abort}()
\end{aligned}$$

$$\boxed{[\![\Xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \boldsymbol{\Xi}} \quad \text{„Compile } \mathrm{L_{tms}} \text{ libraries to } \mathrm{L_{ms}} \text{ libraries."}$$

$$\begin{aligned}
[\![[\cdot]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\cdot] \\
[\![\mathsf{F}, \Xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{F}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}, [\![\Xi]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}
\end{aligned}$$

Figure 42: Compiler from $\mathrm{L_{tms}}$ components to $\mathrm{L_{ms}}$ components.

$$\boxed{[\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{K}} \quad \text{„Compile } \mathrm{L_{tms}} \text{ evaluation contexts to } \mathrm{L_{ms}} \text{ evaluation contexts."}$$

$$\begin{aligned}
[\![[\cdot]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\cdot] \\
[\![\mathsf{K} \oplus \mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \oplus [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{v} \oplus \mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{v}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \oplus [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{x}[\mathsf{K}]]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[[\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}] \\
[\![\mathsf{let\ x} = \mathsf{K\ in\ e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{let}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{in}\ [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{x}[\mathsf{K}] \leftarrow \mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[[\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}] \leftarrow [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{x}[\mathsf{v}] \leftarrow \mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[[\![\mathsf{v}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}] \leftarrow [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{let\ x} = \mathsf{new\ K\ in\ e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{let}\ [\![\mathsf{x}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{new}\ [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{in}\ [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{ifz\ K\ then\ e_1\ else\ e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{ifz}\ [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ \mathbf{then} \\
&\quad\ \ [\![\mathsf{e_1}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
&\quad\ \mathbf{else} \\
&\quad\ \ [\![\mathsf{e_2}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{call\ foo\ K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{call}\ [\![\mathsf{foo}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}\ [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \\
[\![\mathsf{return\ K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} &= \mathbf{return}\ [\![\mathsf{K}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}
\end{aligned}$$

Figure 43: Compiling $\mathrm{L_{tms}}$ evaluation contexts to $\mathrm{L_{ms}}$ evaluation contexts.

Compiling components requires a wrapper to ensure that target contexts invoke the compiled component with the right runtime terms. For example, by adding the dynamic type check we prevent contexts from binding $\langle \mathbf{42}, \mathbf{1729} \rangle$ to $\mathbf{x}$, which is never valid for $\mathrm{L_{tms}}$ components.

$$\boxed{\theta_{\bullet}\left(\mathbf{a}\right)=\mathbf{a}} \text{ „Filter an } \mathrm{L_{ms}} \text{ event."}$$

$$\frac{\begin{array}{c}\mathbf{a_b}=\lightning\vee\mathbf{a_b}=\mathbf{Alloc}\ \ell\ \mathbf{n}\vee\mathbf{a_b}=\mathbf{Dealloc}\ \ell\vee\\ \mathbf{a_b}=\mathbf{Get}\ \ell\ \mathbf{n}\vee\mathbf{a_b}=\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{m}\vee\\ \mathbf{a_b}=\mathbf{Call}\ \emptyset\ \mathbf{foo}\ \mathbf{n}\vee\mathbf{a_b}=\mathbf{Ret}\ \emptyset\ \mathbf{n}\end{array}}{\theta_{\bullet}\left(\mathbf{a_b};\mathbf{ctx}\right)=\varepsilon}\ (\text{trg-filter-context})$$

$$\frac{}{\theta_{\bullet}\left(\varepsilon;\mathbf{t}\right)=\varepsilon}\ (\text{trg-filter-unimportant})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Start};\mathbf{comp}\right)=\varepsilon}\ (\text{trg-filter-comp-start})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Alloc}\ \ell\ \mathbf{n};\mathbf{comp}\right)=\mathbf{Alloc}\ \ell\ \mathbf{n}}\ (\text{trg-filter-comp-alloc})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Dealloc}\ \ell;\mathbf{comp}\right)=\mathbf{Dealloc}\ \ell}\ (\text{trg-filter-comp-dealloc})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Get}\ \ell\ \mathbf{n};\mathbf{comp}\right)=\mathbf{Get}\ \ell\ \mathbf{n}}\ (\text{trg-filter-comp-get})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v};\mathbf{comp}\right)=\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}}\ (\text{trg-filter-comp-set})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Call}\ \mathbf{c}\ \mathbf{foo}\ \mathbf{v};\mathbf{t}\right)=\mathbf{Call}\ \mathbf{c}\ \mathbf{foo}\ \mathbf{v};\mathbf{t}}\ (\text{trg-filter-comp-call})$$

$$\frac{}{\theta_{\bullet}\left(\mathbf{Ret}\ \mathbf{c}\ \mathbf{v};\mathbf{t}\right)=\mathbf{Ret}\ \mathbf{c}\ \mathbf{v};\mathbf{t}}\ (\text{trg-filter-comp-ret})$$

$$\frac{}{\theta_{\bullet}\left(\lightning;\mathbf{t}\right)=\lightning}\ (\text{trg-filter-abort})$$

$$\boxed{\theta_{\bullet}^{*}\left(\overline{\mathbf{a}}\right)=\overline{\mathbf{a}}'} \text{ „Filter an } \mathrm{L_{ms}} \text{ trace."}$$

$$\frac{}{\theta_{\bullet}^{*}\left([\cdot]\right)=[\cdot]}\ (\text{trg-filter-empty})$$

$$\frac{\theta_{\bullet}\left(\mathbf{a}\right)=\mathbf{a}\neq\varepsilon \qquad \theta_{\bullet}^{*}\left(\overline{\mathbf{a}}\right)=\overline{\mathbf{a}}}{\theta_{\bullet}^{*}\left(\mathbf{a}\cdot\overline{\mathbf{a}}\right)=\mathbf{a}\cdot\overline{\mathbf{a}}}\ (\text{trg-filter-cons-relevant})$$

$$\frac{\theta_{\bullet}\left(\mathbf{a}\right)=\varepsilon \qquad \theta_{\bullet}^{*}\left(\overline{\mathbf{a}}\right)=\overline{\mathbf{a}}}{\theta_{\bullet}^{*}\left(\mathbf{a}\cdot\overline{\mathbf{a}}\right)=\overline{\mathbf{a}}}\ (\text{trg-filter-cons-relevant})$$

Figure 44: Filtering of $\mathrm{L_{ms}}$ events, getting rid of unimportant ones, for back-translation.

$\boxed{\mathsf{t} = \mathbf{t}}$ „Equality of sandboxtags between $\mathrm{L_{tms}}$ and $\mathrm{L_{ms}}$."

$$\frac{}{\mathsf{ctx} = \mathbf{ctx}}\ (\text{sandboxtag-ctx}) \qquad \frac{}{\mathsf{comp} = \mathbf{comp}}\ (\text{sandboxtag-comp})$$

$\boxed{\delta(\ell) = \ell}$ „Map from $\mathrm{L_{tms}}$ locations ℓ to $\mathrm{L_{ms}}$ locations ℓ."

$\boxed{\overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \overline{\mathbf{a}}}$ „The $\mathrm{L_{tms}}$ trace $\overline{\mathsf{a}}$ describes the same actions as $\mathrm{L_{ms}}$ trace $\overline{\mathbf{a}}$. Any action in X is ignored, these are generated by the backtranslation wrapper."

$$\frac{}{[\cdot] \approxeq^{*}_{\delta;\mathsf{X}} [\cdot]}\ (\text{empty-trace-eq}) \qquad \frac{\mathsf{a_b} \notin \mathsf{X} \quad \mathsf{t} = \mathbf{t} \quad \mathsf{a_b} \approxeq_{\delta} \mathbf{a_b} \quad \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \overline{\mathbf{a}}}{\mathsf{a_b};\mathsf{t} \cdot \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \mathbf{a_b};\mathbf{t} \cdot \overline{\mathbf{a}}}\ (\text{cons-trace-eq})$$

$$\frac{\mathsf{a_b} \in \mathsf{X} \quad \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \overline{\mathbf{a}}}{\mathsf{a_b};\mathsf{t} \cdot \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \overline{\mathbf{a}}}\ (\text{ignore-cons-trace-eq})$$

$\boxed{\mathsf{a_b} \approxeq_{\delta} \mathbf{a_b}}$ „The $\mathrm{L_{tms}}$ event $\mathsf{a_b}$ describes the same action as $\mathrm{L_{ms}}$ event $\mathbf{a_b}$."

$$\frac{}{\mathsf{Start} \approxeq_{\delta} \mathbf{Start}}\ (\text{start-event-eq}) \qquad \frac{\mathsf{v} = \langle\langle\mathbf{v}\rangle\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\partial}}{\mathsf{End}\ \mathsf{v} \approxeq_{\delta} \mathbf{End}\ \mathbf{v}}\ (\text{end-event-eq})$$

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{n} \quad \delta(\ell) = \ell}{\mathsf{Alloc}\ \ell\ \mathsf{n} \approxeq_{\delta} \mathbf{Alloc}\ \ell\ \mathbf{n}}\ (\text{alloc-event-eq}) \qquad \frac{\delta(\ell) = \ell}{\mathsf{Dealloc}\ \ell \approxeq_{\delta} \mathbf{Dealloc}\ \ell}\ (\text{dealloc-event-eq})$$

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{n} \quad \delta(\ell) = \ell}{\mathsf{Get}\ \ell\ \mathsf{n} \approxeq_{\delta} \mathbf{Get}\ \ell\ \mathbf{n}}\ (\text{get-event-eq})$$

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{n} \quad [\![\mathsf{v}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{v} \quad \delta(\ell) = \ell}{\mathsf{Set}\ \ell\ \mathsf{n}\ \mathsf{v} \approxeq_{\delta} \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}}\ (\text{set-event-eq})$$

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{n} \quad [\![\mathsf{foo}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{foo} \quad \mathsf{c} = \mathbf{c}}{\mathsf{Call}\ \mathsf{c}\ \mathsf{foo}\ \mathsf{n} \approxeq_{\delta} \mathbf{Call}\ \mathbf{c}\ \mathbf{foo}\ \mathbf{n}}\ (\text{call-event-eq})$$

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} = \mathbf{n} \quad \mathsf{c} = \mathbf{c}}{\mathsf{Ret}\ \mathsf{c}\ \mathsf{n} \approxeq_{\delta} \mathbf{Ret}\ \mathbf{c}\ \mathbf{n}}\ (\text{ret-event-eq}) \qquad \frac{}{\varepsilon \approxeq_{\delta} \varepsilon}\ (\varepsilon\text{-event-eq}) \qquad \frac{}{\lightning \approxeq_{\delta} \lightning}\ (\lightning\text{-event-eq})$$

$\boxed{\mathsf{c} = \mathbf{c}}$ „Communications are equal."

$$\frac{}{? = ?}\ (? = ?) \qquad \frac{}{! = !}\ (! = !) \qquad \frac{}{\varnothing = \varnothing}\ (\varnothing = \varnothing)$$

Figure 45: Trace Relation from $\mathrm{L_{tms}}$ to $\mathrm{L_{ms}}$.

$\boxed{\delta(\ell) = \ell}$ „The $\mathrm{L_{tms}}$ memory location ℓ corresponds to the $\mathrm{L_{ms}}$ memory location ℓ."

$\boxed{\Omega \approx_{\delta;\mathsf{L}} \mathbf{\Omega}}$ „The $\mathrm{L_{tms}}$ state Ω agrees with $\mathrm{L_{ms}}$ state $\mathbf{\Omega}$. L contains locations introduced by the backtranslation wrapper, which are subsequently ignored."

$$\frac{\Omega = \Phi; t; \Psi \quad \mathbf{\Omega} = \mathbf{\Phi}; \mathbf{t}; \mathbf{\Psi} \quad t = \mathbf{t} \quad \Phi \approx \mathbf{\Phi} \quad \Psi \approx_{\delta;\mathsf{L}} \mathbf{\Psi}}{\Omega \approx_{\delta;\mathsf{L}} \mathbf{\Omega}}\ \text{(state-eq)} \qquad \frac{}{\lightning \approx_{\delta;\mathsf{L}} \lightning}\ \text{(abort-state-eq)}$$

$\boxed{\Psi \approx_{\delta;\mathsf{L}} \mathbf{\Psi}}$ „The $\mathrm{L_{tms}}$ memory-state Ψ agrees with $\mathrm{L_{ms}}$ one $\mathbf{\Psi}$. L contains locations introduced by the backtranslation wrapper."

$$\frac{}{[\cdot]\,;[\cdot]\,;[\cdot] \approx_{\delta;\mathsf{L}} [\cdot]\,;[\cdot]\,;[\cdot]}\ \text{(empty-memstate-eq)}$$

$$\frac{\ell \notin \mathsf{L} \quad \mathsf{n} = \mathbf{n} \quad \rho = \rho \quad \delta(\ell) = \ell \quad \ell, \mathsf{n} \vdash_\delta \mathsf{H^{comp}} \approx \mathbf{H^{comp}} \quad \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{\Delta_2}}{\mathsf{H^{ctx}}; \mathsf{H^{comp}}; \mathsf{x} \mapsto (\ell; \mathsf{comp}; \rho; \mathsf{n}), \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{n}), \mathbf{\Delta_2}}\ \text{(comp-cons-memstate-eq)}$$

$$\frac{\ell \notin \mathsf{L} \quad \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta}}{\mathsf{H^{ctx}}; \mathsf{H^{comp}}; \mathsf{x} \mapsto (\ell; \mathsf{ctx}; \rho; \mathsf{n}), \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta}}\ \text{(ctx-cons-memstate-eq)}$$

$$\frac{\ell \in \mathsf{L} \quad \mathsf{H^{ctx}}; \mathsf{H^{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta}}{\mathsf{H^{ctx}}; \mathsf{H^{comp}}; \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta \approx_{\delta;\mathsf{L}} \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta}}\ \text{(whatever-cons-memstate-eq)}$$

$\boxed{\ell, \mathsf{n} \vdash_\delta \mathsf{H^{comp}} \approx \mathbf{H^{comp}}}$ „The heaps $\mathsf{H^{comp}}$ and $\mathbf{H^{comp}}$ are related at ℓ for n memory cells."

$$\frac{[\![\mathsf{n}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{n} \quad \delta(\ell) = \ell \quad \forall 0 \leq \mathsf{m} < \mathsf{n},\ \mathbf{0} \leq \mathbf{m} < \mathbf{n}, \mathsf{H^{comp}}(\ell + \mathsf{m}) = \mathbf{H^{comp}}(\ell + \mathbf{m})}{\ell, \mathsf{n} \vdash_\delta \mathsf{H^{comp}} \approx \mathbf{H^{comp}}}\ \text{(heap-related)}$$

$\boxed{\Phi \approx \mathbf{\Phi}}$ „The $\mathrm{L_{tms}}$ control-flow-state Φ agrees with $\mathrm{L_{ms}}$ one $\mathbf{\Phi}$."

$$\frac{\Phi = \xi; \Xi; \overline{\mathsf{K}} \quad \mathbf{\Phi} = \xi; \mathbf{\Xi}; \overline{\mathbf{K}} \quad \Xi \approx \mathbf{\Xi} \quad \overline{\mathsf{K}} \approx_\xi \overline{\mathbf{K}} \quad \xi = [\![\xi]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}}{\Phi \approx \mathbf{\Phi}}\ \text{(cfstate-eq)}$$

$\boxed{\rho = \rho}$ „$\mathrm{L_{tms}}$ poison equals $\mathrm{L_{ms}}$ one ρ."

$$\frac{}{☣ = ☣}\ \text{(☣-equal)} \qquad \frac{}{\square = \square}\ \text{(□-equal)}$$

Figure 46: State Relation from $\mathrm{L_{tms}}$ to $\mathrm{L_{ms}}$. This is meant to relate the states whenever we are „inside" a component.

$\boxed{\Xi \approx \Xi}$ „The procedures of L_{tms} agree with L_{ms} ones."

$$\frac{}{[\cdot] \approx \Xi}\ (\text{empty-commlib-lib-eq})$$

$$\frac{[\![\text{let foo x} : \tau_\lambda := \text{e}]\!]^{L_{tms} \to L_{ms}} = \textbf{let foo x} := \textbf{e} \qquad \Xi \approx \Xi_1, \Xi_2}{(\text{let foo x} : \tau_\lambda := \text{e}), \Xi \approx \Xi_1, (\textbf{let foo x} := \textbf{e}), \Xi_2}\ (\text{cons-commlib-eq})$$

$\boxed{\overline{\mathsf{K}} \approx_\xi \overline{\mathbf{K}}}$ „The stack of L_{tms} continuations $\overline{\mathsf{K}}$ agrees with L_{ms} one $\overline{\mathbf{K}}$. Only checks foo $\in \xi$"

$$\frac{}{[\cdot] \approx_\xi [\cdot]}\ (\text{empty-kontstack-eq})$$

$$\frac{\text{foo} \notin \xi \qquad \textbf{foo} = [\![\text{foo}]\!]^{L_{tms} \to L_{ms}} \qquad \overline{\mathsf{K}} \approx_\xi \overline{\mathbf{K}}}{\overline{\mathsf{K}} \approx_\xi (\mathbf{K}; \textbf{foo}), \overline{\mathbf{K}}}\ (\text{ign-cons-kontstack-eq})$$

$$\frac{\text{foo} \in \xi \qquad \text{foo} = \textbf{foo} \qquad \overline{\mathsf{K}} \approx_\xi \overline{\mathbf{K}}}{(\mathsf{K}; \text{foo}), \overline{\mathsf{K}} \approx_\xi (\mathbf{K}; \textbf{foo}), \overline{\mathbf{K}}}\ (\text{cons-kontstack-eq})$$

Figure 47: Memory-State Relations from L_{tms} to L_{ms}.

$\boxed{\delta(\ell) = \ell}$ „The L_{tms} memory location ℓ corresponds to the L_{ms} memory location ℓ."

$\boxed{\boldsymbol{\Omega} \multimap_{\delta;L} \Omega}$ „The L_{tms} state Ω agrees with L_{ms} state $\boldsymbol{\Omega}$. L contains locations introduced by the backtranslation wrapper, which are subsequently ignored."

$$\frac{\begin{array}{cc} \Omega = \Phi; \text{t}; \Psi & \boldsymbol{\Omega} = \boldsymbol{\Phi}; \textbf{t}; \boldsymbol{\Psi} \\ \boldsymbol{\Phi} \multimap \Phi & \boldsymbol{\Psi} \multimap_{\delta;L} \Psi \end{array}}{\boldsymbol{\Omega} \multimap_{\delta;L} \Omega}\ (\text{state-qe}) \qquad \frac{}{\lightning \multimap_{\delta;L} \lightning}\ (\text{abort-state-qe})$$

$\boxed{\boldsymbol{\Psi} \multimap_{\delta;L} \Psi}$ „The L_{tms} memory-state Ψ agrees with L_{ms} one $\boldsymbol{\Psi}$. L contains locations introduced by the backtranslation wrapper."

$$\frac{\Psi \approx_{\delta;L} \boldsymbol{\Psi}}{\boldsymbol{\Psi} \multimap_{\delta;L} \Psi}\ (\text{memstate-qe})$$

$\boxed{\boldsymbol{\Phi} \multimap \Phi}$ „The L_{tms} control-flow-state Φ agrees with L_{ms} one $\boldsymbol{\Phi}$."

$$\frac{\Phi \approx \boldsymbol{\Phi}}{\boldsymbol{\Phi} \multimap \Phi}\ (\text{cfstate-qe})$$

Figure 48: State Relation from L_{ms} to L_{tms}. This is meant to relate the states whenever we are „inside" a context.

Lemma 49 (Compiler Injective)**.** *If*

(a) $[\![\text{e}]\!]^{L_{tms} \to L_{ms}} = \textbf{e}$

(b) $[\![\text{e}]\!]^{L_{tms} \to L_{ms}} = \textbf{e}'$

Then

(i) $\textbf{e} = \textbf{e}'$

Proof. Simple induction on e. ∎

Lemma 50 (Substitution Commutes with $[\![\bullet]\!]^{L_{tms}\to L_{ms}}$)**.**

(i) $[\![e[v/x]]\!]^{L_{tms}\to L_{ms}} = [\![e]\!]^{L_{tms}\to L_{ms}}[\![[v/x]]\!]^{L_{tms}\to L_{ms}}$

Proof. Induction on e. ☐

Lemma 51 (Step Forward Simulation)**.** *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a} \Omega' \triangleright e'\gamma'$

(b) $\Omega \approx_{\delta;L} \boldsymbol{\Omega}$

(c) $\vdash \Omega \triangleright e\gamma : \mathbb{N}$

Then $\exists\delta'$ $\boldsymbol{\Omega'}$ $\mathbf{a}$*,*

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![e]\!]^{L_{tms}\to L_{ms}}[\![\gamma]\!]^{L_{tms}\to L_{ms}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega'} \triangleright [\![e']\!]^{L_{tms}\to L_{ms}}[\![\gamma']\!]^{L_{tms}\to L_{ms}}$

(iii) $a \approxeq_{\delta'} \mathbf{a}$

(iv) $\Omega' \approx_{\delta'} \boldsymbol{\Omega'}$

Proof. We proceed by induction on Assumption (a).

Case $e-\oplus$**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Omega \approx_{\delta;L} \boldsymbol{\Omega}$

(H_3) $\vdash \Omega \triangleright n_1 + n_2 : \mathbb{N}$

(H_4) $n_3 = n_1 + n_2$

Then $\exists\delta'$ $\boldsymbol{\Omega'}$ $\mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![n_1 + n_2]\!]^{L_{tms}\to L_{ms}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega'} \triangleright [\![n_3]\!]^{L_{tms}\to L_{ms}}$

(iii) $\varepsilon \approxeq_{\delta} \mathbf{a}$

(iv) $\Omega \approx_{\delta;L} \boldsymbol{\Omega'}$

Instantiate the goal with $\delta' = \delta$, $\boldsymbol{\Omega'} = \boldsymbol{\Omega}$, and $\mathbf{a} = \varepsilon$, so that what is left to prove is:

(i) $\delta \subseteq \delta$

(ii) $\boldsymbol{\Omega} \triangleright \mathbf{n_1} + \mathbf{n_2} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{n_3}$

(iii) $\varepsilon \approxeq_{\delta} \varepsilon$

(iv) $\Omega \approx_{\delta;L} \boldsymbol{\Omega}$

Goal (i) follows immediately from reflexivity of the subset relation.

Goal (ii) follows using Rule $e-\oplus$ and Assumption (H_4).

Goal (iii) follows using Rule ε-event-eq.

Goal (iv) follows using Assumption (H_2).

Case $e-\mathsf{delete}$**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta_2 \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

(H_3) $\vdash \Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta_2 \triangleright \mathsf{delete}\ \mathsf{x} : \ \mathbb{N}$

Then $\exists \delta'\ \boldsymbol{\Omega}'\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{delete}\ \mathsf{x}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright [\![0]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\mathsf{Dealloc}\ \ell \approxeq_{\delta} \mathbf{a}$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2 \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}'$

First note that $\boldsymbol{\Omega} = \mathbf{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}), \mathbf{\Delta_2}$, because of Assumption (H_2).

Instantiate the goal with $\delta' = \delta$, $\boldsymbol{\Omega}' = \mathbf{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \text{☣}; \mathbf{n}), \mathbf{\Delta_2}$, and $\mathbf{a} = \mathbf{Dealloc}\ \ell; \mathbf{t}'$ so that what is left to prove is:

(i) $\delta \subseteq \delta$

(ii) $\mathbf{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}), \mathbf{\Delta_2} \triangleright \mathbf{delete}\ \mathbf{x} \xrightarrow{\mathbf{Dealloc}\ \ell} \mathbf{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}$
$\mathbf{0}$

(iii) $\mathsf{Dealloc}\ \ell; \mathsf{t}' \approxeq_{\delta} \mathbf{Dealloc}\ \ell; \mathbf{t}'$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \text{☣}; \mathsf{n}), \Delta_2 \approx_{\delta;\mathsf{L}} \mathbf{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \text{☣}; \mathbf{n}), \mathbf{\Delta_2}$

Goal (i) follows immediately from reflexivity of the subset relation.

Goal (ii) follows using Rule $e-$ **delete**.

For Goal (iii), apply Rule dealloc-event-eq, what is left to show is $\delta(\ell) = \ell$.

Similarly for Goal (iv), apply Rule state-eq and Rule comp-cons-memstate-eq or Rule ctx-cons-memstate-eq (depending on the shape of t) „suitably" so that what is left to show is $\Phi \approx_{\delta;\mathsf{L}} \mathbf{\Phi}$, $\mathsf{H} \approx_{\delta;\mathsf{L}} \mathbf{H}$, $\Delta_1 \approx_{\delta;\mathsf{L}} \mathbf{\Delta_1}$, $\Delta_2 \approx_{\delta;\mathsf{L}} \mathbf{\Delta_2}$, $[\![\mathsf{x}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{x}$, and, like in the previous case, $\delta(\ell) = \ell$.

$[\![\mathsf{x}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{x}$ follows by definition of the compiler, anything else follows immediately by inverting Assumption (H_2) „suitably".

Case $e-\mathsf{get}-\in$**:** Analogous to next case, up to nested case analysis.

Case $e-\mathsf{get}-\notin$**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

(H_3) $\vdash \Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \triangleright \mathsf{x}[\mathsf{n}] : \ \mathbb{N}$

(H_4) $\ell \in \operatorname{dom} \mathsf{H}^{\mathsf{t}'} \implies \mathsf{v} = \mathsf{H}^{\mathsf{t}'}(\ell + \mathsf{n})$

(H_5) $\ell \notin \operatorname{dom} \mathsf{H}^{\mathsf{t}'} \implies \mathsf{v} = 1729$

Then $\exists \delta'\ \boldsymbol{\Omega}'\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\mathbf{\Omega} \triangleright [\![\mathsf{x}[\mathsf{n}]]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}} \xrightarrow{\mathbf{a}} \mathbf{\Omega}' \triangleright [\![\mathsf{v}]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}}$

(iii) $\mathsf{Get}\ \ell\ \mathsf{n}; \mathsf{t}' \approxeq_\delta \mathbf{a}$

(iv) $\Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \mathbf{\Omega}'$

First note that $\mathbf{\Omega} = \mathbf{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \mathbf{\Delta_2}$, because of Assumption (H_2).

Instantiate the goal with $\delta' = \delta$, $\mathbf{\Omega}' = \mathbf{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \mathbf{\Delta_2}$, and $\mathbf{a} = \mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{t}'$, so that what is left to prove is:

(i) $\delta \subseteq \delta$

(iii) $\mathsf{Get}\ \ell\ \mathsf{n}; \mathsf{t}' \approxeq_\delta \mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{t}'$

(iv) $\Phi; \mathsf{t}; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \mathbf{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \mathbf{\Delta_2}$

(ii) $\mathbf{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \mathbf{\Delta_2} \triangleright \mathbf{x}[\mathbf{n}] \xrightarrow{\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{t}'} \mathbf{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'$
$\mathbf{v}$

Goal (i) follows immediately from reflexivity of the subset relation.

For Goal (iii), invert Assumption (H_2) „suitably" to obtain the assumption $\delta(\ell) = \ell$. The claim then follows by applying Rule get-event-eq.

Goal (iv) follows immediately from Assumption (H_2).

Goal (ii) is a bit technical. First apply Rules $e - \mathbf{get} - \in$ and $e - \mathbf{get} - \notin$, what is left to show is:

(v) $\ell \in \operatorname{dom} \mathbf{H}^{\mathbf{t}'} \implies \mathbf{v} = \mathbf{H}^{\mathbf{t}'}(\ell + \mathbf{n})$

(vi) $\ell \notin \operatorname{dom} \mathbf{H}^{\mathbf{t}'} \implies \mathbf{v} = \mathbf{1729}$

Now, note that $\ell \in \operatorname{dom} \mathsf{H}^{\mathsf{t}'} \Leftrightarrow \delta(\ell) = \ell \in \operatorname{dom} \mathbf{H}^{\mathbf{t}'}$. We continue with a case distinction.

Case $\ell \in \operatorname{dom} \mathsf{H}^{\mathsf{t}'}$**:**

Assumption (H_4) gives $\mathsf{v} = \mathsf{H}^{\mathsf{t}'}(\ell + \mathsf{n})$. Since $\mathbf{v} = [\![\mathsf{v}]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}}$, $\mathbf{v} = [\![\mathsf{H}^{\mathsf{t}'}(\ell + \mathsf{n})]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}}$. Using Assumption (H_2), $\mathbf{v} = \mathbf{H}^{\mathbf{t}'}(\ell + \mathbf{n})$, done.

Case $\ell \notin \operatorname{dom} \mathsf{H}^{\mathsf{t}'}$**:**

Assumption (H_5) gives $\mathsf{v} = \mathsf{1729}$. Since $\mathbf{v} = [\![\mathsf{v}]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}}$, $\mathbf{v} = [\![\mathsf{1729}]\!]^{\mathsf{L_{tms}} \to \mathsf{L_{ms}}} = \mathbf{1729}$, done.

Case $e - \mathsf{set} - \in$**:** Analogous to next case, up to nested case analysis.

Case $e - \mathsf{set} - \notin$**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \mathbf{\Omega}$

(H_3) $\vdash \Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \triangleright \mathsf{x}[\mathsf{n}] : \mathbb{N}$

(H_4) $\mathsf{H}' = \mathsf{H}(\ell + \mathsf{n} \mapsto \mathsf{v})$

Then $\exists \delta'\ \mathbf{\Omega}'\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{x}[\mathsf{n}] \leftarrow \mathsf{v}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright [\![\mathsf{v}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\mathsf{Set}\ \ell\ \mathsf{n}; \mathsf{t}' \approxeq_{\delta} \mathbf{a}$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}'}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}'$

First note that $\boldsymbol{\Omega} = \boldsymbol{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \boldsymbol{\Delta_2}$, because of Assumption (H_2).

Instantiate the goal with $\delta' = \delta$, $\boldsymbol{\Omega}' = \boldsymbol{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}'}; \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \boldsymbol{\Delta_2}$, and $\mathbf{a} = \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{m}$, so that what is left to prove is:

(i) $\delta \subseteq \delta$

(ii) $\boldsymbol{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \boldsymbol{\Delta_2} \triangleright \mathbf{x}[\mathbf{n}] \leftarrow \mathbf{v} \xrightarrow{\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{m}; \mathbf{t}'}$ $\boldsymbol{\Phi}; \mathbf{t}; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}'}; \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \boldsymbol{\Delta_2} \triangleright \mathbf{v}$

(iii) $\mathsf{Set}\ \ell\ \mathsf{n}\ \mathsf{m}; \mathsf{t}' \approxeq_{\delta} \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{m}; \mathbf{t}'$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}'}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}'), \Delta_2 \approx_{\delta;\mathsf{L}} \boldsymbol{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}'}; \boldsymbol{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{t}; \rho; \mathbf{n}'), \boldsymbol{\Delta_2}$

Goal (i) follows immediately from reflexivity of the subset relation.

For Goal (ii), apply Rules $e-\mathbf{set}-ctx$ and $e-\mathbf{set}-comp$ depending on the shape of $\mathbf{t}$. What is left to prove is $\mathbf{H}^{\mathbf{t}'} = \mathbf{H}^{\mathbf{t}}(\ell + \mathbf{n} \mapsto \mathbf{v})$. This follows by Assumption (H_2).

Apply Rule set-event-eq on Goal (iii), what is left to show is $[\![\mathsf{n}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{n}$, which is trivial, and $\delta(\ell) = \ell$. The latter can be obtained by a „suitable" inversion of Assumption (H_2).

Goal (iv) is assumed in Assumption (H_2).

Case $e - \mathsf{new}$**:**

If

(H_1) $\gamma = [\cdot]$

(H_2) $\gamma' = [\mathsf{z}/\mathsf{x}]$

(H_3) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

(H_4) $\vdash \Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \triangleright \mathsf{let}\ \mathsf{x}\ =\ \mathsf{new}\ \mathsf{n}\ \mathsf{in}\ \mathsf{e} : \ \mathbb{N}$

(H_5) $\mathsf{H}^{\mathsf{t}'} = \mathsf{H}^{\mathsf{t}'} \ll \mathsf{n}$

Then $\exists \delta'\ \boldsymbol{\Omega}'\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{let}\ \mathsf{x}\ =\ \mathsf{new}\ \mathsf{n}\ \mathsf{in}\ \mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} [\![[\mathsf{z}/\mathsf{x}]]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\mathsf{Alloc}\ \ell\ \mathsf{n}; \mathsf{t}' \approxeq_{\delta} \mathbf{a}$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}'}; \mathsf{z} \mapsto (\ell; \mathsf{t}; \Box; \mathsf{n}), \Delta \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}'$

First note that $\boldsymbol{\Psi} = \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{z} \mapsto (\ell; \mathbf{t}; \Box; \mathbf{n}), \boldsymbol{\Delta}$, because of Assumption (H_3).

Instantiate the goal with $\delta' = \delta \cup \{\ell \mapsto \ell\}$, $\boldsymbol{\Omega}' = \boldsymbol{\Phi}; \mathbf{t}'; \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{z} \mapsto (\ell; \mathbf{t}; \Box; \mathbf{n}), \boldsymbol{\Delta}$, and $\mathbf{a} = \mathbf{Alloc}\ \ell\ \mathbf{n}; \mathbf{t}'$, so that what is left to prove is:

(i) $\delta \subseteq \delta \cup \{\ell \mapsto \ell\}$

(ii) $\mathbf{\Phi; t'; H^{ctx}; H^{comp}; \Delta \triangleright let\ x\ =\ new\ n\ in\ } [\![e]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\mathbf{Alloc}\ \ell\ \mathbf{n; t'}}$
$\mathbf{\Phi; t'; H^{ctx}; H^{comp'}; z \mapsto (\ell; t; \square; n), \Delta} \triangleright [\![e]\!]^{L_{tms} \to L_{ms}}[\mathbf{z}/\mathbf{x}]$

(iii) $\mathsf{Alloc}\ \ell\ \mathsf{n}; \mathsf{t}' \approxeq_{\delta \cup \{\ell \mapsto \ell\}} \mathbf{Alloc}\ \ell\ \mathbf{n}; \mathbf{t}'$

(iv) $\Phi; \mathsf{t}'; \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}'}; \mathsf{z} \mapsto (\ell; \mathsf{t}; \square; \mathsf{n}), \Delta \approx_{\delta \cup \{\ell \mapsto \ell\}} \mathbf{\Phi; t'; H^{ctx}; H^{comp'}; z \mapsto (\ell; t; \square; n), \Delta}$

Goal (i) follows by rewriting it as $\forall x, x \in \delta \implies x \in \delta \vee x \in \{\ell \mapsto \ell\}$, then just choose the left side of the disjunction.

Apply Rules $e-\mathbf{new}-ctx$ and $e-\mathbf{new}-comp$ depending on the shape of t on Goal (ii), what is left to show is $\mathbf{H^t} = \mathbf{H^t} \ll \mathbf{n}$. This follows by Assumption (H_3).

Apply Rule set-event-eq on Goal (iii), what is left to show is $[\![\mathsf{n}]\!]^{L_{tms} \to L_{ms}} = \mathbf{n}$, follows immediately by definition of the compiler, and $(\delta \cup \{\ell \mapsto \ell\})(\ell) = \ell$, which follows by definition.

Apply Rules state-eq, comp-cons-memstate-eq and ctx-cons-memstate-eq depending on the shape of t on Goal (iv), what is left to show is $[\![\mathsf{x}]\!]^{L_{tms} \to L_{ms}} = \mathbf{x}$, follows immediately by definition of the compiler, and $(\delta \cup \{\ell \mapsto \ell\})(\ell) = \ell$, which follows by definition.

Case $e-\mathsf{abort}$**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Omega \approx_{\delta;\mathsf{L}} \mathbf{\Omega}$

(H_3) $\vdash \Omega \triangleright \mathsf{abort}() : \mathbb{N}$

Then $\exists \delta'\ \mathbf{\Omega'}\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\mathbf{\Omega} \triangleright [\![\mathsf{abort}()]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\mathbf{a}} \mathbf{\Omega'} \triangleright [\![\mathsf{stuck}]\!]^{L_{tms} \to L_{ms}}$

(iii) $\lightning; \mathsf{t} \approxeq_{\delta} \mathbf{a}$

(iv) $\lightning \approx_{\delta;\mathsf{L}} \mathbf{\Omega'}$

Instantiate the goal with $\delta' = \delta$, $\mathbf{\Omega'} = \lightning$, and $\mathbf{a} = \lightning; \mathbf{t}$. All goals go through easily.

Case $e-\mathsf{let}-\mathsf{f}$**:**

If

(H_1) $\gamma = [\cdot]$

(H_2) $\gamma' = [\mathsf{v}/\mathsf{x}]$

(H_3) $\Omega \approx_{\delta;\mathsf{L}} \mathbf{\Omega}$

(H_4) $\vdash \Omega \triangleright \mathsf{let\ x\ = v\ in\ e} : \mathbb{N}$

Then $\exists \delta'\ \mathbf{\Omega'}\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{let}\ \mathsf{x}\ = \mathsf{v}\ \mathsf{in}\ \mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} [\![[\mathsf{v}/\mathsf{x}]]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\varepsilon \approxeq_{\delta} \mathbf{a}$

(iv) $\Omega \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}'$

Instantiate the goal with $\delta' = \delta$, $\boldsymbol{\Omega}' = \boldsymbol{\Omega}$, and $\mathbf{a} = \varepsilon$, so that what is left to prove is:

(i) $\delta \subseteq \delta$

(ii) $\boldsymbol{\Omega} \triangleright \mathbf{let}\ \mathbf{x}\ = \mathbf{v}\ \mathbf{in}\ [\![\mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} [\mathbf{v}/\mathbf{x}]$

(iii) $\varepsilon \approxeq_{\delta} \varepsilon$

(iv) $\Omega \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

Goal (i) follows by reflexivity of the subset relation.

Goal (ii) follows by definition of Rule $e - \mathbf{let} - \mathbf{f}$.

Goal (iii) follows by definition of Rule ε-event-eq.

Goal (iv) follows by Assumption (H_3).

Case $e - \mathsf{ifz}-$true**:**

If

(H_1) $\gamma = \gamma' = [\cdot]$

(H_2) $\Omega \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

(H_3) $\vdash \Omega \triangleright \mathsf{ifz}\ 0\ \mathsf{then}\ \mathsf{e}_1\ \mathsf{else}\ \mathsf{e}_2 : \mathbb{N}$

Then $\exists \delta'\ \boldsymbol{\Omega}'\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{ifz}\ 0\ \mathsf{then}\ \mathsf{e}_1\ \mathsf{else}\ \mathsf{e}_2]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright [\![\mathsf{e}_1]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\varepsilon \approxeq_{\delta} \mathbf{a}$

(iv) $\Omega \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}'$

Instantiate the goal with $\delta' = \delta$, $\boldsymbol{\Omega}' = \boldsymbol{\Omega}$, and $\mathbf{a} = \varepsilon$, so that what is left to prove is:

(i) $\delta \subseteq \delta$

(ii) $\boldsymbol{\Omega} \triangleright \mathbf{ifz}\ \mathbf{0}\ \mathbf{then}\ [\![\mathsf{e}_1]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}\ \mathbf{else}\ [\![\mathsf{e}_2]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright [\![\mathsf{e}_1]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(iii) $\varepsilon \approxeq_{\delta} \varepsilon$

(iv) $\Omega \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega}$

Goal (i) follows by reflexivity of the subset relation.

Goal (ii) follows by definition of Rule $e - \mathbf{ifz}-$true.

Goal (iii) follows by definition of Rule ε-event-eq.

Goal (iv) follows by Assumption (H_2).

Case $e - \mathsf{ifz}-$false**:**

This case is completely analogous to the previous case, the only difference is the use of Rule $e - \mathbf{ifz}-$false instead of Rule $e - \mathbf{ifz}-$true.

$\square$

Lemma 52 (Ctx Step Forward Simulation). *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a}_{ctx} \Omega' \triangleright e'\gamma'$

(b) $\Omega \approx_{\delta;L} \boldsymbol{\Omega}$

(c) $\vdash \Omega \triangleright e\gamma : \mathbb{N}$

Then $\exists \delta'\ \boldsymbol{\Omega'}\ \mathbf{a}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![e\gamma]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\mathbf{a}}_{ctx} \boldsymbol{\Omega'} \triangleright [\![e'\gamma']\!]^{L_{tms} \to L_{ms}}$

(iii) $a \approxeq_{\delta} \mathbf{a}$

(iv) $\Omega' \approx_{\delta;L} \boldsymbol{\Omega'}$

Proof. Induction on Assumption (a). $\square$

Lemma 53 (Forward Simulation). *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{\bar{a}}{}^{*}_{ctx} \Omega' \triangleright f_{\natural}\gamma'$

(b) $\Omega \approx_{\delta;L} \boldsymbol{\Omega}$

(c) $\vdash \Omega \triangleright e\gamma : \mathbb{N}$

Then $\exists \delta'\ \boldsymbol{\Omega'}\ \bar{\mathbf{a}}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![e]\!]^{L_{tms} \to L_{ms}} [\![\gamma]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\bar{\mathbf{a}}}{}^{*}_{ctx} \boldsymbol{\Omega'} \triangleright [\![f_{\natural}]\!]^{L_{tms} \to L_{ms}} [\![\gamma']\!]^{L_{tms} \to L_{ms}}$

(iii) $\bar{a} \approxeq^{*}_{\delta'} \bar{\mathbf{a}}$

(iv) $\Omega' \approx_{\delta'} \boldsymbol{\Omega'}$

Proof. Induction on Assumption (a).

Case $es-$refl**:** Trivial. Witnesses are $\delta, \boldsymbol{\Omega}$, and $[\cdot]$, goals follow immediately by reflexivity of the respective relation.

Case $es-$trans$-$important**:** Very similar to next case.

Case $es-$trans$-$unimportant**:**

(H_1) $\Omega \triangleright e\gamma \xrightarrow{a}_{\text{ctx}} \Omega_0 \triangleright e_0\gamma_0$

(H_2) $\Omega_0 \triangleright e_0\gamma_0 \xrightarrow{\bar{a}}{}^{*}_{\text{ctx}} \Omega' \triangleright f_{\natural}\gamma'$

With the inductive hypothesis: $\forall \delta_{\forall}\ \boldsymbol{\Omega_{\forall}}$, If

$\left(IH_1^{(as)}\right)$ $\Omega_0 \approx_{\delta_{\forall}} \boldsymbol{\Omega_{\forall}}$

$\left(IH_2^{(as)}\right)$ $\vdash \Omega_0 \triangleright e_0\gamma_0 : \mathbb{N}$

Then $\exists \delta_{IH}\ \boldsymbol{\Omega_{\mathbb{IH}}}\ \overline{\mathbf{a_{\mathbb{IH}}}}$,

(IH_1) $\delta_\forall \subseteq \delta_{IH}$

(IH_2) $\boldsymbol{\Omega_\forall} \triangleright [\![\mathsf{e_0}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma_0]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\overline{\mathbf{a_{\mathbb{IH}}}}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega_{\mathbb{IH}}} \triangleright [\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma']\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

(IH_3) $\overline{\mathsf{a}} \approxeq^{*}_{\delta_{IH}} \overline{\mathbf{a_{\mathbb{IH}}}}$

(IH_4) $\Omega' \approx_{\delta_{IH}} \boldsymbol{\Omega_{\mathbb{IH}}}$

Now apply Lemma 52 (Ctx Step Forward Simulation) on Assumption (H_1) using Assumptions (b) and (c), giving us witnesses $\delta_0\ \boldsymbol{\Omega_0}\ \mathbf{a}$,

(F_1) $\delta \subseteq \delta_0$

(F_2) $\boldsymbol{\Omega} \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\mathbf{a}}_{\mathrm{ctx}} \boldsymbol{\Omega_0} \triangleright [\![\mathsf{e_0}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma_0]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

(F_3) $\mathsf{a} \approxeq_{\delta_0} \mathbf{a}$

(F_4) $\Omega_0 \approx_{\delta_0} \boldsymbol{\Omega_0}$

Apply Lemma 37 (Ctx Preservation) on Assumptions (H_1) and (c):

(F_5) $\vdash \Omega_0 \triangleright \mathsf{e_0}\gamma_0 : \mathbb{N}$

Instantiate the inductive hypothesis with $\delta_\forall = \delta_0$ and $\boldsymbol{\Omega_\forall} = \boldsymbol{\Omega_0}$, provide Assumption (F_4) and Assumption (F_5), and obtain witnesses $\delta_{IH}, \boldsymbol{\Omega_{\mathbb{IH}}}$, and $\overline{\mathbf{a_{\mathbb{IH}}}}$ such that:

(IH_1') $\delta_0 \subseteq \delta_{IH}$

(IH_2') $\boldsymbol{\Omega_0} \triangleright [\![\mathsf{e_0}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma_0]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\overline{\mathbf{a_{\mathbb{IH}}}}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega_{\mathbb{IH}}} \triangleright [\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}[\![\gamma']\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

(IH_3') $\overline{\mathsf{a}} \approxeq^{*}_{\delta_{IH}} \overline{\mathbf{a_{\mathbb{IH}}}}$

(IH_4') $\Omega' \approx_{\delta_{IH}} \boldsymbol{\Omega_{\mathbb{IH}}}$

Our goal looks as follows: $\exists \delta'\ \boldsymbol{\Omega'}\ \overline{\mathbf{a'}}$,

(i) $\delta \subseteq \delta'$

(ii) $\boldsymbol{\Omega} \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega'} \triangleright [\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

(iii) $\mathsf{a} \cdot \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \overline{\mathbf{a'}}$

(iv) $\Omega' \approx_{\delta;\mathsf{L}} \boldsymbol{\Omega'}$

Case $\mathsf{a} \neq \varepsilon$:

(H_1) $\mathsf{a} \neq \varepsilon$

Instantiate the goal with $\delta_{IH}, \boldsymbol{\Omega_{\mathbb{IH}}}, \mathbf{a} \cdot \overline{\mathbf{a}}$. Note that $\delta \subseteq \delta_{IH}$ (Goal (i)) follows by transitivity using Assumptions (F_1) and (IH_1'). Similarily for Goal (iv).

Apply Rule $et-$trans$-$important on Goal (ii). So, what is left to show is

(iii) $\mathsf{a} \cdot \overline{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}} \mathbf{a} \cdot \overline{\mathbf{a}}$

(v) $\boldsymbol{\Omega} \triangleright [\![\mathsf{e}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\mathbf{a}}_{\mathrm{ctx}} \boldsymbol{\Omega_0} \triangleright [\![\mathsf{e_0}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

(vi) $\boldsymbol{\Omega_0} \triangleright [\![\mathsf{e_0}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega_{\mathbb{IH}}} \triangleright [\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$

Goal (iii) follows from Assumptions (IH_3) and (F_3) using Rule cons-trace-eq. Goal (v) is proven by Assumption (F_2). Goal (vi) is proven by Assumption (IH'_2).

Case $\mathsf{a} = \varepsilon$**:** Similar to the other case, but instantiate with $\overline{\mathbf{a}_{\mathbf{IH}}}$ instead of $\mathbf{a} \cdot \overline{\mathbf{a}_{\mathbf{IH}}}$ and use Rule $et-$trans$-$unimportant

$\square$

Lemma 54 (Different Reduction)**.** *If*

(a) $\neg\left(\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{ctx} \Omega' \triangleright \mathsf{f}_{\lightning}\right)$

(b) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{ctx} \Omega' \triangleright \mathsf{f}'_{\lightning}$

Then

(i) $\mathsf{f}_{\lightning} \neq \mathsf{f}'_{\lightning}$

Proof. Assume $\mathsf{f}_{\lightning} = \mathsf{f}'_{\lightning}$, apply Assumption (a) to the goal $\perp$, rewrite in the goal using $\mathsf{f}_{\lightning} = \mathsf{f}'_{\lightning}$ and solve the goal by Assumption (b). $\square$

Lemma 55 (Expression Correctness)**.** *If*

(a) $\mathbf{\Omega} \triangleright [\![\mathsf{e}]\!]^{\mathsf{L}_{tms}\to\mathsf{L}_{ms}}[\![\gamma]\!]^{\mathsf{L}_{tms}\to\mathsf{L}_{ms}} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{ctx} \mathbf{\Omega}' \triangleright [\![\mathsf{f}_{\lightning}]\!]^{\mathsf{L}_{tms}\to\mathsf{L}_{ms}}[\![\gamma']\!]^{\mathsf{L}_{tms}\to\mathsf{L}_{ms}}$

(b) $\Omega \approx_{\delta;\mathsf{L}} \mathbf{\Omega}$

(c) $\vdash \Omega \triangleright \mathsf{e}\gamma : \mathbb{N}$

Then $\exists \delta'\ \Omega'\ \overline{\mathsf{a}}$*,*

(i) $\delta \subseteq \delta'$

(ii) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{ctx} \Omega' \triangleright \mathsf{f}_{\lightning}\gamma'$

(iii) $\Omega' \approx_{\delta';\mathsf{L}} \mathbf{\Omega}'$

(iv) $\overline{\mathsf{a}} \approx^{*}_{\delta';\mathsf{X}} \overline{\mathbf{a}}$

Proof. We prove this by contradiction. So, we get the assumption:

(I_1) $\forall \delta'\ \Omega'\ \overline{\mathsf{a}}, \delta' \subset \delta \vee \neg\left(\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{\text{ctx}} \Omega' \triangleright \mathsf{f}_{\lightning}\gamma'\right) \vee \Omega' \not\approx_{\delta'} \mathbf{\Omega}' \vee \overline{\mathsf{a}} \not\approx^{*}_{\delta'} \overline{\mathbf{a}}$

and need to prove $\perp$. Apply Lemma 40 (Progress) on Assumption (c), so:

(I_2) $\exists \Omega'\ \mathsf{f}_{\lightning}\ \overline{\mathsf{a}}, \Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{\text{ctx}} \Omega' \triangleright \mathsf{f}_{\lightning}$

First extract the witnesses and obtain:

(H_1) $\Omega \triangleright \mathsf{e}\gamma \xrightarrow{\overline{\mathsf{a}}}{}^{*}_{\text{ctx}} \Omega' \triangleright \mathsf{f}'_{\lightning}$

Apply Lemma 53 (Forward Simulation) to Assumption (H_1) with Assumptions (b) and (c) to get:

(F_1) $\delta \subseteq \delta_v$

(F_2) $\Omega \triangleright [\![e]\!]^{L_{tms} \to L_{ms}} [\![\gamma]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\overline{a_v}}{}^{*}_{ctx} \Omega'_v \triangleright [\![f'_{\not{\downarrow}}]\!]^{L_{tms} \to L_{ms}}$

(F_3) $\Omega' \approx_{\delta_v} \Omega'_v$

(F_4) $\overline{a} \approxeq^{*}_{\delta_v} \overline{a_v}$

Use Lemma 48 (Determinism of Steps) on Assumptions (a) and (F_2), giving us:

(K_1) $[\![f_{\not{\downarrow}}]\!]^{L_{tms} \to L_{ms}} = [\![f'_{\not{\downarrow}}]\!]^{L_{tms} \to L_{ms}}$

(K_2) $\Omega' = \Omega'_v$

(K_3) $\overline{a} = \overline{a_v}$

Rewrite using Assumption (K_2) in Assumption (F_3), similarily Assumption (K_3) in Assumption (F_4):

(F'_3) $\Omega' \approx_{\delta_v} \Omega'$

(F'_4) $\overline{a} \approxeq^{*}_{\delta_v} \overline{a}$

Specialize Assumption (I_1) for $\delta' = \delta_v$, $\Omega' = \Omega'$, and $\overline{a} = \overline{a}$. We proceed by case analysis:

Case $\neg(\delta \subseteq \delta_v)$**:** Apply $\neg(\delta \subseteq \delta_v)$ on our goal $\bot$, then $\delta \subseteq \delta_v$ immediately follows from Assumption (F_1).

Case $\neg\left(\Omega \triangleright e\gamma \xrightarrow{\overline{a}}{}^{*}_{ctx} \Omega' \triangleright f_{\not{\downarrow}}\right)$**:** Using Assumption (H_1) and the assumption $\neg\left(\Omega \triangleright e\gamma \xrightarrow{\overline{a}}{}^{*}_{ctx} \Omega' \triangleright f'_{\not{\downarrow}}\right)$, we conclude $f_{\not{\downarrow}} \neq f'_{\not{\downarrow}}$ using Lemma 54 (Different Reduction). Apply Lemma 49 (Compiler Injective) on Assumption (K_1), so $f_{\not{\downarrow}} = f'_{\not{\downarrow}}$, contradicting the above.

Case $\Omega' \not\approx_{\delta_v} \Omega'$**:** Immediate contradiction with Assumption (F'_3).

Case $\overline{a} \not\approxeq^{*}_{\delta_v} \overline{a}$**:** Immediate contradiction with Assumption (F'_4).

□

Lemma 56 (Component Correctness)**.** *If*

(a) $\Omega \triangleright [\![e]\!]^{L_{tms} \to L_{ms}} \xrightarrow{\overline{a}}{}^{n}_{ctx} \Omega' \triangleright \mathbf{K}[\mathbf{return\ v}]$

(b) $\Omega \approx_{\delta;L} \Omega$

(c) $\vdash \Omega \triangleright e : \mathbb{N}$

Then $\exists \delta'\ \Omega'\ \overline{a}$

(i) $\delta \subseteq \delta'$

(ii) $\Omega \triangleright e \xrightarrow{\overline{a}}{}^{n}_{ctx} \Omega' \triangleright \mathbf{K}[\mathbf{return\ v}]$

(iii) $\Omega' \approx_{\delta;L} \Omega'$

(iv) $\overline{a} \approxeq^{*}_{\delta;X} \overline{a}$

Proof. □

4.4.2 Trace-based Backtranslation

$\boxed{\langle\langle\langle \mathbf{v} \rangle\rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{v}}$ „Map an L_{ms} value to an L_{tms} value."

$$\frac{}{\langle\langle\langle \mathbf{n} \rangle\rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{n}}\ \text{(backtrans-value)}$$

Figure 49: Backtranslation of L_{ms} values to L_{tms} values.

$\boxed{\langle\langle \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}$ „Backtranslate a non-interface trace $\overline{\mathbf{a}}$ to an L_{tms} expression."

$$\frac{}{\langle\langle [\cdot] \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{42}}\ \text{(backtrans-empty)}$$

$$\frac{\langle\langle \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}{\langle\langle \mathbf{Alloc}\ \ell\ \mathbf{s}; \mathbf{t} \cdot \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}\ \text{(backtrans-alloc)}$$

$$\frac{\langle\langle \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}{\langle\langle \mathbf{Dealloc}\ \ell; \mathbf{t} \cdot \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}\ \text{(backtrans-dealloc)}$$

$$\frac{\langle\langle \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}{\langle\langle \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}; \mathbf{t} \cdot \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}\ \text{(backtrans-set)}$$

$$\frac{\langle\langle \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}{\langle\langle \mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{t} \cdot \overline{\mathbf{a}} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{e}}\ \text{(backtrans-get)}$$

$$\frac{}{\langle\langle \lightning; \mathbf{t} \rangle\rangle^{L_{ms}\to L_{tms}} = \mathsf{abort}()}\ \text{(backtrans-abort)}$$

Figure 50: Trace-Based Backtranslation from L_{ms} backtranslation-events to L_{tms} terms.

$\boxed{\eth(\ell) = \mathsf{x}}$ „Map an L_{tms} location to an L_{tms} identifier."

$\boxed{{}^{!}\langle\langle \mathbf{a} \rangle\rangle^{L_{ms}\to L_{tms}}_{\eth;\delta;\overline{\ell}} = \overline{\ell'}; \mathsf{e}}$ „Given a `VarMap` $\eth$, δ, and $\overline{\ell}$, yield another stack of locations $\overline{\ell'}$, and an L_{tms} expression from target !-interaction-event $\mathbf{a}$."

$$\frac{}{{}^{!}\langle\langle \mathbf{Start} \rangle\rangle^{L_{ms}\to L_{tms}}_{\emptyset;\emptyset;[\cdot]} = [\cdot]; \mathsf{42}}\ \text{(backtrans-start)}$$

$$\frac{\delta(\ell) = \ell \qquad \mathsf{z} = \eth(\ell) \qquad \mathsf{e} = \mathsf{delete}\ \mathsf{z}}{{}^{!}\langle\langle \mathbf{Ret}\ !\ \mathbf{v} \rangle\rangle^{L_{ms}\to L_{tms}}_{\eth;\delta;\ell,\overline{\ell}} = \overline{\ell}; \mathsf{e}; \langle\langle\langle \mathbf{v} \rangle\rangle\rangle^{L_{ms}\to L_{tms}}}\ \text{(backtrans-ret-comptoctx)}$$

$\boxed{{}^{?}\langle\langle \mathbf{a} \rangle\rangle^{L_{ms}\to L_{tms}}_{\eth;\delta;\overline{\ell}} = \eth'; \delta'; \overline{\ell'}; \mathsf{e}}$ „Given a `VarMap` $\eth$, δ, and $\overline{\ell}$, construct a `VarMap` $\eth'$, a δ', another stack of locations $\overline{\ell'}$, and an L_{tms} expression from target ?-interaction-event $\mathbf{a}$."

$$\frac{\begin{array}{c}\eth, \delta \vdash \ell\ \mathit{fresh} \qquad \eth \vdash \mathsf{z}\ \mathit{fresh} \qquad \delta \vdash \ell\ \mathit{fresh} \\ \eth_z = \eth \cup \{\ell \mapsto \mathsf{z}\} \qquad \delta' = \delta \cup \{\ell \mapsto \ell\} \\ \mathsf{e} = \mathsf{let}\ \mathsf{z} = \mathsf{new}\ \mathsf{42}\ \mathsf{in}\ \mathsf{call}\ \langle\langle \mathbf{foo} \rangle\rangle^{L_{ms}\to L_{tms}}\ \langle\langle\langle \mathbf{v} \rangle\rangle\rangle^{L_{ms}\to L_{tms}}\end{array}}{{}^{?}\langle\langle \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v} \rangle\rangle^{L_{ms}\to L_{tms}}_{\eth;\delta;\overline{\ell}} = \eth_z; \delta'; \ell, \overline{\ell}; \mathsf{e}}\ \text{(backtrans-call-ctxtocomp)}$$

$$\frac{\begin{array}{c}\eth, \delta \vdash \ell\ \mathit{fresh} \qquad \eth \vdash \mathsf{z}\ \mathit{fresh} \qquad \delta \vdash \ell\ \mathit{fresh} \\ \eth_z = \eth \cup \{\ell \mapsto \mathsf{z}\} \qquad \delta' = \delta \cup \{\ell \mapsto \ell\} \\ \mathsf{e} = \mathsf{let}\ \mathsf{z} = \mathsf{new}\ \mathsf{42}\ \mathsf{in}\ \mathsf{delete}\ \mathsf{z}; \mathsf{return}\ \langle\langle\langle \mathbf{v} \rangle\rangle\rangle^{L_{ms}\to L_{tms}}\end{array}}{{}^{?}\langle\langle \mathbf{End}\ \mathbf{v} \rangle\rangle^{L_{ms}\to L_{tms}}_{\eth;\delta;[\cdot]} = \eth_z; \delta'; [\cdot]; \mathsf{e}}\ \text{(backtrans-end-ctxtocomp)}$$

Figure 51: Backtranslation of interaction-events from L_{ms} backtranslation-events to L_{tms} terms.

$$\frac{}{\vdash \varepsilon \text{ non-int-}\mathbf{a}}\ (\text{non-interface-}\varepsilon) \qquad \frac{}{\vdash \mathbf{Alloc}\ \ell\ \mathbf{n};\mathbf{t} \text{ non-int-}\mathbf{a}}\ (\text{non-interface-Alloc})$$

$$\frac{}{\vdash \mathbf{Dealloc}\ \ell;\mathbf{t} \text{ non-int-}\mathbf{a}}\ (\text{non-interface-Dealloc}) \qquad \frac{}{\vdash \mathbf{Get}\ \ell\ \mathbf{n};\mathbf{t} \text{ non-int-}\mathbf{a}}\ (\text{non-interface-Get})$$

$$\frac{}{\vdash \mathbf{Set}\ \ell\ \mathbf{n};\mathbf{t} \text{ non-int-}\mathbf{a}}\ (\text{non-interface-Set}) \qquad \frac{}{\vdash \lightning;\mathbf{t} \text{ non-int-}\mathbf{a}}\ (\text{non-interface-}\lightning) \qquad \frac{}{\vdash [\cdot] \text{ non-int-}\overline{\mathbf{a}}}\ (\text{non-interface-trace-empty})$$

$$\frac{\vdash \mathbf{a} \text{ non-int-}\overline{\mathbf{a}} \qquad \vdash \overline{\mathbf{a}} \text{ non-int-}\overline{\mathbf{a}}}{\vdash \mathbf{a}\cdot\overline{\mathbf{a}} \text{ non-int-}\overline{\mathbf{a}}}\ (\text{non-interface-trace-cons})$$

Figure 52: Non-Interfacing events.

$$\boxed{{}^{!?}\langle\!\langle \mathbf{a_1}\cdot\overline{\mathbf{a}}\cdot\mathbf{a_2}\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta;\overline{\ell}} = \eth';\delta';\overline{\ell'};\mathsf{e_1};\mathsf{e_2};\mathsf{e_3}}$$

„Given a `VarMap` $\eth$, δ, and $\overline{\ell}$, construct a `VarMap` $\eth'$, a δ', another stack of locations $\overline{\ell'}$, and an $\mathrm{L_{tms}}$ ensemble of expressions from target-trace $\overline{\mathbf{a}^{ms}}$ which starts with a ? event and ends in a !."

$$\frac{{}^{!}\langle\!\langle \mathbf{a_1}\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta;\overline{\ell}} = \overline{\ell'};\mathsf{e_1} \qquad \begin{matrix}\vdash \overline{\mathbf{a}} \text{ non-int-}\overline{\mathbf{a}} \\ \langle\!\langle \overline{\mathbf{a}}\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}} = \mathsf{e_2}\end{matrix} \qquad {}^{?}\langle\!\langle \mathbf{a_2}\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta;\overline{\ell'}} = \eth';\delta';\overline{\ell''};\mathsf{e_3}}{{}^{!?}\langle\!\langle \mathbf{a_1}\cdot\overline{\mathbf{a}}\cdot\mathbf{a_2}\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta;\overline{\ell}} = \eth';\delta';\overline{\ell''};\mathsf{e_1};\mathsf{e_2};\mathsf{e_3}}\ (\text{subtoplevel-backtrans})$$

Figure 53: Interaction-Trace-Based Backtranslation from $\mathrm{L_{ms}}$ backtranslation-events to $\mathrm{L_{tms}}$ terms.

$$\boxed{\langle\!\langle\!\langle \overline{\mathbf{a}^{ms}}\rangle\!\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta} = \eth';\delta;\Xi}$$

„Top-Level backtranslation of $\overline{\mathbf{a}}$ to $\mathrm{L_{tms}}$ library Ξ."

$$\frac{\begin{matrix}\overline{\mathbf{a}} = \overline{\mathbf{a_0}}\cdot\overline{\mathbf{a}^{comp}}\cdot\overline{\mathbf{a_1}} \qquad \vdash \overline{\mathbf{a}^{comp}} \text{ non-int-}\overline{\mathbf{a}} \\ \overline{\mathbf{a_0}} = \mathbf{Start}\cdot\overline{\mathbf{a}'_0}\cdot\mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_1} \qquad \overline{\mathbf{a_1}} = \mathbf{Ret}\ !\ \mathbf{v_2}\cdot\overline{\mathbf{a}'_0}\cdot\mathbf{End}\ \mathbf{v_3} \\ {}^{!?}\langle\!\langle \theta^*_{\bullet}\ (\overline{\mathbf{a_0}})\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = \eth;\delta;\overline{\ell};\mathsf{e_0};\mathsf{e}'_0;\mathsf{let}\ \mathsf{x_0} = \mathsf{new}\ \mathsf{n_0}\ \mathsf{in}\ \mathsf{e}''_0 \\ {}^{!?}\langle\!\langle \theta^*_{\bullet}\ (\overline{\mathbf{a_1}})\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\eth;\delta;\overline{\ell}} = \eth';\delta';[\cdot];\mathsf{e_1};\mathsf{e}'_1;\mathsf{e}''_1 \\ \mathsf{e} = \mathsf{e_0};\mathsf{e}'_0;\mathsf{let}\ \mathsf{x_0} = \mathsf{new}\ \mathsf{n_0}\ \mathsf{in}\ (\mathsf{e}''_0;\mathsf{e_1});\mathsf{e}'_1;\mathsf{e}''_1\end{matrix}}{\langle\!\langle\!\langle \overline{\mathbf{a}}\rangle\!\rangle\!\rangle^{\mathrm{L_{ms}}\to\mathrm{L_{tms}}}_{\emptyset;\emptyset} = \eth';\delta';\mathsf{let}\ \mathsf{main}\ \mathsf{x} : \mathbb{N}\to\mathbb{N} := \mathsf{e}, [\cdot]}\ (\text{backtrans-top-level})$$

Figure 54: Top-Level trace-based Backtranslation from $\mathrm{L_{ms}}$ trace $\overline{\mathbf{a}}$ to $\mathrm{L_{tms}}$ context Ξ.

4.4.3 Proofs and Auxiliary Lemmas

Lemma 57 ($\theta^*_{\bullet}$ distributive)**.**

(i) $\theta^*_{\bullet}\ (\overline{\mathbf{a_1}}\cdot\overline{\mathbf{a_2}}) = \theta^*_{\bullet}\ (\overline{\mathbf{a_1}})\cdot\theta^*_{\bullet}\ (\overline{\mathbf{a_2}})$

Proof. Induction on $\overline{\mathbf{a_1}}$. ☐

Lemma 58 ($\theta_{\bullet}$ invert concrete)**.** *If*

(i) $\mathbf{a} = \mathbf{a_b}; \mathbf{t}$

(ii) $\forall \ell\ \mathbf{n}\ \mathbf{m}, \mathbf{t} = \mathbf{ctx} \implies \mathbf{event_b} \notin \{\mathbf{Alloc}\ \ell\ \mathbf{n}, \mathbf{Dealloc}\ \ell, \mathbf{Get}\ \ell\ \mathbf{n}, \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{m}\}$

Then

(a) $\theta(\mathbf{a}) = \mathbf{a}$

Proof. Induction on $\overline{\mathbf{a_b}}$. □

Lemma 59 ($\xi; \Xi; [\cdot] \approx_{\delta;\mathsf{L}} \xi; \boldsymbol{\Xi}; [\cdot]$ holds)**.** *If*

(a) $\Xi \approx_{\delta;\mathsf{L}} \boldsymbol{\Xi}$

Then

(i) $\xi; \Xi; [\cdot] \approx_{\delta;\mathsf{L}} \xi; \boldsymbol{\Xi}; [\cdot]$

Proof. Easy. □

Lemma 60 (if $\ell \in \mathsf{L}$, then ρ doesn't matter)**.** *If*

(a) $\Psi = \mathsf{H}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

(b) $\Psi' = \mathsf{H}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho'; \mathsf{m}), \Delta_2$

(c) $\ell \in \mathsf{L}$

(d) $\boldsymbol{\Psi} \multimap_{\delta;\mathsf{L}} \Psi$

Then

(i) $\boldsymbol{\Psi} \multimap_{\delta;\mathsf{L}} \Psi'$

Proof. Easy. □

Lemma 61 ($\ell \in \operatorname{dom}\delta$ decomposes Δ)**.** *If*

(a) $\ell \in \operatorname{dom}\delta$

(b) $\mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta}$

Then $\exists \Delta_1\ \mathsf{x}\ \mathsf{t}\ \rho\ \mathsf{m}\ \Delta_2$,

(i) $\Delta = \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{m}), \Delta_2$

Proof. Easy. □

Lemma 62 (MS-Location Generator)**.** *If*

(a) *there is a set of locations* L

Then $\exists \delta_{MS}$

(i) *left-total, injective* $\delta_{MS} : \mathsf{L} \to \underline{L}$

Proof. This can be done by just generating fresh names $\underline{\ell}$ and assign them to each $\ell \in \mathsf{L}$. □

Lemma 63 (Filters Equal)**.** *If*

(a) $\forall \ell\ \ell, \delta(\ell) = \ell \implies \delta_{MS}(\ell) = \delta_{MS}(\delta^{-1}(\ell))$

(b) $\overline{a} \approx^{*}_{\delta;X} \overline{a}$

Then

(i) $\theta^{*}_{\delta_{MS}}(\overline{a}) = \theta^{*}_{\delta'_{MS}}(\overline{a})$

Proof. Induction on Assumption (b). □

Lemma 64 (Backtranslation for Non-Interacting Traces is Well-Typed)**.** *If*

(a) $\vdash \overline{a}$ non-int-$\overline{a}$

(b) $\langle\langle \theta^{*}_{\bullet}(\overline{a}) \rangle\rangle^{L_{ms} \to L_{tms}} = e$

Then

(i) $\vdash e : \mathbb{N}$

Proof. Note that $\theta^{*}_{\bullet}(\overline{a}) = [\cdot] \vee \theta^{*}_{\bullet}(\overline{a}) = \lightning$ because of Assumption (a). So, the only left cases are $e = 42$ and $e = \mathsf{abort}()$. Both terms easily typecheck under an empty gamma. □

Lemma 65 (Backtranslation for !-Interacting Events is Well-Typed)**.** *If*

(a) $^{!}\langle\langle \theta^{*}_{\bullet}(a) \rangle\rangle^{L_{ms} \to L_{tms}}_{\eth;\delta;\overline{\ell}} = e$

Then $\exists \Gamma$,

(i) $\Gamma \vdash e : \mathbb{N}$

Proof. Inverting Assumption (a) yields two cases, so we continue with them:

Case $a = \mathbf{Start}$**:**

Here, $e = 42$. Let $\Gamma = [\cdot]$, it's easy to see that $\vdash 42 : \mathbb{N}$ holds by Rule $t\!-\!\mathbb{N}$.

Case $a = \mathbf{Ret}\ !\ \mathbf{v}$**:**

$e = \mathsf{delete}\ z; \langle\langle\langle \mathbf{v} \rangle\rangle\rangle^{L_{ms} \to L_{tms}}$, where $\delta(\ell) = \ell$, $z = \eth(\ell)$ and ℓ is at the top of the stack of locations $\overline{\ell}$. Choose $\Gamma = z : \mathsf{ref}_1\ \mathbb{N}, [\cdot]$, the typing follows using Rule $t - \mathsf{delete}$.

□

Lemma 66 (Backtranslation for Call-? is Well-Typed)**.** *If*

(a) $^{?}\langle\langle \theta^{*}_{\bullet}(\mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v}) \rangle\rangle^{L_{ms} \to L_{tms}}_{\eth;\delta;\ell,\overline{\ell}} = \eth'; \delta; \ell, \overline{\ell'}; e'$

(b) $\delta(\ell) = \ell$

(c) $z = \eth'(\ell)$

(d) $e = e';\ \mathsf{delete}\ z; \langle\langle\langle \mathbf{v}' \rangle\rangle\rangle^{L_{ms} \to L_{tms}}$

(e) $\Gamma = \Gamma_1, \mathsf{foo} : \mathbb{N} \to \mathbb{N}, \Gamma_2$

Then,

(i) $\Gamma \vdash \mathsf{e} : \mathbb{N}$

Proof. Invert Assumption (a) giving $\mathsf{e} = \mathsf{let\ z = new\ 42\ in\ call}\ \langle\!\langle\!\langle \mathbf{foo} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}};\ \mathsf{delete\ z};\langle\!\langle\!\langle \mathbf{v'} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}$. The claim follows using Rules $t - \mathsf{new}$ and $t - \mathsf{delete}$ and Rules $t - \mathsf{let}$ and $t - \mathsf{call}$, where the latter uses Assumption (e). □

Lemma 67 (Backtranslation for End-? is Well-Typed). *If*

(a) ${}^{?}\langle\!\langle \theta^{*}_{\bullet}\ (\mathbf{End\ v}) \rangle\!\rangle^{\mathrm{L}_{ms} \to \mathrm{L}_{tms}}_{\eth;\delta;[\cdot]} = \eth'; \delta'; [\cdot]; \mathsf{e}$

Then,

(i) $\vdash \mathsf{e} : \mathbb{N} \to \bot$

Proof. Invert Assumption (a) giving $\mathsf{e} = \mathsf{let\ z = new\ 42\ in\ delete\ z;\ return}\ \langle\!\langle \mathbf{v} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\eth}$, where $\eth \vdash \mathsf{z}$ *fresh*. The claim follows using Rules $t - \mathsf{new}$, $t - \mathsf{delete}$ and $t - \mathsf{return}$. □

Lemma 68 (Backtranslation is Well-Typed). *If*

(a) $\langle\!\langle\!\langle \overline{\mathbf{a}} \rangle\!\rangle\!\rangle^{\mathrm{L}_{ms} \to \mathrm{L}_{tms}}_{\emptyset;\emptyset} = \eth; \delta; \Xi_{\mathsf{ctx}}$

(b) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

(c) $\Gamma \vdash \Xi_{\mathsf{comp}}$ ok

Then,

(i) $\Gamma \vdash \Xi$ ok

Proof. First up, note that $\Gamma \vdash \Xi_{\mathsf{comp}}$ ok, so it suffices to check $\Gamma \vdash \Xi_{\mathsf{ctx}}$ ok by Lemma 23 (Typed Linking Recomposition). By inverting Assumption (a) we get $\overline{\mathbf{a}} = \overline{\mathbf{a_0}} \cdot \overline{\mathbf{a^{comp}}} \cdot \overline{\mathbf{a_1}}$ and $\mathsf{e} = \mathsf{e_0}; \mathsf{e'_0}; \mathsf{let\ x_0 = new\ n_0\ in}\ (\mathsf{e''_0}; \mathsf{e_1}); \mathsf{e'_1}; \mathsf{e''_1}$, where:

(H_1) $\vdash \overline{\mathbf{a^{comp}}}$ non-int-$\overline{\mathbf{a}}$

(H_2) $\langle\!\langle \theta^{*}_{\bullet}\ (\overline{\mathbf{a_0}}) \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = \eth'; \delta'; \overline{\ell}; \mathsf{e_0}; \mathsf{e'_0}; \mathsf{let\ x_0 = new\ n_0\ in\ e''_0}$

(H_3) $\langle\!\langle \theta^{*}_{\bullet}\ (\overline{\mathbf{a_1}}) \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\eth';\delta';\overline{\ell}} = \eth''; \delta''; [\cdot]; \mathsf{e_1}; \mathsf{e'_1}; \mathsf{e''_1}$.

Inverting Assumptions (H_2) and (H_3), first take note that $\overline{\mathbf{a_0}} = \mathbf{Start} \cdot \overline{\mathbf{a'_0}} \cdot \mathbf{Call\ ?\ foo\ v_2}$ and $\overline{\mathbf{a_1}} = \mathbf{Ret\ ?\ v_3} \cdot \overline{\mathbf{a'_1}} \cdot \mathbf{End\ v_4}$. Furthermore, $\mathsf{e_0} = 42$, $\langle\!\langle \overline{\mathbf{a'_0}} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} = \mathsf{e'_0}$, and $\mathsf{e''_0} = \mathsf{let\ z = new\ 42\ in\ call\ foo\ n}$, where $\mathsf{z} = \eth'(\ell)$ and $\delta(\ell) = \ell$, $\mathsf{n} = \langle\!\langle\!\langle \mathbf{v_2} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}$, and $[\![\mathsf{foo}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{foo}$ with $\mathsf{foo} \in$ dom Ξ_{comp}. Also, $\mathsf{e_1} = \mathsf{delete\ z}$, $\langle\!\langle \overline{\mathbf{a'_1}} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} = \mathsf{e'_1}$, and $\mathsf{e''_1} = \mathsf{let\ x = new\ 42\ in\ delete\ x; return\ m}$, where $\mathsf{m} = \langle\!\langle\!\langle \mathbf{v_4} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}$ and $\eth'' \vdash \mathsf{x}$ *fresh*. Therefore:

$$\mathsf{e} = 42; \mathsf{e'_0}; \mathsf{let\ z = new\ 42\ in}\ (\mathsf{call\ foo\ n; delete\ z}); \mathsf{e'_1}; \mathsf{let\ x = new\ 42\ in\ delete\ x; return\ m}$$

This typechecks easily, making use of Rules $t - \mathsf{new}$ and $t - \mathsf{delete}$ and Rules $t - \mathsf{let}$, $t - \mathsf{call}$ and $t - \mathsf{return}$, as well as Lemma 64 (Backtranslation for Non-Interacting Traces is Well-Typed). □

Lemma 69 (Backtranslation Correctness of **Start**). *If*

(a) $\Xi = \Xi_{\mathrm{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}$

(b) $\xi = [\![\xi]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}} = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}$

(c) $\xi; \Xi; [\cdot]; \mathbf{comp}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathbf{call\ main\ 0} \xrightarrow{\mathbf{Start};\mathbf{comp}}_{ctx} \xi; \Xi; ([\cdot]; \mathbf{main}), [\cdot]; \mathbf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathbf{e}_{\mathbf{main}}[\mathbf{0}/\mathbf{x}]$

(d) $\langle\!\langle \theta_{\bullet}\ (\mathbf{Start}) \rangle\!\rangle^{\mathrm{L}_{ms}\to\mathrm{L}_{tms}}_{\emptyset;\emptyset;[\cdot]} = [\cdot]; 42$

(e) $\xi; \Xi; [\cdot]; \mathsf{comp}; [\cdot]; [\cdot]; [\cdot] \approx_{\emptyset;[\cdot]} \xi; \Xi; [\cdot]; \mathbf{comp}; [\cdot]; [\cdot]; [\cdot]$

(f) $\Xi = \{\mathsf{main} \mapsto 42; \mathsf{e}_{\mathsf{main}}\} \bowtie \Xi_{\mathsf{comp}}$

Then $\exists n$

(i) $\xi; \Xi; [\cdot]; \mathsf{comp}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\mathsf{Start};\mathsf{comp}}_{ctx}{}^{n} \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{e}_{\mathsf{main}}[0/\mathsf{x}]$

(ii) $\xi; \Xi; ([\cdot]; \mathbf{main}), [\cdot]; \mathbf{ctx}; [\cdot]; [\cdot]; [\cdot] \multimap_{\emptyset;[\cdot]} \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]$

(iii) $\mathsf{Start} \approxeq^{*}_{\emptyset;[\cdot]} \mathbf{Start}$

Proof. Note using Rule $e-$ ctx $-$ call$-$main we can easily conclude:

(H_1) $\xi; \Xi; [\cdot]; \mathsf{comp}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\mathsf{Start};\mathsf{comp}}{}^{1}_{\mathrm{ctx}} \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{let}\ \_ = 42\ \mathsf{in}\ \mathsf{e}_{\mathsf{main}}[0/\mathsf{x}]$

On Assumption (H_1) apply Rules $e-$ let $-$ f and $e-$ ctx

(H_2) $\xi; \Xi; ([\cdot], \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{let}\ \_ = 42\ \mathsf{in}\ \mathsf{e}_{\mathsf{main}}[0/\mathsf{x}] \xrightarrow{\varepsilon}{}^{1}_{\mathrm{ctx}} \xi; \Xi; ([\cdot], \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{e}_{\mathsf{main}}[0/\mathsf{x}]$

Use Assumptions (H_1) and (H_2) on Rules $en-$ refl and $en-$ trans$-$important:

(H_3) $\xi; \Xi; [\cdot]; \mathsf{comp}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{call\ main\ 0} \xrightarrow{\mathsf{Start}}{}^{2}_{\mathrm{ctx}} \xi; \Xi; ([\cdot], \mathsf{main}), [\cdot]; \mathsf{ctx}; [\cdot]; [\cdot]; [\cdot] \triangleright \mathsf{e}_{\mathsf{main}}[0/\mathsf{x}]$

So, instantiating $n = 2$ concludes Goal (i).

Goal (ii) by Rule state-qe, Rules memstate-qe and cfstate-qe and Rules empty-memstate-eq, empty-commlib-lib-eq and empty-kontstack-eq. Goal (iii) follow Rule start-event-eq. ☐

Lemma 70 (Backtranslation Correctness of **Ret**). *If*

(a) $\Omega = \xi; \Xi; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}; \mathbf{comp}; \Psi$

(b) $\Xi = \Xi_{\mathrm{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}$

(c) $\xi = [\![\xi]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}} = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}$

(d) $\Omega \triangleright \mathbf{K}_{\mathbf{component}}[\mathbf{return\ v}] \xrightarrow{\mathbf{Ret\ !\ v};\mathbf{comp}}_{ctx} \Omega' \triangleright \mathbf{K}[\mathbf{v}]$

(e) $\mathbf{K}_{\mathbf{component}} = [\![\mathsf{K}_{\mathsf{component}}]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}$

(f) $\langle\!\langle \theta_{\bullet}\ (\mathbf{Ret\ !\ v}) \rangle\!\rangle^{\mathrm{L}_{ms}\to\mathrm{L}_{tms}}_{\eth;\delta;\overline{\ell}} = \overline{\ell'}; \mathsf{delete\ z}; \langle\!\langle \mathbf{v} \rangle\!\rangle^{\mathrm{L}_{ms}\to\mathrm{L}_{tms}}$

(g) $\Omega \approx_{\delta;\mathsf{L}} \Omega$

(h) $\Omega = \xi; \Xi; (\mathsf{K}, \mathsf{foo}), \overline{\mathsf{K}}; \mathsf{comp}; \Psi$

(i) $\mathsf{K} = [\cdot]\,; \mathsf{delete}\ \mathsf{z}; \mathsf{K}[\langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}]$

(j) $\Xi = \{\mathsf{main} \mapsto \mathsf{e}_{\mathsf{main}}\} \bowtie \Xi_{\mathsf{comp}}$

(k) $\delta(\ell) = \ell$

(l) $\mathsf{z} = \eth(\ell)$

(m) $\ell \in \mathsf{L}$

(n) $\mathbf{foo} \in \mathbf{\Xi_{comp}}$

then $\exists n\ \bar{\mathsf{a}}\ \Psi'$,

(i) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Dealloc}\ \ell\}$

(ii) $\Omega \triangleright \mathsf{K}_{\mathsf{component}}[\mathsf{return}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}] \xrightarrow{\bar{\mathsf{a}}}_{ctx} n\ \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}]$

(iii) $\mathbf{\Omega}' \multimap_{\delta;\mathsf{L}} \xi; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi'$

(iv) $\bar{\mathsf{a}} \approxeq^{*}_{\delta;\mathsf{X}'} \mathbf{Ret\ !\ v}$

Proof. From Assumption (k) it is clear that:

(H_1) $\ell \in \operatorname{dom} \delta$

Invert Assumption (g):

(H_2) $\Omega = \Phi; \mathsf{comp}; \Psi$

(H_3) $\mathbf{\Omega} = \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}$

(H_4) $\Phi \approx \mathbf{\Phi}$

(H_5) $\Psi \approx_{\delta;\mathsf{L}} \mathbf{\Psi}$

Decompose Ψ into $\mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta$. Make use of Assumptions (H_1) and (H_5) with Lemma 61 ($\ell \in \operatorname{dom} \delta$ decomposes Δ):

(H_6) $\Delta = \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}'; \rho; \mathsf{m}), \Delta_2$

For the goals, instantiate the existentials with:

(E_1) $n = 3$

(E_2) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Dealloc}\ \ell\}$

(E_3) $\bar{\mathsf{a}} = \mathsf{Dealloc}\ \ell \cdot \mathsf{Ret}\ !\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L_{ms}} \to \mathsf{L_{tms}}}_{\eth}$

(E_4) $\Psi' = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}'; ☣; \mathsf{m}), \Delta_2$

Goal (i) is easy. Let $\mathsf{n} = \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L_{ms}} \to \mathsf{L_{tms}}}$. For Goal (ii), first note Assumption (h), use Rule $e - \mathrm{ctx} - \mathsf{return} - \mathrm{notsame}$ and rewrite K using Assumption (i):

(H_7) $\xi; \Xi; (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}}; \mathsf{comp}; \Psi \triangleright \mathsf{K}_{\mathsf{component}}\,[\mathsf{return}\ \mathsf{n}] \xrightarrow{\mathsf{Ret}\ !\ \mathsf{n}}_{\mathrm{ctx}} \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi \triangleright \mathsf{n}; \mathsf{delete}\ \mathsf{z}; \mathsf{K}[\mathsf{n}]$

This works, because of Assumption (n), then $\rho(\mathsf{comp}) = !$. Note, Assumption (H_7), continue with Rules $e - \mathsf{let} - \mathsf{f}$ and $e - \mathrm{ctx}$:

(H_8) $\xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi \triangleright \mathsf{K}_{\mathsf{component}}[\mathsf{return\ n}] \xrightarrow{\varepsilon}_{\mathrm{ctx}} \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi \triangleright \mathsf{delete\ z}; \mathsf{K}[\mathsf{n}]$

Finally, use Rules $e - \mathsf{delete}$ and $e - \mathrm{ctx}$ on Assumption (H_8):

(H_9) $\xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi \triangleright \mathsf{delete\ z}; \mathsf{K}[\mathsf{n}] \xrightarrow{\mathsf{Dealloc}\ \ell}_{\mathrm{ctx}} \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{n}]$

Use Assumptions (H_7) to (H_9) on Rule $en -$ trans−important, Rules $en -$ refl and $en -$ trans−unimportant to get exactly what Goal (ii) wants us to prove.

With Assumption (h), invert Assumption (g):

(H_{10}) $\boldsymbol{\Omega} = \boldsymbol{\Phi}; \mathbf{comp}; \boldsymbol{\Psi}$

(H_{11}) $\boldsymbol{\Phi} = \xi; \boldsymbol{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}$

(H_{12}) $\Phi \approx \boldsymbol{\Phi}$

(H_{13}) $\Psi \approx_{\delta;\mathsf{L}} \boldsymbol{\Psi}$

Invert Assumption (d) to extract:

(H_{14}) $\boldsymbol{\Omega}' = \boldsymbol{\Phi}'; \mathbf{ctx}; \boldsymbol{\Psi}$

(H_{15}) $\boldsymbol{\Phi}' = \xi; \boldsymbol{\Xi}; \overline{\mathbf{K}}$

Let $\Phi' = \xi; \Xi; \overline{\mathsf{K}}$.

Use Rule state-qe on Goal (iii) giving goals:

(v) $\Phi' \approx \boldsymbol{\Phi}'$

(vi) $\Psi' \approx_{\delta;\mathsf{L}'} \boldsymbol{\Psi}$

Goal (v) follows by Assumption (H_{12}).

Note that $\Psi' = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}'; \cancel{\otimes}; \mathsf{m}), \Delta_2$ and remember that $\Psi = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta_1, \mathsf{x} \mapsto (\ell; \mathsf{t}'; \rho; \mathsf{m}), \Delta_2$, of which we know Assumption (H_{13}). So, Goal (vi) follows from Lemma 60 (if $\ell \in \mathsf{L}$, then ρ doesn't matter), which needs Assumption (m). Now Goal (iv): First apply Rule ignore-cons-trace-eq, given that $\mathsf{Dealloc}\ \ell \in \mathsf{X}'$. What is left to show is:

(vii) $\mathsf{Ret}\ !\ \mathsf{v} \approxeq^{*}_{\delta;\mathsf{X}'} \mathbf{Ret}\ !\ \mathbf{v}$

Goal (vii) follows by Rules empty-trace-eq, cons-trace-eq and ret-event-eq. □

Lemma 71 (Middle of Backtranslation Correctness)**.** *If*

(a) $\boldsymbol{\Omega} = \xi; \boldsymbol{\Xi}; \overline{\mathbf{K}}; \mathbf{ctx}; \boldsymbol{\Psi}$

(b) $\boldsymbol{\Xi} = \boldsymbol{\Xi}_{\mathbf{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathsf{L}_{ms}}$

(c) $\xi = [\![\xi]\!]^{\mathsf{L}_{tms} \to \mathsf{L}_{ms}} = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathsf{L}_{ms}}$

(d) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\overline{\mathbf{a}}}_{ctx}{}^{n}\ \boldsymbol{\Omega}' \triangleright \mathbf{e}'\gamma'$

(e) $\langle\!\langle \theta^{*}_{\bullet}\ (\overline{\mathbf{a}}) \rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}} = \mathsf{e}$

(f) $\vdash \overline{\mathbf{a}}$ non-int- $\overline{\mathbf{a}}$

(g) $\mathbf{\Omega} \multimap_{\delta;\mathsf{L}} \Omega$

(h) $\Omega = \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi$

(i) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

then $\exists n'\ \overline{\mathsf{a}}$,

(i) $\Omega \triangleright \mathsf{K}[\mathsf{e}] \xrightarrow{\overline{\mathsf{a}}}_{ctx}{}^{n'} \xi; \Xi; \overline{\mathsf{K}}; \Psi' \triangleright \mathsf{K}[42]$

(ii) $\mathbf{\Omega}' \multimap_{\delta;\mathsf{L}} \xi; \Xi; \overline{\mathsf{K}}; \Psi'$

(iii) $\overline{\mathsf{a}} \approxeq^{*}_{\delta;\emptyset} [\cdot]$

Proof. Induction on Assumption (f).

Case $\overline{\mathbf{a}} = [\cdot]$

Invert Assumption (e) to obtain:

(H_1) $\mathsf{e} = 42$

$\mathsf{e} = \mathsf{abort}()$ is not an option, since it contradicts Assumption (d).

Instantiate our goals with:

(H_2) $n = 0$

(H_3) $\overline{\mathsf{a}} = [\cdot]$

Goal (i) follows by reflexivity.

Use Assumption (g) to solve Goal (ii).

For Goal (iii), use Rule empty-trace-eq.

Case $\overline{\mathbf{a}} = \mathbf{a} \cdot \overline{\mathbf{a}}$**:**

Regardless of the shape of $\mathbf{a}$ (noting that it cannot be $\lightning$) note that $\langle\!\langle \mathbf{a} \cdot \overline{\mathbf{a}} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} = \langle\!\langle \overline{\mathbf{a}} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}$. Thus, all cases follow immediately by the induction hypothesis.

$\square$

Lemma 72 (Backtranslation Correctness of $\mathbf{Call}\ ?$)**.** *If*

(a) $\mathbf{\Omega} = \xi; \mathbf{\Xi}; \overline{\mathbf{K}}; \mathbf{ctx}; \mathbf{\Psi}$

(b) $\overline{\mathbf{K}} = (\mathbf{K}'; \mathbf{bar}), \overline{\mathbf{K}}'$

(c) $\mathbf{\Xi} = \mathbf{\Xi_{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathrm{L}_{ms}}$

(d) $\xi = [\![\xi]\!]^{\mathsf{L}_{tms} \to \mathrm{L}_{ms}} = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathrm{L}_{ms}}$

(e) $\mathbf{\Omega} \triangleright \mathbf{K}\,[\mathbf{call\ foo\ v}] \xrightarrow{\mathbf{Call\ ?\ foo\ v;ctx}}_{ctx} \xi; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}; \mathbf{comp}; \mathbf{\Psi} \triangleright [\![\mathsf{e}_{\mathsf{foo}}]\!]^{\mathsf{L}_{tms} \to \mathrm{L}_{ms}}[\mathbf{v}/\mathbf{y}]$

(f) ${}^{?}\langle\!\langle \theta_{\bullet}\ (\mathbf{Call\ ?\ foo\ v}) \rangle\!\rangle^{\mathrm{L}_{ms} \to \mathsf{L}_{tms}}_{\mho;\delta;\overline{\ell}} = \mho'; \delta'; \overline{\ell'}; \mathsf{let\ z = new\ 42\ in\ call\ foo}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathrm{L}_{ms} \to \mathsf{L}_{tms}}$

(g) $\mathbf{\Omega} \multimap_{\delta;\mathsf{L}} \Omega$

(h) $\Omega = \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi$

(i) $\mathsf{let\ foo\ y} : \tau_\lambda := \mathsf{e_{foo}} \in \Xi_{\mathsf{comp}}$

(j) $\eth, \delta \vdash \ell$ *fresh*

(k) $\eth \vdash \mathsf{z}$ *fresh*

(l) $\delta \vdash \ell$ *fresh*

(m) $\delta'(\ell) = \ell$

(n) $\eth'(\ell) = \mathsf{z}$

(o) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

then $\exists n\ \bar{\mathsf{a}}\ \Psi'$,

(i) $\mathsf{L}' = \mathsf{L} \cup \{\ell\}$

(ii) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Alloc}\ \ell\ 42\}$

(iii) $\Omega \triangleright \mathsf{K}[\mathsf{let\ z = new\ 42\ in\ call\ foo}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathrm{L}_{ms} \to \mathrm{L}_{tms}}] \xrightarrow{\bar{\mathsf{a}}}_{ctx}{}^{n}\ \xi; \Xi; (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}}; \mathsf{comp}; \Psi' \triangleright \mathsf{e_{foo}}\gamma$

(iv) $\xi; \Xi; (\mathsf{K}, \mathsf{foo}), \overline{\mathsf{K}}; \mathsf{comp}; \Psi' \approx_{\delta'; \mathsf{L}'} \xi; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}; \mathbf{comp}; \mathbf{\Psi}$

(v) $\bar{\mathsf{a}} \mathrel{\approx\!\!\approx}^{*}_{\delta'; \mathsf{X}'} \mathbf{Call}\ ?\ \mathbf{foo\ v; ctx}$

Proof. First, we unfold the source memory state:

(H_1) $\Psi = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta$

Let $\mathsf{v} = \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}$. Next, we instantiate the existentials in the goals as follows:

(H_6) $n = 2$

(H_7) $\mathsf{L}' = \mathsf{L} \cup \{\ell\}$

(H_8) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Alloc}\ \ell\ 42\}$

(H_9) $\bar{\mathsf{a}} = \mathsf{Alloc}\ \ell\ 42; \mathsf{ctx} \cdot \mathsf{Call}\ ?\ \mathsf{foo\ v; ctx}$

(H_{10}) $\Psi' = \underbrace{0, \cdots, 0}_{42-times}, \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \mathsf{z} \mapsto (\ell; \mathsf{ctx}; \square; 42), \Delta$

(H_{11}) $\gamma = [\langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} / \mathsf{y}]$

Goals (i) and (ii) follow easily. For Goal (iii), first use Rules $e-$new and $e-$ctx, noting Assumptions (j) and (k):

(H_{11}) $\Omega \triangleright \mathsf{K}[\mathsf{let\ z = new\ 42\ in\ call\ foo\ v}] \xrightarrow{\mathsf{Alloc}\ \ell\ 42; \mathsf{ctx}}_{\mathsf{ctx}}{}^{1}\ \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{call\ foo\ v}]$

Now use Rule $e-$ctx$-$return$-$notsame:

(H_{12}) $\xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{call\ foo\ v}] \xrightarrow{\mathsf{Call}\ ?\ \mathsf{foo\ z; ctx}}_{\mathsf{ctx}}{}^{1}\ \xi; \Xi; (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{e_{foo}}[\mathsf{v}/\mathsf{y}]$

Use Assumptions (H_{11}) and (H_{12}) on Rules $en-$refl and $en-$trans$-$important to get exactly what Goal (iii) wants us to prove.

Invert Assumption (g):

(H_{12}) $\Omega = \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi$

(H_{13}) $\boldsymbol{\Omega} = \xi; \boldsymbol{\Xi}; \overline{\mathbf{K}}; \mathbf{ctx}; \boldsymbol{\Psi}$

(H_{14}) $\boldsymbol{\Phi} \multimap \Phi$

(H_{15}) $\boldsymbol{\Psi} \multimap_{\delta;\mathsf{L}} \Psi$

Invert Assumption (H_{14}):

(H_{16}) $\Xi \approx \boldsymbol{\Xi}$

(H_{17}) $\overline{\mathbf{K}} \multimap \overline{\mathsf{K}}$

Invert Assumption (H_{15})

(H_{18}) $\boldsymbol{\Psi} = \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta}$

(H_{19}) $\mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta}$

Apply Rule state-eq on Goal (iv):

(v) $\underbrace{0, \ldots, 0}_{42-\mathsf{times}}, \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \mathsf{z} \mapsto (\ell; \mathsf{ctx}; \square; 42), \Delta \approx_{\delta';\mathsf{L}'} \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta}$

(vi) $\xi; \Xi; (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}} \approx \xi; \boldsymbol{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}$

Use Rule whatever-cons-memstate-eq on Goal (v), since $\ell \in \mathsf{L}'$:

(vii) $\mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \approx_{\delta';\mathsf{L}'} \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \boldsymbol{\Delta}$

Goal (vii) can be resolved by Assumption (H_{19}).
Apply Rule cfstate-eq on Goal (vi):

(ix) $\Xi \approx \boldsymbol{\Xi}$

(x) $(\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}} \approx (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}$

Goal (ix) is resolved by Assumption (H_{16}).
Apply Rule cons-kontstack-eq to Goal (x), leaving us with:

(xi) $\overline{\mathsf{K}} \approx \overline{\mathbf{K}}$

Invert Assumption (H_{17}):

(H_{20}) $\overline{\mathsf{K}} \approx \overline{\mathbf{K}}$

Assumption (H_{20}) can now be used to solve Goal (xi).
Lastly, apply Rule ignore-cons-trace-eq on Goal (v), given that $\mathsf{Alloc}\ \ell\ 42 \in \mathsf{X}'$.
What is left to show is:

(vii) $\mathsf{Call}\ ?\ \mathsf{foo}\ \mathsf{v} \approxeq^{*}_{\delta;\mathsf{X}'} \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v}$

Goal (vii) follows by Rules empty-trace-eq, cons-trace-eq and call-event-eq. ☐

Lemma 73 (Backtranslation Correctness of **End**)**.** *If*

(a) $\boldsymbol{\Xi} = \boldsymbol{\Xi}_{\mathbf{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathbf{L}_{ms}}$

(b) $\xi = [\![\xi]\!]^{\mathsf{L}_{tms} \to \mathbf{L}_{ms}} = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathsf{L}_{tms} \to \mathbf{L}_{ms}}$

(c) $\xi; \Xi; ([\cdot]; \mathbf{main}), [\cdot]; \mathbf{ctx}; \Psi \triangleright \mathbf{K}[\mathbf{return\ n}] \xrightarrow{\mathbf{End\ n;ctx}}_{ctx} \xi; \Xi; [\cdot]; \mathbf{comp}; \Psi \triangleright \mathbf{n}$

(d) $^{?}\langle\!\langle \theta_{\bullet}\ (\mathbf{End\ n}) \rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}_{\eth;\delta;[\cdot]} = \eth'; \delta'; [\cdot]; \mathsf{let\ z = new\ 42\ in\ delete\ z; return}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}$

(e) $\xi; \Xi; ([\cdot]; \mathbf{main}), [\cdot]; \mathbf{ctx}; \Psi \multimap_{\delta;\mathsf{L}} \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]; \mathsf{ctx}; \Psi$

(f) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

(g) $\eth, \delta \vdash \ell$ *fresh*

(h) $\eth \vdash \mathsf{z}$ *fresh*

(i) $\delta \vdash \ell$ *fresh*

(j) $\delta'(\ell) = \ell$

(k) $\eth'(\ell) = \mathsf{z}$

then $\exists n\ \bar{\mathsf{a}}\ \Psi'$,

(i) $\mathsf{L}' = \mathsf{L} \cup \{\ell\}$

(ii) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Alloc}\ \ell\ 42, \mathsf{Dealloc}\ \ell\}$

(iii) $\Omega \triangleright \mathsf{K}[\mathsf{let\ z = new\ 42\ in\ delete\ z; return}\ \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}] \xrightarrow{\bar{\mathsf{a}}}_{ctx}{}^{n} \xi; \Xi; [\cdot]; \mathsf{comp}; \Psi' \triangleright \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}$

(iv) $\xi; \Xi; [\cdot]; \mathsf{comp}; \Psi' \approx_{\delta';\mathsf{L}'} \xi; \Xi; [\cdot]; \mathbf{comp}; \Psi$

(v) $\bar{\mathsf{a}} \approxeq^{*}_{\delta';\mathsf{X}'} \mathbf{End\ v; ctx}$

Proof. First, we unfold the source memory state:

(H_1) $\Psi = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta$

Let $\mathsf{v} = \langle\!\langle\!\langle \mathbf{v} \rangle\!\rangle\!\rangle^{\mathsf{L}_{ms} \to \mathsf{L}_{tms}}$. Next, we instantiate the existentials in the goals as follows:

(H_6) $n = 3$

(H_7) $\mathsf{L}' = \mathsf{L} \cup \{\ell\}$

(H_8) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Alloc}\ \ell\ 42, \mathsf{Dealloc}\ \ell\}$

(H_9) $\bar{\mathsf{a}} = \mathsf{Alloc}\ \ell\ 42; \mathsf{ctx} \cdot \mathsf{Dealloc}\ \ell; \mathsf{ctx} \cdot \mathsf{End\ v}; \mathsf{ctx}$

(H_{10}) $\Psi' = \underbrace{0, \cdots, 0}_{42-times}, \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \mathsf{z} \mapsto (\ell; \mathsf{ctx}; ☣; 42), \Delta$

Let $\Psi_{\square} = \underbrace{0, \cdots, 0}_{42-times}, \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \mathsf{z} \mapsto (\ell; \mathsf{ctx}; \square; 42), \Delta$.

Let $\overline{\mathsf{K}} = ([\cdot]; \mathsf{main}), [\cdot]$. Goals (i) and (ii) follow easily. For Goal (iii), first use Rules $e-\mathsf{new}$ and $e-\mathrm{ctx}$, noting Assumptions (g) and (h):

(H_{10}) $\Omega \triangleright \mathsf{K}[\mathsf{let\ z = new\ 42\ in\ delete\ z; return\ v}] \xrightarrow{\mathsf{Alloc}\ \ell\ 42;\mathsf{ctx}}_{\mathsf{ctx}} \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx};\ \Psi_{\square} \triangleright \mathsf{K}[\mathsf{delete\ z; return\ v}]$

Next, use Rules $e{-}\mathsf{let}{-}\mathsf{f}$, $e{-}\mathsf{delete}$, $e{-}\mathrm{ctx}$, $en{-}\mathrm{refl}$ and $en{-}\mathrm{trans}{-}\mathrm{unimportant}$ on Assumption (H_{10}):

(H_{11}) $\xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi_{\square} \triangleright \mathsf{K}[\mathsf{let}\ \_ = \mathsf{delete\ z\ in\ return\ v}] \xrightarrow{\mathsf{Dealloc}\ \ell;\mathsf{ctx}}_{\mathrm{ctx}}{}^{2}\ \xi; \Xi; \overline{\mathsf{K}}; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{return\ v}]$

Now, apply Rules $e - \mathrm{ctx}$ and $e - \mathrm{ctx} - \mathsf{return}{-}\mathrm{main}$ on Assumption (H_{11}).

(H_{12}) $\xi; \Xi; ([\cdot], \mathsf{main}), [\cdot]; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{return\ v}] \xrightarrow{\mathsf{End\ v;ctx}}_{\mathrm{ctx}} \xi; \Xi; [\cdot]; \mathsf{comp}; \Psi' \triangleright \mathsf{v}$

Use Assumptions (H_{10}) to (H_{12}) on Rules $en{-}\mathrm{refl}$ and $en{-}\mathrm{trans}{-}\mathrm{important}$ to get exactly what Goal (iii) wants us to prove.

Invert Assumption (e):

(H_{13}) $\Omega = \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]; \mathsf{ctx}; \Psi$

(H_{14}) $\mathbf{\Omega} = \xi; \mathbf{\Xi}; ([\cdot]; \mathbf{main}), [\cdot]; \mathbf{ctx}; \mathbf{\Psi}$

(H_{15}) $\xi; \mathbf{\Xi}; ([\cdot]; \mathbf{main}), [\cdot] \multimap \xi; \Xi; ([\cdot]; \mathsf{main}), [\cdot]$

(H_{16}) $\mathbf{\Psi} \multimap_{\delta;\mathsf{L}} \Psi_0$

Invert Assumption (H_{15}):

(H_{16}) $\Xi \approx \mathbf{\Xi}$

(H_{17}) $([\cdot]; \mathbf{main}), [\cdot] \multimap_{\xi} ([\cdot]; \mathsf{main}), [\cdot]$

Invert Assumption (H_{16})

(H_{19}) $\Psi_0 = \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta$

(H_{20}) $\mathbf{\Psi} = \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta}$

(H_{21}) $\mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \approx_{\delta;\mathsf{L}} \mathbf{H}^{\mathbf{ctx}}; \mathbf{H}^{\mathbf{comp}}; \mathbf{\Delta}$

Use Assumptions (H_{16}) and (H_{21}) and rule empty-kontstack-eq to solve Goal (iv). Lastly, apply Rule ignore-cons-trace-eq on Goal (v), given that $\{\mathsf{Alloc}\ \ell\ 42, \mathsf{Dealloc}\ \ell\} \subseteq \mathsf{X}'$. What is left to show is:

(vii) $\mathsf{End\ v} \approxeq^{*}_{\delta;\mathsf{X}'} \mathbf{End\ v}$

Goal (vii) follows by Rules empty-trace-eq, cons-trace-eq and call-event-eq. $\square$

Lemma 74 (Backtranslation Correctness)**.** *If*

(a) $\mathbf{\Omega} \triangleright \mathbf{call\ main\ 0} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{ctx}\ \mathbf{\Omega}' \triangleright \mathbf{f}_{\lightning}$

(b) $\langle\langle\!\langle \overline{\mathbf{a}} \rangle\!\rangle\rangle^{\mathbf{L}_{ms} \to \mathsf{L}_{tms}}_{\emptyset;\emptyset} = \eth; \delta; \Xi_{\mathsf{ctx}}$

(c) $\Omega \approx_{\emptyset;\emptyset} \mathbf{\Omega}$

(d) $\mathbf{\Omega} = \mathbf{foo}, [\cdot]; \mathbf{\Xi}; [\cdot]; \mathbf{comp}; [\cdot]; [\cdot]; [\cdot]$

(e) $\Omega = \xi; \Xi; \overline{\mathsf{K}}; \mathsf{comp}; \Psi$

(f) $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$

(g) $\Gamma \vdash \Xi_{comp}$ ok

then $\exists \delta_e\ \mathsf{L_e}\ \mathsf{X_e}\ \Omega'_e\ \bar{\mathsf{a}}_e$,

(i) $\Omega \triangleright \mathsf{call\ main\ 0} \xrightarrow{\bar{a}_e}{}^{*}_{ctx} \Omega'_e \triangleright \langle\langle\langle \mathbf{f}_{\not{\iota}} \rangle\rangle\rangle^{\mathbf{L}_{ms}\to\mathsf{L}_{tms}}$

(ii) $\Omega'_e \approx_{\delta_e;\mathsf{L_e}} \mathbf{\Omega'}$

(iii) $\bar{\mathsf{a}}_e \approxeq^{*}_{\delta_e;\mathsf{X_e}} \bar{\mathbf{a}}$

Proof. Inverting Assumption (b) gives:

(H_1) $\theta^*_{\bullet}(\bar{\mathbf{a}}) = \mathbf{Start};\mathbf{comp} \cdot \overline{\mathbf{a_0}} \cdot \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0};\mathbf{ctx} \cdot \overline{\mathbf{a_{comp}}} \cdot \mathbf{Ret}\ !\ \mathbf{v_1};\mathbf{comp} \cdot \overline{\mathbf{a_1}} \cdot \mathbf{End}\ \mathbf{v_2};\mathbf{ctx}$

(H_2) $\vdash \overline{\mathbf{a_{comp}}}$ non-int-$\bar{\mathbf{a}}$

(H_3) ${}^{?}\langle\langle \mathbf{Start} \cdot \overline{\mathbf{a_0}} \cdot \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = \eth;\delta;\bar{\ell};\mathsf{e_0};\mathsf{e'_0};\mathsf{let}\ \mathsf{x_0} = \mathsf{new}\ \mathsf{n_0}\ \mathsf{in}\ \mathsf{e''_0}$

(H_4) ${}^{!}\langle\langle \mathbf{Ret}\ !\ \mathbf{v_1} \cdot \overline{\mathbf{a_1}} \cdot \mathbf{End}\ \mathbf{v_2} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\eth;\delta;\bar{\ell}} = \eth';\delta';[\cdot];\mathsf{e_1};\mathsf{e'_1};\mathsf{let}\ \mathsf{x_1} = \mathsf{new}\ \mathsf{n_1}\ \mathsf{in}\ \mathsf{e''_1}$

(H_5) $\mathsf{e} = \mathsf{e_0};\mathsf{e'_0};\mathsf{let}\ \mathsf{x_0} = \mathsf{new}\ \mathsf{n_0}\ \mathsf{in}\ (\mathsf{e''_0};\mathsf{e_1});\mathsf{e'_1};\mathsf{let}\ \mathsf{x_1} = \mathsf{new}\ \mathsf{n_1}\ \mathsf{in}\ \mathsf{e''_1}$

(H_6) $\Xi_{ctx} = \mathsf{let}\ \mathsf{main}\ \mathsf{x} : \mathbb{N} \to \mathbb{N} := \mathsf{e}, [\cdot]$

Inverting Assumption (H_3):

(H_7) $\vdash \overline{\mathbf{a_0}}$ non-int-$\bar{\mathbf{a}}$

(H_8) ${}^{!}\langle\langle \mathbf{Start} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = \overline{\ell_0};\mathsf{e_0}$

(H_9) $\langle\langle \overline{\mathbf{a_0}} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}} = \mathsf{e'_0}$

(H_{10}) ${}^{?}\langle\langle \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\emptyset;\emptyset;\overline{\ell_0}} = \eth;\delta;\bar{\ell},\mathsf{let}\ \mathsf{x_0} = \mathsf{new}\ \mathsf{n_0}\ \mathsf{in}\ \mathsf{e''_0}$

Inverting Assumption (H_4):

(H_{11}) $\vdash \overline{\mathbf{a_1}}$ non-int-$\bar{\mathbf{a}}$

(H_{12}) ${}^{!}\langle\langle \mathbf{Ret}\ !\ \mathbf{v_1} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\eth;\delta;\bar{\ell}} = \overline{\ell_1};\mathsf{e_1}$

(H_{13}) $\langle\langle \overline{\mathbf{a_1}} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}} = \mathsf{e'_1}$

(H_{14}) ${}^{?}\langle\langle \mathbf{End}\ \mathbf{v_2} \rangle\rangle^{\mathbf{L_{ms}}\to\mathsf{L_{tms}}}_{\eth;\delta;\overline{\ell_1}} = \eth';\delta';[\cdot];\mathsf{let}\ \mathsf{x_1} = \mathsf{new}\ \mathsf{n_1}\ \mathsf{in}\ \mathsf{e''_1}$

By Lemmas 57 and 58 and assumptions (H_7) and (H_{11}) and noting that $\overline{\mathbf{a_0}}$ and $\overline{\mathbf{a_1}}$ are all tagged with $\mathbf{ctx}$, we can conclude:

(H_{15}) $\overline{\mathbf{a_0}} = [\cdot]$

(H_{16}) $\overline{\mathbf{a_1}} = [\cdot]$

Invert Assumption (a) and noting Assumption (H_1):

(H_{15}) $\bar{\mathbf{a}} = \mathbf{Start};\mathbf{comp} \cdot \overline{\mathbf{a_a}}$

(H_{16}) $\mathbf{\Omega} \triangleright \mathbf{call\ main\ 0} \xrightarrow{\mathbf{Start};\mathbf{comp}}_{\mathrm{ctx}} \mathbf{foo}, [\cdot];\mathbf{\Xi};([\cdot];\mathbf{main}), [\cdot];\mathbf{ctx};[\cdot];[\cdot];[\cdot] \triangleright \mathbf{e_0}[\mathbf{0}/\mathbf{x}]$

(H_{17}) $\mathbf{foo}, [\cdot]\,; \mathbf{\Xi}; ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathbf{e_0}[\mathbf{0}/\mathbf{x}] \xrightarrow{\overline{\mathbf{a_a}}}{}^{*}_{\mathrm{ctx}} \mathbf{\Omega}' \triangleright \mathbf{f}$

Invert Assumption (c):

(H_{18}) $\Omega = \Phi; \mathsf{comp}; \Psi$

(H_{19}) $\mathbf{\Omega} = \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}$

(H_{20}) $\Phi \approx \mathbf{\Phi}$

(H_{21}) $\Psi \approx_{\emptyset;\emptyset} \mathbf{\Psi}$

Subsequently inverting Assumption (H_{20}):

(H_{22}) $\Phi = \xi; \Xi; \overline{\mathsf{K}}$

(H_{23}) $\mathbf{\Phi} = \mathbf{foo}, [\cdot]\,; \mathbf{\Xi}; [\cdot]$

(H_{24}) $\Xi \approx \mathbf{\Xi}$

(H_{25}) $\overline{\mathsf{K}} \approx [\cdot]$

Inverting Assumption (H_{25}):

(H_{26}) $\overline{\mathsf{K}} = [\cdot]$

Similarily, inverting Assumption (H_{21}) and noting that $\mathbf{\Psi} = [\cdot]\,; [\cdot]\,; [\cdot]$:

(H_{27}) $\Psi = [\cdot]\,; [\cdot]\,; [\cdot]$

Use Assumption (H_{24}) to conclude:

(H_{28}) $\mathbf{\Xi} = \mathbf{\Xi_{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(H_{29}) $\xi = [\![\xi]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathrm{dom}[\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

Note by inversion on Assumption (H_8) we have:

(H_{30}) $\ell_0 = [\cdot]$

(H_{31}) $\mathsf{e}_0 = 42$

(H_{32}) $^{!}\langle\!\langle \mathbf{Start} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = [\cdot]; 42$

Thus also:

(H_{33}) $^{?}\langle\!\langle \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0} \rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\emptyset;\emptyset;[\cdot]} = \eth; \delta; \overline{\ell}; \mathsf{let}\ \mathsf{x}_0 = \mathsf{new}\ \mathsf{n}_0\ \mathsf{in}\ \mathsf{e}_0''$

From previous assumptions, we also know:

(H_{34}) $\mathsf{foo}, [\cdot]\,; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]\,; [\cdot]\,; [\cdot] \approx_{\emptyset;\emptyset} \mathbf{foo}, [\cdot]\,; \mathbf{\Xi}; [\cdot]\,; \mathbf{comp}; [\cdot]\,; [\cdot]\,; [\cdot]$

(H_{35}) $\Xi_{\mathsf{ctx}} = \mathsf{let}\ \mathsf{main}\ \mathsf{x} : \mathbb{N} \to \mathbb{N} := 42; \mathsf{e}_{01}, [\cdot]$

Now use Lemma 69 (Backtranslation Correctness of **Start**) on Assumptions (H_{16}), (H_{28}), (H_{29}), (H_{32}), (H_{34}) and (H_{35}) to get

(H_{36}) $\mathsf{foo}, [\cdot]\,; \Xi; [\cdot]\,; \mathsf{comp}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathsf{call\ main}\ 0 \xrightarrow{\mathsf{Start};\mathsf{comp}}{}^{n_0'}_{\mathrm{ctx}} \xi; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathsf{e}_{01}[0/\mathsf{x}]$

(H_{37}) $\xi;\Xi;([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};[\cdot];[\cdot];[\cdot] \multimap_{\emptyset;\emptyset} \xi;\Xi;([\cdot];\mathsf{main}),[\cdot];\mathsf{ctx};[\cdot];[\cdot];[\cdot]$

(H_{38}) $\mathsf{Start};\mathsf{comp} \approxeq^{*}_{\emptyset;\emptyset} \mathbf{Start};\mathbf{comp}$

By transitivity, we can split Assumption (H_{17}) arbitrarily, noting Assumption (H_7). So let n_1 be such that:

(H_{39}) $\mathbf{foo},[\cdot];\Xi;([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};[\cdot];[\cdot];[\cdot]\triangleright\mathbf{e_0}[\mathbf{0}/\mathbf{x}] \xrightarrow{\overline{\mathbf{a_0}}}_{\mathrm{ctx}}{}^{n_0} \mathbf{foo},[\cdot];\Xi;\overline{\mathbf{K}},([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};\mathbf{H_0^{ctx}};[\cdot];\mathbf{\Delta_0}$
$\mathbf{e_1}$

(H_{40}) $\mathbf{foo},[\cdot];\Xi;([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};\mathbf{H^{ctx}};[\cdot];\mathbf{\Delta_0}\triangleright\mathbf{e_1} \xrightarrow{\overline{\mathbf{a_b}}}{}^{*}_{\mathrm{ctx}} \mathbf{\Omega'}\triangleright\mathbf{f}$

(H_{41}) $\overline{\mathbf{a_a}} = \overline{\mathbf{a_0}}\cdot\overline{\mathbf{a_b}}$

With Assumptions (H_6), (H_7), (H_9), (H_{28}), (H_{29}), (H_{37}) and (H_{39}), we can use Lemma 71 (Middle of Backtranslation Correctness) to obtain:

(H_{42}) $\mathsf{foo},[\cdot];\Xi;([\cdot];\mathsf{main}),[\cdot];\mathsf{ctx};[\cdot];[\cdot];[\cdot]\triangleright\mathsf{e_{01}}[0/\mathsf{x}] \xrightarrow{\overline{\mathsf{a_0}}}{}^{n_1'}_{\mathrm{ctx}} \xi;\Xi;([\cdot];\mathsf{main}),[\cdot];\mathsf{ctx};[\cdot];[\cdot];[\cdot]\triangleright$
$\mathsf{K}[42]$

(H_{43}) $\mathbf{foo},[\cdot];\Xi;([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};[\cdot];[\cdot];[\cdot] \multimap_{\emptyset;\emptyset} \xi;\Xi;([\cdot];\mathsf{main}),[\cdot];\mathsf{ctx};[\cdot];[\cdot];[\cdot]$

(H_{44}) $\overline{\mathsf{a_0}} \approxeq^{*}_{\emptyset;\emptyset} \overline{\mathbf{a_0}}$

By previous assumptions, it is clear that $\overline{\mathbf{a_b}} = \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0};\mathbf{ctx}\cdot\overline{\mathbf{a_c}}$, so by inversion on the execution Assumption (H_{40}), we know $\mathbf{e_1} = \mathbf{K'}[\mathbf{call}\ \mathbf{foo}\ \mathbf{v_0}]$.
By transitivity, we can split Assumption (H_{40}) arbitrarily:

(H_{45}) $\mathbf{foo},[\cdot];\Xi;\overline{\mathbf{K}},([\cdot];\mathbf{main}),[\cdot];\mathbf{ctx};\mathbf{H_0^{ctx}};[\cdot];\mathbf{\Delta_0}\triangleright\mathbf{K}[\mathbf{call}\ \mathbf{foo}\ \mathbf{v_0}] \xrightarrow{\mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0};\mathbf{ctx}}$
${}_{\mathrm{ctx}}$ $\mathbf{foo},[\cdot];\Xi;(\mathbf{K};\mathbf{foo}),\overline{\mathbf{K}},([\cdot];\mathbf{main}),[\cdot];\mathbf{comp};\mathbf{H_0^{ctx}};[\cdot];\mathbf{\Delta_0}\triangleright\mathbf{e_{foo}}[\mathbf{v_0}/\mathbf{y}]$

(H_{46}) $\mathbf{foo},[\cdot];\Xi;(\mathbf{K};\mathbf{foo}),\overline{\mathbf{K}},([\cdot];\mathbf{main}),[\cdot];\mathbf{comp};\mathbf{H_0^{ctx}};[\cdot];\mathbf{\Delta_0}\triangleright\mathbf{e_{foo}} \xrightarrow{\overline{\mathbf{a_c}}}{}^{*}_{\mathrm{ctx}} \mathbf{\Omega'}\triangleright$
$\mathbf{f}$

(H_{47}) $\overline{\mathbf{a_b}} = \mathbf{Call}\ ?\ \mathbf{foo}\ \mathbf{v_0};\mathbf{ctx}\cdot\overline{\mathbf{a_c}}$

By the fact that $\mathbf{foo} \in \xi$, we have $\mathbf{foo} \in \mathrm{dom}[\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$ and thus $\mathbf{e_{foo}} = [\![\mathsf{e_{foo}}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$, where $\mathsf{let}\ \mathsf{foo}\ \mathsf{x} : \tau_\lambda := \mathsf{e_{foo}} \in \Xi_{\mathsf{comp}}$. Now, let:

(H_{45}) $\vdash \ell$ *fresh*

(H_{46}) $\Delta_0 \vdash \ell$ *fresh*

(H_{47}) $\vdash \mathsf{z}$ *fresh*

(H_{48}) $\delta = \{\ell \mapsto \ell\}$

(H_{49}) $\eth = \{\ell \mapsto \mathsf{z}\}$

We are ready to apply Lemma 72 (Backtranslation Correctness of $\mathbf{Call}\ ?$) with Assumptions (H_{28}), (H_{29}), (H_{33}), (H_{43}) and (H_{45}) to (H_{49}):

(H_{50}) $\mathsf{L} = \{\ell\}$

(H_{51}) $\mathsf{X} = \{\mathsf{Alloc}\ \ell\ 42\}$

(H_{52}) $\mathsf{foo}, [\cdot] ; \Xi; ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{ctx}; [\cdot] ; [\cdot] ; [\cdot] \triangleright \mathsf{K}'[\mathsf{let}\ \mathsf{x} = \mathsf{new}\ 42\ \mathsf{in}\ \mathsf{call}\ \mathsf{foo}\ \langle\!\langle\!\langle \mathbf{v_0} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}] \xrightarrow{\overline{\mathsf{a_1}}}{}_{\mathsf{ctx}}^{n_2'}$ $\mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}, \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi \triangleright \mathsf{e_{foo}}[\langle\!\langle\!\langle \mathbf{v_0} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} / \mathsf{y}]$

(H_{53}) $\mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}; \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi \approx_{\delta;\mathsf{L}} \mathbf{foo}, [\cdot] ; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot] ; \mathbf{main}), [\cdot] ; \mathbf{comp}; \mathbf{H_0^{ctx}}; [\cdot] ; \boldsymbol{\Delta}$

(H_{54}) $\overline{\mathsf{a_1}} \approxeq^{*}_{\delta;\mathsf{X}} \mathbf{Call\ ?\ foo\ v_0; ctx}$

By transitivity, we can split Assumption (H_{46}) arbitrarily, noting that the immediate next action after $\overline{\mathbf{a_{comp}}}$ is $\mathbf{Ret\ !\ v_1}$. So let n_3 be such that:

(H_{55}) $\mathbf{foo}, [\cdot] ; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot] ; \mathbf{main}), [\cdot] ; \mathbf{comp}; \mathbf{H_0^{ctx}}; [\cdot] ; \boldsymbol{\Delta_0} \triangleright \mathbf{e_{foo}}[\mathbf{v_0}/\mathbf{y}] \xrightarrow{\overline{\mathbf{a_{comp}}}}{}_{\mathsf{ctx}}^{n_3}$ $\mathbf{foo}, [\cdot] ; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot] ; \mathbf{main}), [\cdot] ; \mathbf{comp}; \mathbf{H_0^{ctx}}; \mathbf{H_0^{comp}}; \boldsymbol{\Delta_1} \triangleright \mathbf{K_c}[\mathbf{return\ v_1}]$

(H_{56}) $\mathbf{foo}, [\cdot] ; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot] ; \mathbf{main}), [\cdot] ; \mathbf{comp}; \mathbf{H_0^{ctx}}; \mathbf{H_0^{comp}}; \boldsymbol{\Delta_1} \triangleright \mathbf{K_c}[\mathbf{return\ v_1}] \xrightarrow{\overline{\mathbf{a_d}}}{}_{\mathsf{ctx}}^{*}$ $\boldsymbol{\Omega}' \triangleright \mathbf{f}$

(H_{57}) $\overline{\mathbf{a_c}} = \overline{\mathbf{a_{comp}}} \cdot \overline{\mathbf{a_d}}$

Remember that $[\![\mathsf{e_{foo}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{e_{foo}}$.
With Assumption (g) use Lemma 68 (Backtranslation is Well-Typed):

(H_{58}) $\Gamma \vdash \Xi\,\mathrm{ok}$

Now note that Lemma 22 (Static Typing implies Runtime Typing (Toplevel)) with Assumption (H_{58}) gives:

(H_{59}) $\vdash \mathsf{foo}, [\cdot] ; \Xi; [\cdot] ; \mathsf{comp}; [\cdot] ; [\cdot] ; [\cdot] \triangleright \mathsf{call\ main}\ 0 : \mathbb{N}$

By Lemma 39 (Steps Preservation) with Assumptions (H_{36}) and (H_{59}):

(H_{60}) $\vdash \xi; \Xi; ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{ctx}; [\cdot] ; [\cdot] ; [\cdot] \triangleright \mathsf{e_{01}}[0/\mathsf{x}] : \mathbb{N}$

By Lemma 39 (Steps Preservation) with Assumptions (H_{42}) and (H_{60}):

(H_{61}) $\vdash \xi; \Xi; ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{ctx}; [\cdot] ; [\cdot] ; [\cdot] \triangleright \mathsf{K}[42] : \mathbb{N}$

By Lemma 39 (Steps Preservation) with Assumptions (H_{52}) and (H_{61}):

(H_{62}) $\vdash \mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}, \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi \triangleright \mathsf{e_{foo}}[\langle\!\langle\!\langle \mathbf{v_0} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}} / \mathsf{y}] : \mathbb{N}$

Using Assumption (H_{12}):

(H_{58}) $\mathsf{K} = [\cdot] ; \mathsf{delete\ z}; \mathsf{K_r}[\langle\!\langle\!\langle \mathbf{v_1} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}]$

Use Assumptions (H_{53}), (H_{55}) and (H_{62}) to apply Lemma 56 (Component Correctness):

(H_{60}) $\delta \subseteq \delta'$

(H_{61}) $\mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}; \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi \triangleright \mathsf{e_{foo}}[\mathsf{v_0}/\mathsf{y}] \xrightarrow{\overline{\mathsf{a_{comp}}}}{}_{\mathsf{ctx}}^{n_3'}$ $\mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}; \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi' \triangleright$ $\mathsf{K_c}[\mathsf{return\ v_1}]$

(H_{62}) $\mathsf{foo}, [\cdot] ; \Xi; (\mathsf{K}; \mathsf{foo}), ([\cdot] ; \mathsf{main}), [\cdot] ; \mathsf{comp}; \Psi' \approx_{\delta';\mathsf{L}} \mathbf{foo}, [\cdot] ; \mathbf{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot] ; \mathbf{main}), [\cdot] ; \mathbf{comp}; \mathbf{H_0^{ctx}}; \mathbf{H_0^{co}}$

(H_{63}) $\overline{\mathsf{a_{comp}}} \approxeq^{*}_{\delta';\mathsf{X}} \overline{\mathbf{a_{comp}}}$

By transitivity, we can split Assumption (H_{56}) arbitrarily:

(H_{64}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; (\mathbf{K}; \mathbf{foo}), \overline{\mathbf{K}}, ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{comp}; \boldsymbol{\Psi}' \triangleright \mathbf{K_c}[\mathbf{return}\ \mathbf{v_1}] \xrightarrow{\mathbf{Ret}\ !\ \mathbf{v_1}}_{\mathrm{ctx}} \mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; \overline{\mathbf{K}}, ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}' \triangleright \mathbf{K}[\mathbf{v_1}]$

(H_{65}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; \overline{\mathbf{K}}, ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}' \triangleright \mathbf{K}[\mathbf{v_1}] \xrightarrow{\overline{\mathbf{a_e}}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}$

(H_{66}) $\overline{\mathbf{a_d}} = \mathbf{Ret}\ !\ \mathbf{v_1} \cdot \overline{\mathbf{a_e}}$

Remember that $[\![\mathsf{e_{foo}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{e_{foo}}$, so:

(H_{67}) $[\![\mathsf{K_c}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} = \mathbf{K_c}$.

Using Assumptions (H_{48}) and (H_{60}):

(H_{68}) $\delta'(\ell) = \ell$

We are ready to apply Lemma 70 (Backtranslation Correctness of **Ret**) with Assumptions (H_{12}), (H_{28}), (H_{29}), (H_{49}), (H_{50}), (H_{58}), (H_{62}), (H_{64}), (H_{67}) and (H_{68}):

(H_{69}) $\mathsf{X}' = \mathsf{X} \cup \{\mathsf{Dealloc}\ \ell\}$

(H_{70}) $\mathsf{foo}, [\cdot]\,; \Xi; (\mathsf{K}; \mathsf{foo}), ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{comp}; \Psi' \triangleright \mathsf{K_c}[\mathsf{return}\ \mathsf{v_1}] \xrightarrow{\overline{\mathsf{a_2}}}{}^{n_4'}_{\mathsf{ctx}} \mathsf{foo}, [\cdot]\,; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{v_1}]$

(H_{71}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; \overline{\mathbf{K}}, ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}' \multimap_{\delta'; \mathsf{X}'} \mathsf{foo}, [\cdot]\,; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi'$

(H_{72}) $\overline{\mathsf{a_2}} \approxeq^{*}_{\delta'; \mathsf{L}} \mathbf{Ret}\ !\ \mathbf{v_1}$

By transitivity, we can split Assumption (H_{65}) arbitrarily, noting that the immediate next action after $\overline{\mathbf{a_1}}$ is $\mathbf{End}\ \mathbf{v_2}$. So, let n_5 be such that:

(H_{73}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; \overline{\mathbf{K}}, ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}' \triangleright \mathbf{K}[\mathbf{v_1}] \xrightarrow{\overline{\mathbf{a_1}}}{}_{\mathrm{ctx}}{}^{n_5} \mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}'' \triangleright \mathbf{K_e}[\mathbf{v_2}]$

(H_{74}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}'' \triangleright \mathbf{K_e}[\mathbf{v_2}] \xrightarrow{\mathbf{End}\ \mathbf{v_2}}{}^{*}_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}$

(H_{75}) $\overline{\mathbf{a_e}} = \overline{\mathbf{a_1}} \cdot \mathbf{End}\ \mathbf{v_2}$

With Assumptions (H_6), (H_{11}), (H_{13}), (H_{28}), (H_{29}), (H_{71}) and (H_{73}), we can use Lemma 71 (Middle of Backtranslation Correctness) to obtain:

(H_{76}) $\mathsf{foo}, [\cdot]\,; \Xi; \overline{\mathsf{K}}, ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi' \triangleright \mathsf{K}[\mathsf{v_1}] \xrightarrow{\overline{\mathsf{a_3}}}{}^{n_5'}_{\mathsf{ctx}} \xi; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi' \triangleright \mathsf{K_e}\,[42]$

(H_{77}) $\mathbf{foo}, [\cdot]\,; \boldsymbol{\Xi}; ([\cdot]\,; \mathbf{main}), [\cdot]\,; \mathbf{ctx}; \boldsymbol{\Psi}'' \multimap_{\delta'; \emptyset} \xi; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi'$

(H_{78}) $\overline{\mathsf{a_3}} \approxeq^{*}_{\delta'; \emptyset} \overline{\mathbf{a_1}}$

Let:

(H_{79}) $\partial, \delta' \vdash \ell'$ *fresh*

(H_{80}) $\partial \vdash \mathsf{w}$ *fresh*

(H_{81}) $\delta \vdash \ell'$ *fresh*

(H_{82}) $\delta'' = \delta' \cup \{\ell' \mapsto \ell'\}$

(H_{83}) $\partial' = \partial \cup \{\ell' \mapsto \mathsf{w}\}$

Finally, use Lemma 73 (Backtranslation Correctness of **End**) with Assumptions (H_{14}), (H_{28}), (H_{29}), (H_{74}), (H_{77}) and (H_{79}) to (H_{83}) to acquire:

(H_{79}) $\mathsf{L}' = \mathsf{L} \cup \{\ell'\}$

(H_{80}) $\mathsf{X}'' = \mathsf{X}' \cup \{\mathsf{Alloc}\ \ell'\ 42, \mathsf{Dealloc}\ \ell'\}$

(H_{81}) $\mathsf{foo}, [\cdot]\,; \Xi; ([\cdot]\,; \mathsf{main}), [\cdot]\,; \mathsf{ctx}; \Psi'' \triangleright \mathsf{K_e}[42] \xrightarrow{\overline{\mathsf{a}_4}}{}^{n_6'}_{\mathsf{ctx}}\ \ \xi; \Xi; [\cdot]\,; \mathsf{ctx}; \Psi'' \triangleright \mathsf{v_2}$

(H_{82}) $\xi; \Xi; [\cdot]\,; \mathsf{ctx}; \Psi'' \approx_{\delta'';\mathsf{L}'} \mathbf{foo}, [\cdot]\,; \mathbf{\Xi}; [\cdot]\,; \mathbf{ctx}; \mathbf{\Psi''}$

(H_{83}) $\overline{\mathsf{a}_4} \approxeq^*_{\delta'';\mathsf{X}''} \mathbf{End\ v_2}$

Finally, instantiate the existentials in our goals as follows:

(H_{84}) $\delta_e = \delta''$

(H_{85}) $\mathsf{L_e} = \mathsf{L}'$

(H_{86}) $\mathsf{X_e} = \mathsf{X}''$

(H_{87}) $\Omega_{\mathsf{e}}' = \xi; \Xi; [\cdot]\,; \mathsf{ctx}; \Psi''$

(H_{88}) $\overline{\mathsf{a}_\mathsf{e}} = \mathsf{Start}; \mathsf{comp} \cdot \overline{\mathsf{a}_0} \cdot \overline{\mathsf{a}_1} \cdot \overline{\mathsf{a}_{\mathsf{comp}}} \cdot \overline{\mathsf{a}_2} \cdot \overline{\mathsf{a}_3} \cdot \overline{\mathsf{a}_4}$

We solve Goal (i) by transitivity, making use of Assumptions (H_{36}), (H_{42}), (H_{52}), (H_{61}), (H_{70}), (H_{76}) and (H_{81}) and Lemma 20 ($\rightarrow^n_{ctx}$ and $\rightarrow^*_{ctx}$ yield $\rightarrow^*_{ctx}$).

Goal (ii) by Assumption (H_{82}).

Lastly, Goal (iii) by transitivity, making use of Assumptions (H_{38}), (H_{44}), (H_{54}), (H_{63}), (H_{72}), (H_{78}) and (H_{83}). □

Definition 41 ($\mathsf{L}_{\mathrm{tms}}$ Robust Satisfaction)**.** *We write* $\Xi_{\mathsf{comp}} \vDash_R \pi$ *for If*

(a) $\mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \xRightarrow{\overline{\mathsf{a}}} \Omega \triangleright \mathsf{f}_{\lightning}$

Then $\exists \delta_{MS}$

(i) $\theta_{\delta_{MS}}(\overline{\mathsf{a}}) \in \pi$

Definition 42 ($\mathbf{L}_{\mathrm{ms}}$ Robust Satisfaction)**.** *We write* $\mathbf{\Xi_{comp}} \vDash_R \pi$ *for If*

(a) $\mathbf{prog}\ \mathbf{\Xi_{ctx}}\ \mathbf{\Xi_{comp}} \xRightarrow{\overline{\mathbf{a}}} \mathbf{\Omega} \triangleright \mathbf{f}_{\lightning}$

Then $\exists \delta_{MS}$

(i) $\theta_{\delta_{MS}}(\overline{\mathbf{a}}) \in \pi$

Theorem 3 (Robust TMS Preservation)**.**

(i) $\vdash [\![\bullet]\!]^{\mathsf{L}_{tms} \rightarrow \mathbf{L}_{ms}} : \lceil \text{tmssafe} \rceil$

Proof. Unfolding the definition: If

(a) $\pi \in \lceil \text{tmssafe} \rceil$

(b) $\Xi_{\mathsf{comp}} \vDash_R \pi$

then

(i) $[\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \vDash_R \pi$

Unfold Assumption (i): If

(F_1) $\mathbf{prog}\ \mathbf{\Xi_{ctx}}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}} \overset{\overline{a}}{\Longrightarrow} \mathbf{\Omega} \triangleright \mathbf{f}_{\not{\imath}}$

then $\exists \delta_{\mathrm{MS}}$

(i) $\theta_{\delta_{\mathrm{MS}}}(\overline{\mathbf{a}}) \in \pi$

Suppose $\mathbf{f}_{\not{\imath}} = \mathbf{v}$.Invert Assumption (F_1):

(H_1) $\mathbf{\Xi} = \mathbf{\Xi_{ctx}} \bowtie [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(H_2) $\mathbf{main} \notin \xi$

(H_3) $\xi = \mathrm{dom}\ [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$

(H_4) $[\![\xi]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}; \mathbf{\Xi}; \emptyset; \mathbf{comp}; \emptyset; \emptyset; \emptyset \triangleright \mathbf{call\ main\ 0} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{\mathrm{ctx}} \mathbf{\Omega} \triangleright \mathbf{v}$

We backtranslate:

(H_5) $\langle\!\langle\!\langle \overline{\mathbf{a}} \rangle\!\rangle\!\rangle^{\mathrm{L_{ms}} \to \mathrm{L_{tms}}}_{\emptyset;\emptyset} = \eth; \delta; \mathsf{let\ main\ z\ := e}, [\cdot]$.

Invert Assumption (H_5), so we can decompose $\overline{\mathbf{a}}$ into $\mathbf{Start} \cdot \overline{\mathbf{a}_0^{\mathbf{ctx}}} \cdot \mathbf{Call\ ?\ foo\ n} \cdot \overline{\mathbf{a^{comp}}} \cdot \mathbf{Ret\ !\ m} \cdot \overline{\mathbf{a}_1^{\mathbf{ctx}}} \cdot \mathbf{End\ n'}$ such that $\vdash \overline{\mathbf{a^{comp}}}$ non-int-**trace**, $\vdash \overline{\mathbf{a}_0^{\mathbf{ctx}}}$ non-int-**trace**, and $\vdash \overline{\mathbf{a}_1^{\mathbf{ctx}}}$ non-int-**trace**.

By Lemma 62 (MS-Location Generator) using $\mathrm{dom}\,\delta$, there is a $\delta_{MS} : \bullet \to \underline{\bullet}$ injective.

Instantiate the goal:

(H_6) $\delta'_{MS} : \bullet \to \underline{\bullet}, \ell : \mathbf{L} \mapsto \delta_{MS}(\delta^{-1}(\ell))$

From the shape of the trace $\overline{\mathbf{a}}$, $\mathbf{foo} \in [\![\Xi_{\mathsf{comp}}]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}$.
So $\mathsf{foo} \in \Xi_{\mathsf{comp}}$ and $\mathbf{main} \in \mathbf{\Xi_{ctx}}$.
Let $\Xi_{\mathsf{ctx}} = (\mathsf{let\ main\ z\ := e}), [\cdot]$.
Let $\Xi = \Xi_{\mathsf{ctx}} \bowtie \Xi_{\mathsf{comp}}$ which follows by inverting Assumption (H_1).
Let $\Gamma_0 = \Xi \downarrow$ and note that $\mathsf{main} \in \mathrm{dom}\,\Gamma_0$.
We need to verify that $\Gamma_0 \vdash \Xi$ ok to conclude $\vdash \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} \dashv \Xi, \mathrm{dom}\,\Xi_{\mathsf{comp}}$. This follows from Lemma 68 (Backtranslation is Well-Typed).

By Rules state-eq, empty-memstate-eq, cfstate-eq and empty-commlib-lib-eq to empty-kontstack-eq, we have $\xi; \Xi; \emptyset; \mathsf{comp}; \emptyset; \emptyset; \emptyset \approx_{\delta;\mathrm{L}} [\![\xi]\!]^{\mathrm{L_{tms}} \to \mathrm{L_{ms}}}; \mathbf{\Xi}; \emptyset; \mathbf{comp}; \emptyset; \emptyset; \emptyset$.

Finally, use Lemma 63 (Filters Equal), whose assumptions are satisfied by the results we got from applying backtranslation correctness. $\square$

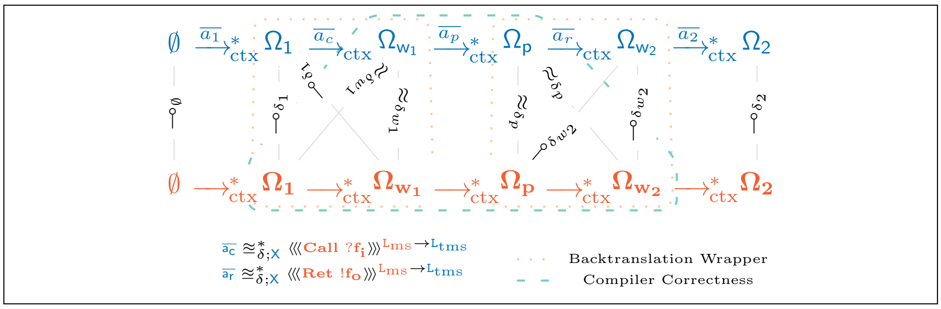

Figure 55: Sketch of the proof of ?? 3 (Robust TMS Preservation). Arrows in the horizontal direction are the usual step relations, but the term is omitted for clarity. Likewise for the emitted target-level traces. Trace $\overline{a_1}$ starts with Start and $\overline{a_2}$ ends with End v, where v is the final result of the program. Target states $\mathbf{\Omega_1}$ and $\mathbf{\Omega_{w_1}}$ are exactly equal, so are $\mathbf{\Omega_p}$ and $\mathbf{\Omega_{w_2}}$. They are drawn for aesthetic purposes.

4.5 Robust SMS Preserving Compiler

For instrumented $\mathrm{L_{ms}}$ code, we'll write L. The syntax and semantics are entirely identical.

4.5.1 Translation to Specification Events

$$\boxed{\delta_{MS}(\ell) = \underline{\ell}} \quad \text{„Map } \mathrm{L_{ms}} \text{ locations } \ell \text{ to abstract locations } \underline{\ell}.\text{"}$$

$$\boxed{T_{\mathrm{SMS}} \simeq_{\delta_{MS}} \mathbf{\Delta}} \quad \text{„Abstract memory state } T_{\mathrm{TMS}} \text{ describes the concrete state } \mathbf{\Delta}.\text{"}$$

$$\frac{}{\emptyset \simeq_{\delta_{MS}} [\cdot]}\ (\textsf{SMS-Empty-Agree}) \qquad \frac{}{\emptyset \simeq_{\delta_{MS}} \lightning}\ (\textsf{SMS-Abort-Agree})$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{\ell} \notin T_{\mathrm{SMS}} \qquad T_{\mathrm{SMS}} \simeq_{\delta_{MS}} \mathbf{\Delta}}{\{(\underline{\ell}, \boldsymbol{m})\} \cup T_{\mathrm{SMS}} \simeq_{\delta_{MS}} \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{n}), \mathbf{\Delta}}\ (\textsf{SMS-Cons-Agree})$$

$$\frac{T_{\mathrm{SMS}} \simeq_{\delta_{MS}} \mathbf{\Delta}}{T_{\mathrm{SMS}} \simeq_{\delta_{MS}} \mathbf{x} \mapsto (\ell; \mathbf{ctx}; \rho; \mathbf{n}), \mathbf{\Delta}}\ (\textsf{SMS-Ignore-Agree})$$

Figure 56: Store Agreement.

$\boxed{\delta_{MS}(\ell) = \underline{\ell}}$ „A map from $\mathrm{L_{ms}}$ memory locations ℓ to specification locations $\underline{\ell}$."

$\boxed{\theta_{\delta_{MS}}(\mathbf{a}) = \overline{a^{\mathrm{ms}}}}$ „Project an $\mathrm{L_{ms}}$ event to specification events."

$$\frac{\mathbf{a_b} \neq \lightning}{\theta_{\delta_{MS}}(\mathbf{a_b};\mathbf{ctx}) = \underline{\varepsilon}}\ \text{(t-filter-context)} \qquad \frac{}{\theta_{\delta_{MS}}(\mathbf{Start};\mathbf{comp}) = \underline{\varepsilon}}\ \text{(t-filter-comp-start)}$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathbf{n}}{\theta_{\delta_{MS}}(\mathbf{Alloc}\ \ell\ \mathbf{n};\mathbf{comp}) = \underline{Alloc\ \ell\ n}}\ \text{(t-filter-comp-alloc)}$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell}}{\theta_{\delta_{MS}}(\mathbf{Dealloc}\ \ell;\mathbf{comp}) = \underline{Dealloc\ \ell}}\ \text{(t-filter-comp-dealloc)}$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathbf{n}}{\theta_{\delta_{MS}}(\mathbf{Get}\ \ell\ \mathbf{n};\mathbf{comp}) = \underline{Use\ \ell\ n}}\ \text{(t-filter-comp-get)}$$

$$\frac{\delta_{MS}(\ell) = \underline{\ell} \qquad \underline{n} = \mathbf{n}}{\theta_{\delta_{MS}}(\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v};\mathbf{comp}) = \underline{Use\ \ell\ n}}\ \text{(t-filter-comp-set)}$$

$$\frac{}{\theta_{\delta_{MS}}(\mathbf{Call\ c\ foo\ v};\mathbf{comp}) = \underline{\varepsilon}}\ \text{(t-filter-comp-call)} \qquad \frac{}{\theta_{\delta_{MS}}(\mathbf{Ret\ c\ v};\mathbf{comp}) = \underline{\varepsilon}}\ \text{(t-filter-comp-ret)}$$

$$\frac{}{\theta_{\delta_{MS}}(\lightning;\mathbf{t}) = \underline{\lightning}}\ \text{(t-filter-abort)}$$

$\boxed{\theta^*_{\delta_{MS}}(\overline{\mathbf{a}}) = \overline{a^{\mathrm{ms}}}}$ „Project an $\mathrm{L_{ms}}$ trace to specification traces."

$$\frac{}{\theta^*_{\delta_{MS}}([\cdot]) = \underline{[\cdot]}}\ \text{(t-filter-empty)} \qquad \frac{\theta_{\delta_{MS}}(\mathbf{a}) = \underline{a} \qquad \theta^*_{\delta_{MS}}(\overline{\mathbf{a}}) = \overline{a^{\mathrm{ms}}} \qquad \underline{a} \neq \underline{\varepsilon}}{\theta^*_{\delta_{MS}}(\mathbf{a}\cdot\overline{\mathbf{a}}) = \underline{a}\cdot\overline{a^{\mathrm{ms}}}}\ \text{(t-filter-cons-relevant)}$$

$$\frac{\theta_{\delta_{MS}}(\mathbf{a}) = \underline{\varepsilon} \qquad \theta^*_{\delta_{MS}}(\overline{\mathbf{a}}) = \overline{a^{\mathrm{ms}}}}{\theta^*_{\delta_{MS}}(\mathbf{a}\cdot\overline{\mathbf{a}}) = \overline{a^{\mathrm{ms}}}}\ \text{(t-filter-cons-relevant)}$$

Figure 57: Projection of $\mathrm{L_{ms}}$ events to specification events.

4.5.2 Compiler

$\boxed{[\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = e'}$ „Compile $\mathrm{L_{ms}}$ expression $\mathbf{e}$ to $\mathrm{L_{ms}}$ expression e'."

$\boxed{[\![[\mathbf{v}/\mathbf{x}]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = [v/x]}$ „Compile $\mathrm{L_{ms}}$ substitution to L substitution."

$\boxed{[\![\xi]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = \xi'}$ „Compile $\mathrm{L_{ms}}$ component library to L component library."

$$
\begin{aligned}
[\![\mathbf{f}_{\lightning}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= f_{\lightning} \\
[\![\mathbf{call\ foo\ e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{call}\ [\![\mathbf{foo}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{return\ e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{return}\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{e_1} \oplus \mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= [\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \oplus [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{x}[\mathbf{e}]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{let}\ x_{ACCESS} = [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in} \\
&\quad\ \mathit{ifz}\ 0 \leq x_{ACCESS} < x_{SIZE}\ \mathit{then} \\
&\quad\quad x[x_{ACCESS}] \\
&\quad\ \mathit{else}\ \mathit{abort}() \\
[\![\mathbf{let\ x = e_1\ in\ e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{let}\ x = [\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in}\ [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{x}[\mathbf{e_1}] \leftarrow \mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{let}\ x_{ACCESS} = [\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in} \\
&\quad\ \mathit{ifz}\ 0 \leq x_{ACCESS} < x_{SIZE}\ \mathit{then} \\
&\quad\quad x[x_{ACCESS}] \leftarrow [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
&\quad\ \mathit{else}\ \mathit{abort}() \\
[\![\mathbf{let\ x = new\ e_1\ in\ e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{let}\ x_{\mathrm{SIZE}} = [\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in} \\
&\quad\ \mathit{let}\ x = \mathit{new}\ x_{\mathrm{SIZE}}\ \mathit{in}\ [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{delete\ x}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{delete}\ x \\
[\![\mathbf{x\ is}\ ☣]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= x\ \mathit{is}\ ☣ \\
[\![(\mathbf{e_1}; \mathbf{e_2})]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= ([\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}; [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}) \\
[\![\pi_1\ \mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \pi_1\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\pi_2\ \mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \pi_2\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
[\![\mathbf{e\ has}\ \tau]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{has}\ \tau \\
[\![\mathbf{ifz\ e_1\ then\ e_2\ else\ e_3}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{ifz}\ [\![\mathbf{e_1}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
&\quad\ \mathit{then}\ [\![\mathbf{e_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
&\quad\ \mathit{else}\ [\![\mathbf{e_3}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \\
\\
[\![[\mathbf{v}/\mathbf{x}]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= [\,[\![\mathbf{v}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} / [\![\mathbf{x}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\,] \\
\\
[\![[\cdot]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= [\cdot] \\
[\![\mathbf{foo}, \xi]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} &= \mathit{foo}, [\![\xi]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}
\end{aligned}
$$

Figure 58: Compiler from $\mathrm{L_{ms}}$ to L.

$$\boxed{[\![\mathbf{F}]\!]^{L_{ms}\to L} = F} \quad \text{„Compile } L_{ms} \text{ procedures to } L \text{ procedures.”}$$

$$[\![\mathbf{let\ foo\ x := e}]\!]^{L_{ms}\to L} = \mathit{let\ foo\ x} := [\![\mathbf{e}]\!]^{L_{ms}\to L}$$

$$\boxed{[\![\mathbf{\Xi}]\!]^{L_{ms}\to L} = \Xi} \quad \text{„Compile } L_{ms} \text{ libraries to } L \text{ libraries.”}$$

$$\begin{aligned}
[\![[\cdot]]\!]^{L_{ms}\to L} &= [\cdot] \\
[\![\mathbf{F}, \mathbf{\Xi}]\!]^{L_{ms}\to L} &= [\![\mathbf{F}]\!]^{L_{ms}\to L}, [\![\mathbf{\Xi}]\!]^{L_{ms}\to L}
\end{aligned}$$

Figure 59: Compiler from L_{ms} components to L components.

$$\boxed{[\![\mathbf{K}]\!]^{L_{ms}\to L} = K} \quad \text{„Compile } L_{ms} \text{ evaluation contexts to } L \text{ evaluation contexts.”}$$

$$\begin{aligned}
[\![[\cdot]]\!]^{L_{ms}\to L} &= [\cdot] \\
[\![\mathbf{K} \oplus \mathbf{e}]\!]^{L_{ms}\to L} &= [\![\mathbf{K}]\!]^{L_{ms}\to L} \oplus [\![\mathbf{e}]\!]^{L_{ms}\to L} \\
[\![\mathbf{v} \oplus \mathbf{K}]\!]^{L_{ms}\to L} &= [\![\mathbf{v}]\!]^{L_{ms}\to L} \oplus [\![\mathbf{K}]\!]^{L_{ms}\to L} \\
[\![\mathbf{x}[\mathbf{K}]]\!]^{L_{ms}\to L} &= \mathit{let}\ x_{ACCESS} = [\![\mathbf{K}]\!]^{L_{ms}\to L}\ \mathit{in} \\
&\quad \mathit{if}\ 0 \le x_{ACCESS} < x_{SIZE}\ \mathit{then} \\
&\qquad x[x_{ACCESS}] \\
&\quad \mathit{else\ abort}() \\
[\![\mathbf{let\ x\ = K\ in\ e}]\!]^{L_{ms}\to L} &= \mathit{let}\ x\ = [\![\mathbf{K}]\!]^{L_{ms}\to L}\ \mathit{in}\ [\![\mathbf{e}]\!]^{L_{ms}\to L} \\
[\![\mathbf{x}[\mathbf{K}] \leftarrow \mathbf{e}]\!]^{L_{ms}\to L} &= \mathit{let}\ x_{ACCESS} = [\![\mathbf{K}]\!]^{L_{ms}\to L}\ \mathit{in} \\
&\quad \mathit{if}\ 0 \le x_{ACCESS} < x_{SIZE}\ \mathit{then} \\
&\qquad x[x_{ACCESS}] \leftarrow [\![\mathbf{e}]\!]^{L_{ms}\to L} \\
&\quad \mathit{else\ abort}() \\
[\![\mathbf{x}[\mathbf{n}] \leftarrow \mathbf{K}]\!]^{L_{ms}\to L} &= \mathit{if}\ 0 \le n < x_{SIZE}\ \mathit{then} \\
&\qquad x[n] \leftarrow [\![\mathbf{K}]\!]^{L_{ms}\to L} \\
&\quad \mathit{else\ abort}() \\
[\![\mathbf{let\ x\ = new\ K\ in\ e}]\!]^{L_{ms}\to L} &= \mathit{let}\ x_{SIZE} = [\![\mathbf{K}]\!]^{L_{ms}\to L}\ \mathit{in} \\
&\quad \mathit{let}\ x = \mathit{new}\ x_{SIZE}\ \mathit{in}\ [\![\mathbf{e}]\!]^{L_{ms}\to L} \\
[\![\mathbf{ifz\ K\ then\ e_1\ else\ e_2}]\!]^{L_{ms}\to L} &= \mathit{ifz}\ [\![\mathbf{K}]\!]^{L_{ms}\to L}\ \mathit{then} \\
&\qquad [\![\mathbf{e_1}]\!]^{L_{ms}\to L} \\
&\quad \mathit{else} \\
&\qquad [\![\mathbf{e_2}]\!]^{L_{ms}\to L} \\
[\![\mathbf{call\ foo\ K}]\!]^{L_{ms}\to L} &= \mathit{call\ foo}\ [\![\mathbf{K}]\!]^{L_{ms}\to L} \\
[\![\mathbf{return\ K}]\!]^{L_{ms}\to L} &= \mathit{return}\ [\![\mathbf{K}]\!]^{L_{ms}\to L}
\end{aligned}$$

Figure 60: Compiling L_{ms} evaluation contexts to L evaluation contexts.

$$\frac{\Delta_1, x \mapsto (\ell; \mathit{comp}; \rho; m), \Delta_2 \Vdash \gamma}{\Delta_1, x \mapsto (\ell; \mathit{comp}; \rho; m), \Delta_2 \Vdash [m/x_{SIZE}], \gamma}\ (\gamma\text{-}x_{SIZE}\text{-preserve}) \qquad \frac{\Delta \Vdash \gamma}{\Delta \Vdash [v/x], \gamma}\ (\gamma\text{-}x_{SIZE}\text{-skip})$$

$$\frac{}{\Delta \Vdash [\cdot]}\ (\gamma\text{-}x_{SIZE}\text{-empty})$$

Figure 61: Well formed L substitutions.

$$\frac{}{[\cdot] \prec [\cdot]}\ \text{(subst-subset-empy)} \qquad \frac{\mathbf{x} = x \quad \mathbf{v} = v \quad \gamma \prec \gamma}{[\mathbf{v}/\mathbf{x}], \gamma \prec [v/x], \gamma}\ \text{(subst-subset-cons)} \qquad \frac{\gamma \prec \gamma}{\gamma \prec [v/x_{SIZE}], \gamma}\ \text{(subst-subset-}x_{SIZE}\text{)}$$

Figure 62: Cross-Language Substitutions subset.

$\boxed{\delta_{\mathsf{sCCT}}(\ell) = \underline{\ell}}$ „A map from L memory locations ℓ to specification locations $\underline{\ell}$."

$\boxed{\theta_{\delta_{\mathsf{sCCT}}}(a) = a^{\mathrm{scct}}}$ „Project an L event to specification events."

$$\frac{a_b \neq ↯}{\theta_{\delta_{\mathsf{sCCT}}}(a_b;\ ctx) = \underline{\varepsilon}}\ \text{(i-scct-filter-context)} \qquad \frac{}{\theta_{\delta_{\mathsf{sCCT}}}(Start;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-start)}$$

$$\frac{}{\theta_{\delta_{\mathsf{sCCT}}}(End\ \ v;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-end)} \qquad \frac{\delta_{\mathsf{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = n}{\theta_{\delta_{\mathsf{sCCT}}}(Alloc\ \ell\ n;\ comp) = \underline{Alloc\ \ell\ n;🔓}}\ \text{(i-scct-filter-comp-alloc)}$$

$$\frac{\delta_{\mathsf{sCCT}}(\ell) = \underline{\ell}}{\theta_{\delta_{\mathsf{sCCT}}}(Dealloc\ \ell;\ comp) = \underline{Dealloc\ \ell;🔓}}\ \text{(i-scct-filter-comp-dealloc)} \qquad \frac{\delta_{\mathsf{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = n}{\theta_{\delta_{\mathsf{sCCT}}}(Get\ \ell\ n;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-get-noleak)}$$

$$\frac{\delta_{\mathsf{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = n}{\theta_{\delta_{\mathsf{sCCT}}}(Set\ \ell\ n\ v;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-set-noleak)} \qquad \frac{}{\theta_{\delta_{\mathsf{sCCT}}}(Call\ \mathbf{c}\ foo\ v;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-call)}$$

$$\frac{}{\theta_{\delta_{\mathsf{sCCT}}}(Ret\ \mathbf{c}\ v;\ comp) = \underline{\varepsilon}}\ \text{(i-scct-filter-comp-ret)} \qquad \frac{}{\theta_{\delta_{\mathsf{sCCT}}}(↯;t) = \underline{↯;🔓}}\ \text{(i-scct-filter-abort)} \qquad \frac{}{\theta_{\delta_{\mathsf{sCCT}}}(\varepsilon) = \underline{\varepsilon}}\ \text{(i-scct-filter-emptyevent)}$$

$\boxed{\theta^*_{\delta_{\mathsf{sCCT}}}(\overline{a}) = \overline{a^{\mathrm{scct}}}}$ „Project an L trace to specification traces."

$$\frac{}{\theta^*_{\delta_{\mathsf{sCCT}}}([\cdot]) = \underline{[\cdot]}}\ \text{(i-scct-filter-empty)} \qquad \frac{\theta_{\delta_{\mathsf{sCCT}}}(a) = \underline{a} \quad \theta^*_{\delta_{\mathsf{sCCT}}}(\overline{a}) = \overline{a^{\mathrm{scct}}} \quad \underline{a} \neq \underline{\varepsilon}}{\theta^*_{\delta_{\mathsf{sCCT}}}(a \cdot \overline{a}) = \underline{a} \cdot \overline{a^{\mathrm{scct}}}}\ \text{(i-scct-filter-cons-relevant)}$$

$$\frac{\theta_{\delta_{\mathsf{sCCT}}}(a) = \underline{\varepsilon} \quad \theta^*_{\delta_{\mathsf{sCCT}}}(\overline{a}) = \overline{a^{\mathrm{scct}}}}{\theta^*_{\delta_{\mathsf{sCCT}}}(a \cdot \overline{a}) = \overline{a^{\mathrm{scct}}}}\ \text{(i-scct-filter-cons-relevant)}$$

Figure 63: Projection of L events to specification events.

Lemma 75 (Injective Compiler (for final expressions))**.** *If*

(a) $[\![\mathbf{f}_{↯}]\!]^{\mathsf{L}_{ms} \to \mathsf{L}} = e_0$

(b) $[\![\mathbf{f}_{↯}]\!]^{\mathsf{L}_{ms} \to \mathsf{L}} = e_1$

Then

(i) $e_0 = e_1$

Proof. Case analysis on f. □

Lemma 76 (Subst $[\![\bullet]\!]^{\mathsf{L}_{ms} \to \mathsf{L}}$ Compatibility)**.**

(i) $[\![\mathbf{e}\gamma]\!]^{\mathsf{L}_{ms} \to \mathsf{L}} = [\![\mathbf{e}]\!]^{\mathsf{L}_{ms} \to \mathsf{L}} [\![\gamma]\!]^{\mathsf{L}_{ms} \to \mathsf{L}}$

Proof. Induction on $\mathbf{e}$. $\square$

Lemma 77 (Prim. Forward Simulation)**.** *If*

(a) $\boldsymbol{\Omega} \triangleright \mathbf{e}\gamma \xrightarrow{\mathbf{a}} \boldsymbol{\Omega}' \triangleright \mathbf{e}'\gamma'$

(b) $\boldsymbol{\Omega} = \Omega = \boldsymbol{\Psi}; \mathbf{comp}; \boldsymbol{\Phi}$

(c) $\vdash T_{SMS} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{SMS}'$

(d) $T_{SMS} \simeq_\delta \boldsymbol{\Omega}$

(e) $\theta_\delta(\mathbf{a}) \cong \boldsymbol{a}$

(f) $\mathbf{a} = a$

(g) $\boldsymbol{\Omega}' = \Omega'$

(h) $\Omega.\Delta \Vdash \gamma$

(i) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\Omega \triangleright [\![\mathbf{e}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}\gamma \xrightarrow{a}{}^{*}_{ctx} \Omega' \triangleright [\![\mathbf{e}']\!]^{\mathrm{L}_{ms}\to\mathrm{L}}\gamma'$

(ii) $T_{SMS}' \simeq_\delta \boldsymbol{\Omega}'$

(iii) $\Omega'.\Delta \Vdash \gamma'$

(iv) $\gamma' \prec \gamma'$

Proof. Induction on Assumption (a).

Case $e - \oplus$**:** If

(H_1) $n_3 = n_1 \oplus n_2$

(H_2) $\gamma = \gamma' = [\cdot]$

(H_3) $\boldsymbol{\Omega} \triangleright \mathbf{n_1} + \mathbf{n_2} \xrightarrow{\varepsilon} \boldsymbol{\Omega} \triangleright \mathbf{n_3}$

(H_4) $\boldsymbol{\Omega} = \Omega = \boldsymbol{\Psi}; \mathbf{comp}; \boldsymbol{\Phi}$

(H_5) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}'$

(H_6) $T_{\mathrm{SMS}} \simeq_\delta \boldsymbol{\Omega}$

(H_7) $\varepsilon = a$

(H_8) $\theta_\delta(\varepsilon) \cong \boldsymbol{a}$

(H_9) $\Omega.\Delta \Vdash \gamma$

(H_{10}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\Omega \triangleright [\![\mathbf{n_1} \oplus \mathbf{n_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma \xrightarrow{a}{}^{*}_{\mathrm{ctx}} \Omega \triangleright [\![\mathbf{n_3}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma'$

(ii) $T_{\mathrm{SMS}}' \simeq_\delta \boldsymbol{\Omega}$

(iii) $\Omega.\Delta \Vdash \gamma'$

(iv) $\gamma' \prec \gamma'$

Just instantiate $\gamma' = [\cdot]$ and note that no substitutions are applicable to $[\![\mathbf{n_1} \oplus \mathbf{n_2}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = n_1 \oplus n_2$ as well as $[\![\mathbf{n_3}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = n_3$. Induce on $\gamma \prec \gamma$ and see that no substitution changes the expression. Now, all goals follow immediately from assumptions.

Case $e - \mathbf{get} - \in$**:** If

(H_1) $\mathbf{\Psi} = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{m}), \mathbf{\Delta_2}$

(H_2) $\ell + \mathbf{n} \in \operatorname{dom} \mathbf{H^{comp}}$

(H_3) $\gamma = \gamma' = [\cdot]$

(H_4) $\mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{x}[\mathbf{n}] \xrightarrow{\mathbf{Get}\ \ell\ \mathbf{n};\mathbf{comp}} \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{H^{comp}}(\ell + \mathbf{n})$

(H_5) $\mathbf{\Omega} = \Omega = \mathbf{\Psi}; \mathbf{comp}; \mathbf{\Phi}$

(H_6) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}'$

(H_7) $T_{\mathrm{SMS}} \simeq_\delta \mathbf{\Omega}$

(H_8) $\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{comp} = a$

(H_9) $\theta_\delta\,(\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{comp}) \cong \boldsymbol{a}$

(H_{10}) $\Omega.\Delta \Vdash \gamma$

(H_{11}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\Omega \triangleright [\![\mathbf{x}[\mathbf{n}]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma \xrightarrow{a}{}^{*}_{\mathrm{ctx}} \Omega \triangleright [\![\mathbf{H^{comp}}(\ell + \mathbf{n})]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma'$

(ii) $T_{\mathrm{SMS}}' \simeq_\delta \mathbf{\Omega}$

(iii) $\Omega.\Delta \Vdash \gamma$

(iv) $\gamma' \prec \gamma'$

Instantiate $\gamma' = [\cdot]$.

Note that by inversion on Assumptions (H_6), (H_7) and (H_9) we have:

(H_1) $\boldsymbol{a} = \boldsymbol{Use}\ \underline{\ell}\ \boldsymbol{n}$

(H_2) $\mathbf{n} = \boldsymbol{n}$

(H_3) $(\underline{\ell}, \boldsymbol{m}) \in T_{\mathrm{SMS}}$

(H_4) $\boldsymbol{n} < \boldsymbol{m}$

(H_5) $\boldsymbol{m} = \mathbf{m}$

(H_6) $T_{\mathrm{SMS}}' = T_{\mathrm{SMS}}$

From that, it's easy to conclude $\mathbf{n} < \mathbf{m} = m$.

Note that $[\![\mathbf{x}[\mathbf{n}]]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = \mathit{let}\ x_{ACCESS} = [\![\mathbf{n}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in\ ifz}\ 0 \leq x_{ACCESS} \leq x_{SIZE}\ \mathit{then}\ x[x_{ACCESS}]\ \mathit{else\ abort}()$.

Induce on Assumption (H_{10}). The base case is spurious, since we have at least $x \mapsto (\ell; comp; \rho; m)$ in $\Omega.\Delta$. The inductive case splits into two other cases: Either we have a substitution with x_{SIZE} or not. If not, make use of the inductive hypothesis. If yes, then we know by definition that $x_{SIZE} = m$.

Thus, we can execute to the desired state Ω with just $\mathit{Get}\ \ell\ n; comp$. The other goals follow by assumption or are an immediate consequence.

Case $e - \mathbf{get} - \notin$**:** If

(H_1) $\mathbf{\Psi} = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{m}), \mathbf{\Delta_2}$

(H_2) $\gamma = \gamma' = [\cdot]$

(H_3) $\ell + \mathbf{n} \notin \mathrm{dom}\, \mathbf{H^{comp}}$

(H_4) $\mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{x}[\mathbf{n}] \xrightarrow{\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{comp}} \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{H^{comp}}(\ell + \mathbf{n})$

(H_5) $\mathbf{\Omega} = \mathit{\Omega} = \mathbf{\Psi}; \mathbf{comp}; \mathbf{\Phi}$

(H_6) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}'$

(H_7) $T_{\mathrm{SMS}} \simeq_\delta \mathbf{\Omega}$

(H_8) $\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{comp} = a$

(H_9) $\theta_\delta\,(\mathbf{Get}\ \ell\ \mathbf{n}; \mathbf{comp}) \cong \boldsymbol{a}$

(H_{10}) $\mathit{\Omega}.\mathit{\Delta} \Vdash \gamma$

(H_{11}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\mathit{\Omega} \triangleright [\![\mathbf{x}[\mathbf{n}]]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} \xrightarrow{a}{}^{*}_{\mathrm{ctx}} \mathit{\Omega} \triangleright [\![\mathbf{H^{comp}}(\ell + \mathbf{n})]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}$

(ii) $T_{\mathrm{SMS}}' \simeq_\delta \mathbf{\Omega}$

(iii) $\mathit{\Omega}.\mathit{\Delta} \Vdash \gamma'$

(iv) $\gamma' \prec \gamma'$

Immediate contradiction of Assumptions (H_3) and (H_7).

Case $e - \mathbf{set} - \mathit{comp}$**:** If

(H_1) $\mathbf{\Psi} = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{m}), \mathbf{\Delta_2}$

(H_2) $\mathbf{H^{comp}}' = \mathbf{H^{comp}}\,(\ell + \mathbf{n} \mapsto \mathbf{v})$

(H_3) $\mathbf{\Psi}' = \mathbf{H^{ctx}}; \mathbf{H^{comp}}'; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \rho; \mathbf{m}), \mathbf{\Delta_2}$

(H_4) $\gamma = \gamma' = [\cdot]$

(H_5) $\mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{x}[\mathbf{n}] \leftarrow \mathbf{v} \xrightarrow{\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}; \mathbf{comp}} \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}' \triangleright \mathbf{v}$

(H_6) $\mathbf{\Omega} = \mathit{\Omega} = \mathbf{\Psi}; \mathbf{comp}; \mathbf{\Phi}$

(H_7) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}'$

(H_8) $\theta_\delta\,(\mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}; \mathbf{comp}) \cong \boldsymbol{a}$

(H_9) $T_{\mathrm{SMS}} \simeq_\delta \mathbf{\Omega}$

(H_{10}) $\mathit{\Omega}' = \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}'$

(H_{11}) $a = \mathbf{Set}\ \ell\ \mathbf{n}\ \mathbf{v}; \mathbf{comp}$

(H_{12}) $\mathit{\Omega}.\mathit{\Delta} \Vdash \gamma$

(H_{13}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\mathit{\Omega} \triangleright [\![\mathbf{x}[\mathbf{n}] \leftarrow \mathbf{v}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} \xrightarrow{a}{}^{*}_{\mathrm{ctx}} \mathit{\Omega}' \triangleright [\![\mathbf{v}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}$

(ii) $T_{\mathrm{SMS}}' \simeq_\delta \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}'$

(iii) $\gamma' \Vdash \Omega.\Delta$

(iv) $\gamma' \prec \gamma'$

Instantiate $\gamma' = [\cdot]$.

Note that by inversion on Assumptions (H_7) to (H_9) we have:

(H_1) $\boldsymbol{a} = \boldsymbol{Use}\ \underline{\ell}\ \boldsymbol{n}$

(H_2) $\mathbf{n} = \boldsymbol{n}$

(H_3) $(\underline{\ell}, \boldsymbol{m}) \in T_{\text{SMS}}$

(H_4) $\boldsymbol{n} < \boldsymbol{m}$

(H_5) $\boldsymbol{m} = \mathbf{m}$

(H_6) $T_{\text{SMS}}{}' = T_{\text{SMS}}$

From that, it's easy to conclude $\mathbf{n} < \mathbf{m} = m$.

Note that $[\![\mathbf{x}[\mathbf{n}] \leftarrow \mathbf{v}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = let\ x_{ACCESS} = [\![\mathbf{n}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ in\ ifz\ 0 \leq x_{ACCESS} \leq x_{SIZE}\ then\ x[x_{ACCESS}] \leftarrow [\![\mathbf{v}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ else\ abort()$.

Induce on Assumption (H_{12}). The base case is spurious, since we have at least $x \mapsto (\ell; comp; \rho; m)$ in $\Omega.\Delta$. The inductive case splits into two other cases: Either we have a substitution with x_{SIZE} or not. If not, make use of the inductive hypothesis.If yes, then we know by definition that $x_{SIZE} = m$.

Thus, we can execute to the desired state Ω with just $Set\ \ell\ n\ v; comp$. The other goals follow by assumption or are an immediate consequence.

Case $e - \mathbf{new} - comp$**:** If

(H_1) $\mathbf{\Psi} = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta}$

(H_2) $\mathbf{\Delta} \vdash \ell\ fresh$

(H_3) $\mathbf{\Delta} \vdash \mathbf{z}\ fresh$

(H_4) $\mathbf{H^{comp}}{}' = \mathbf{H^{comp}} \ll \mathbf{n}$

(H_5) $\mathbf{\Psi}' = \mathbf{H^{ctx}}; \mathbf{H^{comp}}{}'; \mathbf{z} \mapsto (\ell; \mathbf{ctx}; \Box; \mathbf{n}), \mathbf{\Delta}$

(H_6) $\gamma = [\cdot]$

(H_7) $\gamma' = [\mathbf{z}/\mathbf{x}], [\cdot]$

(H_8) $\mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \mathbf{let\ x = new\ n\ in\ e} \xrightarrow{\mathbf{Alloc}\ \ell\ \mathbf{n;comp}} \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}' \triangleright \mathbf{e}[\mathbf{z}/\mathbf{x}]$

(H_9) $\mathbf{\Omega} = \Omega$

(H_{10}) $\vdash T_{\text{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{SMS}}{}'$

(H_{11}) $\theta_\delta\,(\mathbf{Alloc}\ \ell\ \mathbf{n}; \mathbf{comp}) \cong \boldsymbol{a}$

(H_{12}) $T_{\text{SMS}} \simeq_\delta \mathbf{\Omega}$

(H_{13}) $\Omega' = \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}'$

(H_{14}) $a = \mathbf{Alloc}\ \ell\ \mathbf{n}; \mathbf{comp}$

(H_{15}) $\Omega.\Delta \Vdash \gamma$

(H_{16}) $\gamma \prec \gamma$

Then $\exists\gamma'$,

(i) $\Omega \triangleright [\![\textbf{let x = new n in e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma \xrightarrow{a}{}^{*}_{\mathrm{ctx}} \Omega' \triangleright [\![\textbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma'$

(ii) $T_{\mathrm{SMS}}{}' \simeq_\delta \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}'$

(iii) $\gamma' \Vdash \Omega.\Delta$

(iv) $\gamma' \prec \gamma'$

Note that by inversion on Assumptions (H_7) to (H_9) we have:

(H_1) $\boldsymbol{a} = \boldsymbol{Alloc}\ \underline{\ell}\ \boldsymbol{n}$

(H_2) $\mathbf{n} = \boldsymbol{n}$

(H_3) $\underline{\ell} \notin \mathrm{dom}\, T_{\mathrm{SMS}}$

(H_4) $T_{\mathrm{SMS}} = (\underline{\ell}, \boldsymbol{n}) \cup T_{\mathrm{SMS}}$

Note that $[\![\textbf{let x = new n in e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = \begin{array}{l} \mathit{let}\ x_{SIZE} = [\![\mathbf{n}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\ \mathit{in} \\ \mathit{let}\ x = \mathit{new}\ x_{SIZE}\ \mathit{in}\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \end{array}$.

Instantiate $\gamma' = [[\![\mathbf{n}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}/x_{SIZE}], [z/x], \gamma$ and let $n = [\![\mathbf{n}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}$.

We make use of compatibility of substitution with the compiler: $[\![\textbf{let x = new n in e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma = \mathit{let}\ x_{SIZE} = n\ \mathit{in}\ \mathit{let}\ x = \mathit{new}\ n\ \mathit{in}\ ([\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma)$

Now observe that we can perform the steps: (abusing compatiblity of substitution on-the-fly)

- $\Omega \triangleright \mathit{let}\ x_{SIZE} = n\ \mathit{in}\ \mathit{let}\ x = \mathit{new}\ n\ \mathit{in}\ ([\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma) \xrightarrow{\varepsilon;comp}{}^{*}_{\mathrm{ctx}} \Omega \triangleright \mathit{let}\ x = \mathit{new}\ n\ \mathit{in}\ ([\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}[n/x_{SIZE}], \gamma)$
- $\Omega \triangleright \mathit{let}\ x = \mathit{new}\ n\ \mathit{in}\ ([\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}[z/x], [n/x_{SIZE}], \gamma) \xrightarrow{Alloc\ \ell\ n;comp}{}^{*}_{\mathrm{ctx}} \Omega' \triangleright [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}\gamma'$

So, it's easy to see that execute to the desired state Ω' with just $\mathit{Alloc}\ \ell\ n; comp$. The other goals are easy as well. For the last, make use of Rules subst-subset-cons and subst-subset-x_{SIZE}.

Case $e - \mathbf{new} - ctx$**:** Does not apply, since the sandboxtag is **comp**, but the rule requires **ctx**.

Case $e - \mathbf{set} - ctx$**:** Does not apply, since the sandboxtag is **comp**, but the rule requires **ctx**.

Case $e - \mathbf{delete}$**:** If

(H_1) $\mathbf{\Psi} = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; \square; \mathbf{m}), \mathbf{\Delta_2}$

(H_2) $\mathbf{\Psi}' = \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \mathbf{\Delta_1}, \mathbf{x} \mapsto (\ell; \mathbf{comp}; ☣; \mathbf{m}), \mathbf{\Delta_2}$

(H_3) $\gamma = \gamma' = [\cdot]$

(H_4) $\mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi} \triangleright \textbf{delete x} \xrightarrow{\mathbf{Dealloc}\ \ell;\mathbf{comp}} \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}' \triangleright \mathbf{0}$

(H_5) $\mathbf{\Omega} = \Omega = \mathbf{\Phi}; \mathbf{comp}; \mathbf{\Psi}$

(H_6) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}{}'$

(H_7) $T_{\mathrm{SMS}} \simeq_\delta \mathbf{\Omega}$

(H_8) $\theta_\delta\,(\mathbf{Dealloc}\ \ell; \mathbf{comp}) \cong \boldsymbol{a}$

(H_9) $\mathit{\Omega}' = \mathbf{\Omega}' = \mathbf{\Phi};\mathbf{comp};\mathbf{\Psi}'$

(H_{10}) $a = \mathbf{Dealloc}\ \ell;\mathbf{comp}$

(H_{11}) $\mathit{\Omega}.\mathit{\Delta} \Vdash \gamma$

(H_{12}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\mathit{\Omega} \triangleright [\![\mathbf{delete\ x}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} \xrightarrow{a}^{*}_{\mathrm{ctx}} \mathit{\Omega}' \triangleright [\![\mathbf{0}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}$

(ii) $T_{\mathrm{SMS}}' \simeq_{\delta} \mathbf{\Omega}'$

(iii) $\gamma' \Vdash \mathit{\Omega}.\mathit{\Delta}$

(iv) $\gamma' \prec \gamma'$

Just pick $\gamma' = [\cdot]$.

Since $[\![\mathbf{delete\ x}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} = \mathit{delete}\ x$, this case is entirely trivial up to an induction on Assumption (H_{11}).

Case $e - \mathbf{let} - \mathbf{f}$**:** If

(H_1) $\mathbf{\Omega} = \mathit{\Omega} = \mathbf{\Phi};\mathbf{comp};\mathbf{\Psi}$

(H_2) $\gamma = [\cdot]$

(H_3) $\gamma' = [\mathbf{v}/\mathbf{x}], [\cdot]$

(H_4) $\mathbf{\Omega} \triangleright \mathbf{let\ x = v\ in\ e} \xrightarrow{\varepsilon;\mathbf{comp}} \mathbf{\Omega} \triangleright \mathbf{e}[\mathbf{v}/\mathbf{x}]$

(H_5) $\vdash T_{\mathrm{SMS}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\mathrm{SMS}}'$

(H_6) $T_{\mathrm{SMS}} \simeq_{\delta} \mathbf{\Omega}$

(H_7) $\theta_{\delta}\,(\varepsilon;\mathbf{comp}) \cong \boldsymbol{a}$

(H_8) $a = \varepsilon;\mathbf{comp}$

(H_9) $\mathit{\Omega}.\mathit{\Delta} \Vdash \gamma$

(H_{10}) $\gamma \prec \gamma$

Then $\exists \gamma'$,

(i) $\mathit{\Omega} \triangleright [\![\mathbf{let\ x = v\ in\ e}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} \gamma \xrightarrow{a}^{*}_{\mathrm{ctx}} \mathit{\Omega} \triangleright [\![\mathbf{e}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} \gamma'$

(ii) $T_{\mathrm{SMS}} \simeq_{\delta} \mathbf{\Omega}$

(iii) $\gamma' \Vdash \mathit{\Omega}.\mathit{\Delta}$

(iv) $\gamma' \prec \gamma'$

Let $v = [\![\mathbf{v}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}$. Pick $\gamma = [v/x], [\cdot]$.

Note that $[\![\mathbf{let\ x = v\ in\ e}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}} = \mathit{let}\ x = [\![\mathbf{v}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}\ \mathit{in}\ [\![\mathbf{e}]\!]^{\mathrm{L_{ms}} \to \mathrm{L}}$.

Induce on Assumption (H_9). In the base-case, we can immediately step to the desired term and all goals go through trivially. The inductive cases go through easily.

Case $e - \mathbf{abort}$**:** This case and all following are entirely similar to case $e - \mathbf{delete}$ or $e - \oplus$.

Case $e - \mathbf{x\ is}$ ☣$-$yes**:**

Case $e - \mathbf{x}$ **is** ☣$-$no**:**

Case $e - \pi_1$**:**

Case $e - \pi_2$**:**

Case $e - \mathbf{x}$ **has** $\mathbb{N}$**:**

Case $e - \mathbf{n}$ **has** $\mathbb{N}$**:**

Case $e - pair -$ **has** $\mathbb{N}$**:**

Case $e - \mathbf{x}$ **has** $\mathbb{N} \times \mathbb{N}$**:**

Case $e - \mathbf{n}$ **has** $\mathbb{N} \times \mathbb{N}$**:**

Case $e - pair -$ **has** $\mathbb{N} \times \mathbb{N}$**:**

Case $e -$ **ifz**$-$true**:**

Case $e -$ **ifz**$-$false**:**

$\square$

Lemma 78 (Ctx. Forward Simulation)**.** *If*

(a) $\mathbf{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}_{ctx} \mathbf{\Omega}' \triangleright \mathbf{e}'$

(b) $\mathbf{\Omega} = \Omega = \Psi; comp; \Phi$

(c) $\vdash T_{SMS} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{SMS}'$

(d) $T_{SMS} \simeq_\delta \mathbf{\Omega}$

(e) $\theta_\delta\,(\mathbf{a}) \cong \boldsymbol{a}$

(f) $a = \mathbf{a}$

(g) $\mathbf{\Omega}' = \Omega'$

Then

(i) $\Omega \triangleright [\![\mathbf{e}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} \xrightarrow{a}{}^{*}_{ctx} \Omega' \triangleright [\![\mathbf{e}']\!]^{\mathrm{L}_{ms}\to\mathrm{L}}$

(ii) $T_{SMS}' \simeq_\delta \mathbf{\Omega}'$

Proof. Unfolding and using Lemma 77. $\square$

Lemma 79 (Forward Simulation)**.** *If*

(a) $\mathbf{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{ctx} \mathbf{\Omega}' \triangleright \mathbf{f}$

(b) $\mathbf{\Omega} = \Omega$

(c) $\overline{\mathbf{a}} = \overline{a}$

(d) $\mathbf{\Omega}' = \Omega'$

(e) $\vdash T_{SMS} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{SMS}'$

(f) $\theta_\delta^*(\overline{\mathrm{a}}) \cong \overline{\boldsymbol{a}}$

(g) $T_{SMS} \simeq_\delta \boldsymbol{\Omega}$

Then

(i) $\Omega \triangleright [\![\mathbf{e}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} \xrightarrow{\overline{a}}{}^*_{ctx} \Omega' \triangleright [\![\mathbf{f}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}$

(ii) $T_{SMS}' \simeq_\delta \boldsymbol{\Omega}'$

Proof. Proof is similar to Lemma 44 (Steps TMS via Monitor). □

Lemma 80 (Backward Simulation)**.** *If*

(a) $\Omega \triangleright [\![\mathbf{e}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} \xrightarrow{\overline{a}}{}^*_{ctx} \Omega' \triangleright [\![\mathbf{f}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}$

(b) $\boldsymbol{\Omega} = \Omega$

(c) $\overline{\mathbf{a}} = \overline{a}$

(d) $\boldsymbol{\Omega}' = \Omega'$

Then

(i) $\boldsymbol{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^*_{ctx} \boldsymbol{\Omega}' \triangleright \mathbf{f}$

Proof. Case analysis on wether $\exists \mathbf{f}', \boldsymbol{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^*_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}'$ holds or not.

Case $\exists \mathbf{f}', \boldsymbol{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^*_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}'$**:** By Lemma 79 (Forward Simulation) on what we assume in this case:

(H_1) $\Omega \triangleright [\![\mathbf{e}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \xrightarrow{\overline{a}}{}^*_{\mathrm{ctx}} \Omega' \triangleright [\![\mathbf{f}']\!]^{\mathrm{L_{ms}}\to\mathrm{L}}$

(H_2) $\overline{\mathbf{a}} = \overline{a}$

(H_3) $\boldsymbol{\Omega}' = \Omega'$

By Lemma 48 (Determinism of Steps):

(H_4) $[\![\mathbf{f}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} = [\![\mathbf{f}']\!]^{\mathrm{L_{ms}}\to\mathrm{L}}$

Subsequently, by Lemma 75 (Injective Compiler (for final expressions)):

(H_5) $\mathbf{f} = \mathbf{f}'$

Rewrite Assumption (H_1) using Assumption (H_5), solving our goal.

Case $\neg(\exists \mathbf{f}', \boldsymbol{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^*_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}')$**:** We can rewrite the assumption done in this case as:

(H_1) $\forall \mathbf{f}', \boldsymbol{\Omega} \triangleright \mathbf{e} \not\xrightarrow{\overline{\mathbf{a}}}{}^*_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}'$

We prove the goal by contradiction, so we have to prove $\perp$, assuming:

(H_2) $\boldsymbol{\Omega} \triangleright \mathbf{e} \xrightarrow{\overline{\mathbf{a}}}{}^*_{\mathrm{ctx}} \boldsymbol{\Omega}' \triangleright \mathbf{f}$

Assumption (H_2) contradicts Assumption (H_1).

□

Lemma 81 (Top-Level Backward Simulation). *If*

(a) $\Xi = \Xi_{ctx} \bowtie [\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}$

(b) $\Xi_{ctx} = \boldsymbol{\Xi}_{\mathbf{ctx}}$

(c) $\xi = [\![\xi]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} = \mathrm{dom}[\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}$

(d) $\xi; \Xi; [\cdot]\,; comp; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright call\ \ main\ \ 0 \xrightarrow{\overline{a}}{}^{*}_{ctx} \xi; \Xi; \overline{K}; ctx; H^{ctx}; H^{comp}; \Delta \triangleright f$

(e) $\xi = \xi$

Then $\exists \overline{\mathbf{a}}\ \overline{\mathbf{K}}\ \mathbf{H^{ctx}}\ \mathbf{H^{comp}}\ \boldsymbol{\Delta}$,

(i) $\xi; \boldsymbol{\Xi}; [\cdot]\,; \mathbf{comp}; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathbf{call\ main\ 0} \xrightarrow{\overline{\mathbf{a}}}{}^{*}_{ctx} \xi; \boldsymbol{\Xi}; \overline{\mathbf{K}}; \mathbf{ctx}; \mathbf{H^{ctx}}; \mathbf{H^{comp}}; \boldsymbol{\Delta} \triangleright \mathbf{f}$

(ii) $\overline{K} = \overline{\mathbf{K}}$

(iii) $H^{ctx} = \mathbf{H^{ctx}}$

(iv) $H^{comp} = \mathbf{H^{comp}}$

(v) $\Delta = \boldsymbol{\Delta}$

(vi) $\overline{a} = \overline{\mathbf{a}}$

Proof. Suppose control is never handled over to $[\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}$; that is, there is no $Call\ ?\ foo\ n$ in $\overline{a}$. In this case, the goals follow easily, since the semantics of $\mathrm{L_{ms}}$ and L are identical and $\Xi_{ctx} = \boldsymbol{\Xi}_{\mathbf{ctx}}$.

If control is handled over, we have, at some point in the reduction by transitivty, $\Omega_0 \triangleright [\![\mathbf{e_{foo}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}[v/x] \xrightarrow{\overline{a}}{}^{*}_{\mathrm{ctx}} \xi; \Xi; \overline{K}; ctx; H^{ctx}; H^{comp}; \Delta \triangleright f$. Thus, we can apply Lemma 80 (Backward Simulation) and are done. □

Lemma 82 (Program Backward Simulation). *If*

(a) $prog\ \ \Xi_{ctx}\ \ [\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} \xRightarrow{\overline{a}} \Omega \triangleright f_{\lightning}$

(b) $\Xi_{ctx} = \boldsymbol{\Xi}_{\mathbf{ctx}}$

(c) $\Omega = \boldsymbol{\Omega}$

(d) $\overline{a} = \overline{\mathbf{a}}$

(e) $f = \mathbf{f}$

Then

(i) $\mathbf{prog}\ \boldsymbol{\Xi}_{\mathbf{ctx}}\ \boldsymbol{\Xi}_{\mathbf{comp}} \xRightarrow{\overline{\mathbf{a}}} \boldsymbol{\Omega} \triangleright \mathbf{f}_{\lightning}$

Proof. Immediate consequence of Lemma 81 (Top-Level Backward Simulation). □

Theorem 4 (TMS Relation Correctness for $[\![\bullet]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}$).

$$\textit{if } \theta^{*}_{\delta_{MS}}\left(\sigma_{\theta^{*}_{\delta_{MS}}(\bullet)}(\mathrm{tms})\right) = \mathrm{tms}\ \ \textit{and } \sigma_{\approxeq^{*}_{\delta;\mathsf{X}}}(\sigma_{\theta^{*}_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{tms})) \approxeq^{*}_{\delta;\mathsf{X}} \sigma_{\theta^{*}_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{tms})$$

$$\textit{then } \theta^{*}_{\delta_{MS}}\left(\sigma_{\approxeq^{*}_{\delta;\mathsf{X}}}(\sigma_{\theta^{*}_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{tms}))\right) = \mathrm{tms}$$

Proof. unfolding. $\square$

Theorem 5 (SMS Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L_{ms}}\to\mathsf{L}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{sms})\right) = \mathrm{sms} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{sms})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{sms}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{sms}))\right) = \mathrm{sms} \end{aligned}$$

Proof. unfolding, works since memory accesses need to be in bounds. $\square$

Theorem 6 (sCCT Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L_{ms}}\to\mathsf{L}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{scct})\right) = \mathrm{scct} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{scct})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{scct}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{scct}))\right) = \mathrm{scct} \end{aligned}$$

Proof. unfolding. $\square$

Theorem 7 (SS Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L_{ms}}\to\mathsf{L}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{spec})\right) = \mathrm{spec} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{spec})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{spec}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{spec}))\right) = \mathrm{spec} \end{aligned}$$

Proof. unfolding. $\square$

Theorem 8 (SMS Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L}\to\mathsf{L_{scct}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathtt{MS}}}(\bullet)}(\mathrm{sms})\right) = \mathrm{sms} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{sms})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{sms}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{sms}))\right) = \mathrm{sms} \end{aligned}$$

Proof. unfolding. $\square$

Theorem 9 (sCCT Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L}\to\mathsf{L_{scct}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathtt{MS}}}(\bullet)}(\mathrm{scct})\right) = \mathrm{scct} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{scct})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{scct}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{scct}))\right) = \mathrm{scct} \end{aligned}$$

Proof. unfolding, works since there must not occur leaks according to the x-lang relation. $\square$

Theorem 10 (SS Relation Correctness for $\llbracket\bullet\rrbracket^{\mathsf{L}\to\mathsf{L_{scct}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathtt{MS}}}(\bullet)}(\mathrm{spec})\right) = \mathrm{spec} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\mathrm{spec})) \approxeq^*_{\delta;\mathsf{x}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{spec}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{x}}}(\sigma_{\theta^*_{\delta_{MS}}(\bullet)}(\mathrm{spec}))\right) = \mathrm{spec} \end{aligned}$$

Proof. unfolding. $\square$

Definition 43 (L Robust Satisfaction)**.** *We write* $\Xi_{ctx} \vDash_R \pi$ *for*

$$\forall \Xi_{comp}\ \overline{a}\ \Omega\ f_{\lightning},\, prog\ \Xi_{ctx}\ \Xi_{comp} \xRightarrow{\overline{a}} \Omega \triangleright f_{\lightning} \implies \exists \delta_{MS}, \theta_{\delta_{MS}}(\overline{a}) \in \pi$$

Theorem 11 (Robust SMS Preservation)**.**

(i) $\vdash [\![\bullet]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} : \lceil \mathrm{smsafe} \rceil$

Proof. Unfold Assumption (i), so if:

(H_1) $\pi \in \lceil \mathrm{smsafe} \rceil$

(H_2) $\boldsymbol{\Xi}_{\mathbf{comp}} \vDash_R \pi$

then

(i) $[\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L}_{\mathrm{ms}}\to\mathrm{L}} \vDash_R \pi$

Unfold Assumption (i): If

(H_3) $\mathit{prog}\ \Xi_{ctx}\ [\![\boldsymbol{\Xi}_{\mathbf{comp}}]\!]^{\mathrm{L}_{\mathrm{ms}}\to\mathrm{L}} \stackrel{\overline{a}}{\Longrightarrow} \Omega \triangleright f_{\lightning}$

then $\exists \delta_{MS}$

(i) $\theta_{\delta_{MS}}\,(\overline{a}) \in \pi$

Use Lemma 82 (Program Backward Simulation) on Assumption (H_3): (remember that $\overline{a} = \overline{\mathbf{a}}$, since $\mathrm{L} = \mathrm{L}_{\mathrm{ms}}$)

(H_4) $\mathbf{prog}\ \boldsymbol{\Xi}_{\mathbf{ctx}}\ \boldsymbol{\Xi}_{\mathbf{comp}} \stackrel{\overline{\mathbf{a}}}{\Longrightarrow} \boldsymbol{\Omega} \triangleright \mathbf{f}_{\lightning}$

Apply Assumption (H_2) to our goal.

Conclude with Assumption (H_4). □

Theorem 12 (Robust Memory Safety Preservation)**.**

(i) $\vdash [\![[\![\bullet]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}} : \lceil \mathrm{msafe} \rceil$

Proof. Note from **??** 3 (Robust TMS Preservation) and **??** 11 (Robust SMS Preservation):

(H_1) $\vdash [\![\bullet]\!]^{\mathrm{L}_{\mathrm{tms}}\to\mathrm{L}_{\mathrm{ms}}} : \lceil \mathrm{tmsafe} \rceil$

(H_2) $\vdash [\![\bullet]\!]^{\mathrm{L}_{\mathrm{ms}}\to\mathrm{L}} : \lceil \mathrm{smsafe} \rceil$

By Lemma 8 (Sequential Composition with RTP) using Assumptions (H_1) and (H_2), we have:

(H_1) $\vdash [\![[\![\bullet]\!]^{\mathrm{L}_{\mathrm{tms}}\to\mathrm{L}_{\mathrm{ms}}}]\!]^{\mathrm{L}_{\mathrm{ms}}\to\mathrm{L}} : \lceil \mathrm{tmsafe} \rceil \cap \lceil \mathrm{smsafe} \rceil$

This is exactly what we want to prove here, since by Definition 34, msafe $=$ tmsafe $\cap$ smsafe, noting that the powerset is closed under intersection. □

Proof (Alternative). By definition, if

(H_1) $\pi \in \lceil \mathrm{msafe} \rceil$

(H_2) $\Xi_{\mathsf{ctx}} \vDash_R \pi$

Then

(i) $[\![[\![\Xi_{\mathsf{ctx}}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \vDash_R \pi$

By Definition 34, we have msafe = tmsafe ∩ smsafe.
So, since powerset is closed under intersection, we can split Assumption (H_1) into:

(H_3) $\pi \in \lceil \text{tmsafe} \rceil$

(H_4) $\pi \in \lceil \text{smsafe} \rceil$

With Assumptions (H_2) and (H_3) we use **??** 3 (Robust TMS Preservation):

(H_5) $[\![\Xi_{\mathsf{ctx}}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}} \vDash_R \pi$

With Assumptions (H_4) and (H_5) we use **??** 11 (Robust SMS Preservation):

(H_6) $[\![[\![\Xi_{\mathsf{ctx}}]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} \vDash_R \pi$

This solves Goal (i). □

Theorem 13 (TMS Relation Correctness for $[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{tms})\right) = \text{tms} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{tms})) \approxeq^*_{\delta;\mathsf{X}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\text{tms}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{tms}))\right) = \text{tms} \end{aligned}$$

Proof. unfolding. □

Theorem 14 (SMS Relation Correctness for $[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{sms})\right) = \text{sms} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{sms})) \approxeq^*_{\delta;\mathsf{X}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\text{sms}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{sms}))\right) = \text{sms} \end{aligned}$$

Proof. unfolding. □

Theorem 15 (sCCT Relation Correctness for $[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{scct})\right) = \text{scct} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{scct})) \approxeq^*_{\delta;\mathsf{X}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\text{scct}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{scct}))\right) = \text{scct} \end{aligned}$$

Proof. unfolding. □

Theorem 16 (SS Relation Correctness for $[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}$)**.**

$$\begin{aligned} &\textit{if } \theta^*_{\delta_{MS}}\left(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{spec})\right) = \text{spec} \;\; \textit{and } \sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{spec})) \approxeq^*_{\delta;\mathsf{X}} \sigma_{\theta^*_{\delta_{\mathbf{MS}}}(\bullet)}(\text{spec}) \\ &\textit{then } \theta^*_{\delta_{MS}}\left(\sigma_{\approxeq^*_{\delta;\mathsf{X}}}(\sigma_{\theta^*_{\delta_{\mathsf{MS}}}(\bullet)}(\text{spec}))\right) = \text{spec} \end{aligned}$$

Proof. unfolding. □

4.6 Optimising Passes

4.6.1 DCE

$\boxed{[\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} = e'}$ „Compile L expression e to L expression e'."

$\boxed{[\![[v/x]]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} = [v/x]}$ „Compile L substitution to L substitution."

$\boxed{[\![\xi]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} = \xi'}$ „Compile L component library to L component library."

$$
\begin{aligned}
[\![f_{\frac{1}{2}}]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= f_{\frac{1}{2}} \\
[\![call\ foo\ e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= call\ [\![foo]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}\ [\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![return\ e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= return\ [\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![e_1 \oplus e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= [\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \oplus [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![x[e]]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= x[[\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}] \\
[\![let\ x = e_1\ in\ e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= let\ x = [\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}\ in\ [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![x[e_1] \leftarrow e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= x[[\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}] \leftarrow [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![let\ x = new\ e_1\ in\ e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= let\ x = new\ [\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}\ in\ [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![delete\ x]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= delete\ x \\
[\![x\ is\ ☣]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= x\ is\ ☣ \\
[\![(e_1; e_2)]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= ([\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}; [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}) \\
[\![\pi_1\ e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= \pi_1\ [\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![\pi_2\ e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= \pi_2\ [\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
[\![e\ has\ \tau]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= [\![e]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}\ has\ \tau \\
[\![ifz\ true\ then\ e_2\ else\ e_3]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= e_2 \\
[\![ifz\ true\ then\ e_3\ else\ e_3]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= e_3 \\
[\![ifz\ e_1\ then\ e_2\ else\ e_3]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= ifz\ [\![e_1]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
&\quad then\ [\![e_2]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
&\quad else\ [\![e_3]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} \\
\\
[\![[v/x]]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= [[\![v]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} / [\![x]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}] \\
\\
[\![[\cdot]]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= [\cdot] \\
[\![foo, \xi]\!]^{\mathrm{L}\to\mathrm{L}}_{dce} &= foo, [\![\xi]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}
\end{aligned}
$$

Figure 64: Compiler from L to L, performing dead code elimination.

Since the traces do not change after compiling with $[\![\bullet]\!]^{\mathrm{L}\to\mathrm{L}}_{dce}$, the security proof follows easily using a context-based backtranslation that is essentially just the identity. Anything else is similar to Section 4.5.

4.6.2 CF

$\boxed{[\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} = e'}$ „Compile L expression e to L expression e'."

$\boxed{[\![\xi, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} = \xi'}$ „Compile L component library to L component library."

$$
\begin{aligned}
[\![x, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= v \quad \text{if } [v \text{ for } x] \in \overline{\rho} \\
[\![x, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= x \quad \text{if } [v \text{ for } x] \notin \overline{\rho} \\
[\![f_{\lightning}, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= f_{\lightning} \\
[\![\mathit{call}\ \mathit{foo}\ e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{call}\ [\![\mathit{foo}, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}\ [\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![\mathit{return}\ e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{return}\ [\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![n \oplus m, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= k \quad \text{where } k = n \oplus m \\
[\![e_1 \oplus e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= [\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \oplus [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![x[e], \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= x[[\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}] \\
[\![\mathit{let}\ x = v\ \mathit{in}\ e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= [\![e_2, [v \text{ for } x] \cdot \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![\mathit{let}\ x = e_1\ \mathit{in}\ e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{let}\ x = [\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}\ \mathit{in}\ [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![x[e_1] \leftarrow e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= x[[\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}] \leftarrow [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![\mathit{let}\ x = \mathit{new}\ e_1\ \mathit{in}\ e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{let}\ x = \mathit{new}\ [\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}\ \mathit{in}\ [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![\mathit{delete}\ x, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{delete}\ x \\
[\![x\ \mathit{is}\ ☣, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= x\ \mathit{is}\ ☣ \\
[\![(e_1; e_2), \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= ([\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}; [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}) \\
[\![\pi_1\ e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \pi_1\ [\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![\pi_2\ e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \pi_2\ [\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
[\![e\ \mathit{has}\ \tau, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= [\![e, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}\ \mathit{has}\ \tau \\
[\![\mathit{ifz}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{ifz}\ [\![e_1, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
&\quad\ \mathit{then}\ [\![e_2, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
&\quad\ \mathit{else}\ [\![e_3, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} \\
\\
[\![[\cdot], \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= [\cdot] \\
[\![\mathit{foo}, \xi, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}} &= \mathit{foo}, [\![\xi, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}
\end{aligned}
$$

Figure 65: Compiler from L to L, performing constant folding.

Since the traces do not change after compiling with $[\![\bullet, \overline{\rho}]\!]_{cf}^{\mathrm{L}\to\mathrm{L}}$, the security proof follows easily using a context-based backtranslation that is essentially just the identity. Anything else is similar to Section 4.5.

4.7 Object Language

$$
\begin{array}{rl}
\textit{Final Result } \texttt{f} ::= \texttt{v}^{\sigma} \mid \texttt{x}^{\sigma} & \textit{May be a Result } \texttt{f}_{\not{\,}} ::= \texttt{f} \mid \texttt{stuck} \\
\textit{Expressions } \texttt{e} ::= & \texttt{f}_{\not{\,}} \mid \texttt{e}_1 \oplus \texttt{e}_2 \mid \texttt{x}[\texttt{e}] \mid \texttt{getDIT x in e} \mid \texttt{setDIT e} \\
& \mid \texttt{let x} = \texttt{e}_1 \texttt{ in } \texttt{e}_2 \mid \texttt{x}[\texttt{e}_1] \leftarrow \texttt{e}_2 \\
& \mid \texttt{let x} = \texttt{new } \texttt{e}_1 \texttt{ in } \texttt{e}_2 \mid \texttt{delete x} \\
& \mid \texttt{return e} \mid \texttt{call foo e} \mid \texttt{ifz } \texttt{e}_1 \texttt{ then } \texttt{e}_2 \texttt{ else } \texttt{e}_3 \\
& \mid \texttt{abort}() \mid \texttt{x is } \text{☣} \\
& \mid \langle \texttt{e}_1, \texttt{e}_2 \rangle \mid \pi_1 \texttt{e} \mid \pi_2 \texttt{e} \mid \texttt{e has } \tau \\
& \text{where } \oplus \in \{+, -, \times, <, /\} \\
\textit{Functions } \texttt{F} ::= \texttt{let foo x} := \texttt{e} & \textit{Types } \tau ::= \mathbb{N} \mid \mathbb{N} \times \mathbb{N} \\
\textit{Values } \texttt{v} ::= \texttt{n} \in \mathbb{N} & \textit{References } \ell \in \mathbb{N} \\
\textit{Eval.Ctx. } \texttt{K} ::= & [\cdot] \mid \texttt{K} \oplus \texttt{e} \mid \texttt{v} \oplus \texttt{K} \mid \texttt{x}[\texttt{K}] \mid \texttt{let x} = \texttt{K in e} \\
& \mid \texttt{x}[\texttt{K}] \leftarrow \texttt{e} \mid \texttt{x}[\texttt{v}] \leftarrow \texttt{K} \mid \texttt{let x} = \texttt{new K in e} \\
& \mid \texttt{ifz K then } \texttt{e}_1 \texttt{ else } \texttt{e}_2 \mid \texttt{call foo K} \mid \texttt{return K} \\
& \mid \langle \texttt{K}, \texttt{e} \rangle \mid \langle \texttt{v}, \texttt{K} \rangle \mid \pi_1 \texttt{K} \mid \pi_2 \texttt{K} \mid \texttt{K has } \tau \\
& \mid \texttt{ifz K then } \texttt{e}_1 \texttt{ else } \texttt{e}_2 \mid \texttt{setDIT K} \\
\textit{Variables } \texttt{x} \mid \texttt{y} \mid \texttt{foo} \mid \ldots & \textit{Poison } \rho ::= \square \mid \text{☣} \\
\textit{Sandbox Tag } \texttt{t} ::= \texttt{ctx} \mid \texttt{comp} & \\
\textit{Typing.Env. } \Gamma ::= [\cdot] \mid \Gamma, \texttt{x} : \tau & \textit{Store } \Delta ::= [\cdot] \mid \texttt{x} \mapsto (\ell; \texttt{t}; \rho; \texttt{n}), \Delta \\
\textit{Communication } \mathbf{c} ::= ? \mid ! \mid \varnothing & \textit{Heaps } \texttt{H} ::= [\cdot] \mid \texttt{H} :: \texttt{n} \\
\textit{Cont. Stack } \overline{\texttt{K}} ::= [\cdot] \mid (\texttt{K}; \texttt{foo}), \overline{\texttt{K}} & \textit{Library } \Xi ::= [\cdot] \mid \texttt{F}, \Xi \\
\textit{Relevant } \xi ::= [\cdot] \mid \texttt{foo}, \xi & \textit{State } \Omega ::= \Phi; \texttt{t}; \texttt{n}; \Psi \\
\textit{Flow State } \Phi ::= \xi; \Xi; \overline{\texttt{K}} & \textit{Memory State } \Psi ::= \texttt{H}^{\texttt{ctx}}; \texttt{H}^{\texttt{comp}}; \Delta \\
\textit{Programs } \texttt{prog } \Xi_{\texttt{ctx}} \ \Xi_{\texttt{comp}} & \textit{Substitutions } \gamma ::= [\texttt{v}/\texttt{x}], \gamma \mid [\cdot] \\
\textit{Security Tag } \sigma ::= \text{🔒} \mid \text{🔓} &
\end{array}
$$

Figure 66: Syntax of $\mathrm{L}_{\mathrm{scct}}$

The only difference between L and $\mathrm{L}_{\mathrm{scct}}$ is that we have access to model specific registers by means of the `getDIT` and `setDIT` instructions. Thus, we also extend the dynamic state with a special purpose register representing a model specific register for a data operand independent timing mode). Whenever this flag is set, certain instructions will run in a „constant-time mode". Usually, access to model specific registers requires privileged execution. Since we want to model the strongest possible attacker, we disregard privilege and, thus, grant contexts access to that variable.

The language also has security tags to model annotations to variables (and, by taint analysis, to expressions) as high or low. 🔒 means it's high, while 🔓 means it's low. We have an $\cdot \leq \cdot$ such that $\text{🔒} \leq \text{🔓}$. Final results are tagged with the respected taint and the semantics keeps track of this taint during execution.

4.7.1 Dynamic Semantics

$\boxed{\text{dom}\ \Xi = \texttt{foo}, \ldots, \texttt{bar}}$ „Collect function names."

$$\frac{}{\text{dom}\ [\cdot] = [\cdot]}\ (\Xi\text{-dom-}[\cdot]) \qquad \frac{\text{dom}\ \Xi = \texttt{D}}{\text{dom}\ (\texttt{let foo x} : \tau_\lambda := \texttt{e}), \Xi = \texttt{foo}, \texttt{D}}\ (\Xi\text{-dom-cons})$$

$\boxed{\Xi \equiv \Xi_1 \cup \Xi_2}$ „Merging L_{scct} libraries."

$$\frac{\Xi = \Xi_1}{\Xi \equiv \Xi_1 \cup [\cdot]}\ (\text{lib-merge-empty}) \qquad \frac{\Xi \equiv \Xi_1 \cup \Xi_2}{\texttt{F}, \Xi \equiv \Xi_1 \cup \texttt{F}, \Xi_2}\ (\text{lib-merge-cons})$$

$\boxed{\Xi_1 \bowtie \Xi_2 = \Xi}$ „Syntactically linking L_{scct} libraries."

$$\frac{\text{dom}\,\Xi_1 \cap \text{dom}\,\Xi_2 = \emptyset \qquad \Xi \equiv \Xi_1 \cup \Xi_2}{\Xi_1 \bowtie \Xi_2 = \Xi}\ (\text{syntactic-plugging})$$

Figure 67: L_{scct} plugging of libraries and collecting of function names.

$$\begin{aligned}
\textit{Base Events}\ \texttt{a}_\texttt{b} ::= {} & \texttt{Alloc}\ \ell\ \texttt{v} \mid \texttt{Dealloc}\ \ell \mid \texttt{Get}\ \ell\ \texttt{v} \mid \texttt{Set}\ \ell\ \texttt{v}\ \texttt{v}' \mid \lightning \\
& \mid \texttt{Call}\ \mathbf{c}\ \texttt{foo}\ \texttt{v} \mid \texttt{Ret}\ \mathbf{c}\ \texttt{v} \mid \texttt{Start} \mid \texttt{End}\ \texttt{v} \\
& \mid \texttt{Branch}\ \texttt{n} \mid \texttt{Binop}\ \texttt{n}\ \texttt{m} \\
& \mid \texttt{iGet}\ \ell\ \texttt{v} \mid \texttt{iSet}\ \ell\ \texttt{v}\ \texttt{v}' \\
\textit{Events}\ \texttt{a} ::= {} & \varepsilon \mid \texttt{a}_\texttt{b}; \texttt{t}; \sigma
\end{aligned}$$

Figure 68: Events of L_{scct}.

$\boxed{\cdot \leq \cdot}$ „Order of security labels."

$$\frac{}{🔓 \leq 🔓}\ (\text{leq-🔓-refl}) \qquad \frac{}{🔒 \leq \sigma}\ (\text{leq-lock})$$

$\boxed{\cdot \sqcap \cdot}$ „Meet of security labels."

$$\frac{}{\sigma \sqcap \sigma = \sigma}\ (\text{meet-refl}) \qquad \frac{\sigma \neq \sigma'}{\sigma \sqcap \sigma' = 🔓}\ (\text{meet-ineq})$$

Figure 69: Lattice operations for security labels.

$\boxed{\Omega \triangleright \mathtt{e} \xrightarrow{\mathtt{a}} \Omega' \triangleright \mathtt{e}'}$ „Expression $\mathtt{e}$ evaluates under Configuration Ω to $\mathtt{e}'$ and new Configuration Ω', emitting event $\mathtt{a}$."

$$\frac{\mathtt{n}_1 \oplus \mathtt{n}_2 = \mathtt{n}_3 \qquad \oplus \notin \{*, /, <\} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Omega \triangleright \mathtt{n}_1^{\sigma} \oplus \mathtt{n}_2^{\sigma'} \xrightarrow{\varepsilon} \Omega \triangleright \mathtt{n}_3^{\sigma''}} \ (e - \oplus - \text{noleak-op})$$

$$\frac{\mathtt{n}_1 \oplus \mathtt{n}_2 = \mathtt{n}_3 \qquad \oplus \in \{*, /, <\} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}; 0; \Psi \triangleright \mathtt{n}_1^{\sigma} \oplus \mathtt{n}_2^{\sigma'} \xrightarrow{\mathtt{Binop}\ \mathtt{n}_1\ \mathtt{n}_2;\mathtt{t};\sigma''} \Phi; \mathtt{t}; 0; \Psi \triangleright \mathtt{n}_3^{\sigma''}} \ (e - \oplus - \text{leak})$$

$$\frac{\mathtt{n}_1 \oplus \mathtt{n}_2 = \mathtt{n}_3 \qquad \oplus \in \{*, /, <\} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}; \mathtt{m}+1; \Psi \triangleright \mathtt{n}_1^{\sigma} \oplus \mathtt{n}_2^{\sigma'} \xrightarrow{\varepsilon} \Phi; \mathtt{t}; \mathtt{m}+1; \Psi \triangleright \mathtt{n}_3^{\sigma''}} \ (e - \oplus - \text{noleak-doit})$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \rho; \mathtt{m}), \Delta_2 \qquad \ell + \mathtt{n} \in \text{dom } \mathtt{H}^{\mathtt{t}} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}'; 0; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \xrightarrow{\mathtt{Get}\ \ell\ \mathtt{n};\mathtt{t};\sigma''} \Phi; \mathtt{t}'; 0; \Psi \triangleright (\mathtt{H}^{\mathtt{t}}(\ell + \mathtt{n}))^{\sigma''}} \ (e - \mathtt{get} - \in - \text{leak})$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \rho; \mathtt{m}), \Delta_2 \qquad \ell \notin \text{dom } \mathtt{H}^{\mathtt{t}} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}'; 0; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \xrightarrow{\mathtt{Get}\ \ell\ \mathtt{n};\mathtt{t};\sigma''} \Phi; \mathtt{t}'; 0; \Psi \triangleright 1729} \ (e - \mathtt{get} - \notin - \text{leak})$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \rho; \mathtt{m}), \Delta_2 \qquad \ell + \mathtt{n} \in \text{dom } \mathtt{H}^{\mathtt{t}} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}'; \mathtt{m}+1; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \xrightarrow{\mathtt{iGet}\ \ell\ \mathtt{n};\mathtt{t};\sigma''} \Phi; \mathtt{t}'; \mathtt{m}+1; \Psi \triangleright (\mathtt{H}^{\mathtt{t}}(\ell + \mathtt{n}))^{\sigma''}} \ (e - \mathtt{get} - \in - \text{noleak})$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \rho; \mathtt{m}), \Delta_2 \qquad \ell \notin \text{dom } \mathtt{H}^{\mathtt{t}} \qquad \sigma'' = \sigma \sqcap \sigma'}{\Phi; \mathtt{t}'; \mathtt{m}+1; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \xrightarrow{\mathtt{iGet}\ \ell\ \mathtt{n};\mathtt{t};\sigma''} \Phi; \mathtt{t}'; \mathtt{m}+1; \Psi \triangleright 1729} \ (e - \mathtt{get} - \notin - \text{noleak})$$

$$\frac{\begin{array}{ll} \Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{ctx}; \rho; \mathtt{n}), \Delta_2 & \mathtt{H}^{\mathtt{ctx}'} = \mathtt{H}^{\mathtt{ctx}}(\ell + \mathtt{n} \mapsto \mathtt{v}) \\ \Psi' = \mathtt{H}^{\mathtt{ctx}'}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{ctx}; \rho; \mathtt{n}), \Delta_2 & \sigma''' = \sigma \sqcap \sigma' \end{array}}{\Phi; \mathtt{ctx}; 0; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \leftarrow \mathtt{v}^{\sigma''} \xrightarrow{\mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{v};\mathtt{ctx};\sigma'''} \Phi; \mathtt{ctx}; 0; \Psi' \triangleright \mathtt{v}^{\sigma''}} \ (e - \mathtt{set} - ctx - \text{leak})$$

$$\frac{\begin{array}{ll} \Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{comp}; \rho; \mathtt{n}), \Delta_2 & \mathtt{H}^{\mathtt{comp}'} = \mathtt{H}^{\mathtt{comp}}(\ell + \mathtt{n} \mapsto \mathtt{v}) \\ \Psi' = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}'}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{comp}; \rho; \mathtt{n}), \Delta_2 & \sigma''' = \sigma \sqcap \sigma' \end{array}}{\Phi; \mathtt{comp}; 0; \Psi \triangleright \mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}] \leftarrow \mathtt{v}^{\sigma''} \xrightarrow{\mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{v};\mathtt{comp};\sigma'''} \Phi; \mathtt{comp}; 0; \Psi' \triangleright \mathtt{v}^{\sigma''}} \ (e - \mathtt{set} - comp - \text{leak})$$

Figure 70: Evaluation of $\mathtt{L}_{\mathtt{scct}}$ expressions.

$$(e-\texttt{set}-ctx\text{—noleak})$$
$$\frac{\begin{array}{cc}\Psi=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{ctx};\rho;\mathtt{n}),\Delta_2 & \mathtt{H}^{\mathtt{ctx}'}=\mathtt{H}^{\mathtt{ctx}}(\ell+\mathtt{n}\mapsto\mathtt{v})\\ \Psi'=\mathtt{H}^{\mathtt{ctx}'};\mathtt{H}^{\mathtt{comp}};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{ctx};\rho;\mathtt{n}),\Delta_2 & \sigma'''=\sigma\sqcap\sigma'\end{array}}{\Phi;\mathtt{ctx};\mathtt{m}+1;\Psi\triangleright\mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}]\leftarrow\mathtt{v}^{\sigma''}\xrightarrow{\mathtt{iSet}\ \ell\ \mathtt{n}\ \mathtt{v};\mathtt{ctx};\sigma'''}\Phi;\mathtt{ctx};\mathtt{m}+1;\Psi'\triangleright\mathtt{v}^{\sigma''}}$$

$$(e-\texttt{set}-comp\text{—noleak})$$
$$\frac{\begin{array}{cc}\Psi=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{comp};\rho;\mathtt{n}),\Delta_2 & \mathtt{H}^{\mathtt{comp}'}=\mathtt{H}^{\mathtt{comp}}(\ell+\mathtt{n}\mapsto\mathtt{v})\\ \Psi'=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}'};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{comp};\rho;\mathtt{n}),\Delta_2 & \sigma'''=\sigma\sqcap\sigma'\end{array}}{\Phi;\mathtt{comp};\mathtt{m}+1;\Psi\triangleright\mathtt{x}^{\sigma}[\mathtt{n}^{\sigma'}]\leftarrow\mathtt{v}^{\sigma''}\xrightarrow{\mathtt{iSet}\ \ell\ \mathtt{n}\ \mathtt{v};\mathtt{comp};\sigma'''}\Phi;\mathtt{comp};\mathtt{m}+1;\Psi'\triangleright\mathtt{v}^{\sigma''}}$$

$$(e-\texttt{let}-\texttt{f})$$
$$\frac{\sigma''=\sigma\sqcap\sigma'}{\Omega\triangleright\mathtt{let}\ \mathtt{x}^{\sigma}=\mathtt{f}^{\sigma'}\ \mathtt{in}\ \mathtt{e}\xrightarrow{\varepsilon}\Omega\triangleright\mathtt{e}[\mathtt{f}^{\sigma''}/\mathtt{x}]}$$

$$(e-\texttt{delete})$$
$$\frac{\begin{array}{c}\Psi=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{t};\square;\mathtt{n}),\Delta_2\\ \Psi=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta_1,\mathtt{x}\mapsto(\ell;\mathtt{t};\text{☣};\mathtt{n}),\Delta_2\end{array}}{\Phi;\mathtt{t};\mathtt{m};\Psi\triangleright\mathtt{delete}\ \mathtt{x}^{\sigma}\xrightarrow{\mathtt{Dealloc}\ \ell;\mathtt{t};\sigma}\Phi;\mathtt{t};\mathtt{m};\Psi'\triangleright 0^{\text{🔓}}}$$

$$(e-\texttt{new}-ctx)$$
$$\frac{\begin{array}{ccc}\Delta\vdash\ell\ \mathit{fresh} & \Delta\vdash\mathtt{z}\ \mathit{fresh} & \mathtt{H}^{\mathtt{ctx}'}=\mathtt{H}^{\mathtt{ctx}}\ll\mathtt{n}\\ \multicolumn{2}{c}{\Psi=\mathtt{H}^{\mathtt{ctx}'};\mathtt{H}^{\mathtt{comp}};\mathtt{z}\mapsto(\ell;\mathtt{ctx};\square;\mathtt{n}),\Delta} & \sigma''=\sigma\sqcap\sigma'\end{array}}{\Phi;\mathtt{ctx};\mathtt{m};\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta\triangleright\mathtt{let}\ \mathtt{x}^{\sigma}=\mathtt{new}\ \mathtt{n}^{\sigma'}\ \mathtt{in}\ \mathtt{e}\xrightarrow{\mathtt{Alloc}\ \ell\ \mathtt{n};\mathtt{ctx};\sigma''}\Phi;\mathtt{ctx};\mathtt{m};\Psi\triangleright\mathtt{e}[\mathtt{z}^{\sigma''}/\mathtt{x}]}$$

$$(e-\texttt{new}-comp)$$
$$\frac{\begin{array}{ccc}\Delta\vdash\ell\ \mathit{fresh} & \Delta\vdash\mathtt{z}\ \mathit{fresh} & \mathtt{H}^{\mathtt{comp}'}=\mathtt{H}^{\mathtt{comp}}\ll\mathtt{n}\\ \multicolumn{2}{c}{\Psi=\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}'};\mathtt{z}\mapsto(\ell;\mathtt{ctx};\square;\mathtt{n}),\Delta} & \sigma''=\sigma\sqcap\sigma'\end{array}}{\Phi;\mathtt{comp};\mathtt{m};\mathtt{H}^{\mathtt{ctx}};\mathtt{H}^{\mathtt{comp}};\Delta\triangleright\mathtt{let}\ \mathtt{x}^{\sigma}=\mathtt{new}\ \mathtt{n}^{\sigma'}\ \mathtt{in}\ \mathtt{e}\xrightarrow{\mathtt{Alloc}\ \ell\ \mathtt{n};\mathtt{comp};\sigma''}\Phi;\mathtt{comp};\mathtt{m};\Psi\triangleright\mathtt{e}[\mathtt{z}^{\sigma''}/\mathtt{x}]}$$

$$(e-\texttt{ifz}\text{—true-noleak})$$
$$\frac{}{\Phi;\mathtt{t};0;\Psi\triangleright\mathtt{ifz}\ 0^{\sigma}\ \mathtt{then}\ \mathtt{e}_1\ \mathtt{else}\ \mathtt{e}_2\xrightarrow{\mathtt{Branch}\ 0;\mathtt{t};\sigma}\Phi;\mathtt{t};0;\Psi\triangleright\mathtt{e}_1}$$

$$(e-\texttt{ifz}\text{—false-noleak})$$
$$\frac{}{\Phi;\mathtt{t};0;\Psi\triangleright\mathtt{ifz}\ (\mathtt{S}\ \mathtt{n})^{\sigma}\ \mathtt{then}\ \mathtt{e}_1\ \mathtt{else}\ \mathtt{e}_2\xrightarrow{\mathtt{Branch}\ \mathtt{S(n)};\mathtt{t};\sigma}\Phi;\mathtt{t};0;\Psi\triangleright\mathtt{e}_2}$$

$$(e-\texttt{ifz}\text{—true-leak})$$
$$\frac{}{\Phi;\mathtt{t};\mathtt{m}+1;\Psi\triangleright\mathtt{ifz}\ 0^{\sigma}\ \mathtt{then}\ \mathtt{e}_1\ \mathtt{else}\ \mathtt{e}_2\xrightarrow{\varepsilon}\Phi;\mathtt{t};\mathtt{m}+1;\Psi\triangleright\mathtt{e}_1}$$

$$(e-\texttt{ifz}\text{—false-leak})$$
$$\frac{}{\Phi;\mathtt{t};\mathtt{m}+1;\Psi\triangleright\mathtt{ifz}\ (\mathtt{S}\ \mathtt{n})^{\sigma}\ \mathtt{then}\ \mathtt{e}_1\ \mathtt{else}\ \mathtt{e}_2\xrightarrow{\varepsilon}\Phi;\mathtt{t};\mathtt{m}+1;\Psi\triangleright\mathtt{e}_2}$$

$$(e-\texttt{abort})$$
$$\frac{}{\Phi;\mathtt{t};\mathtt{m};\Psi\triangleright\mathtt{abort}()\xrightarrow{\text{↯};\mathtt{t};\text{🔓}}\Phi;\mathtt{t};\mathtt{m};\Psi\triangleright\mathtt{stuck}}$$

Figure 71: Evaluation of $\mathrm{L}_{\mathrm{scct}}$ expressions, continued.

$$\frac{}{\Omega \triangleright \pi_1 \langle \mathtt{n}_1^{\sigma}; \mathtt{n}_2^{\sigma'} \rangle \xrightarrow{\varepsilon} \Omega \triangleright \mathtt{n}_1^{\sigma}}\ (e - \pi_1) \qquad \frac{}{\Omega \triangleright \pi_2 \langle \mathtt{n}_1^{\sigma}; \mathtt{n}_2^{\sigma'} \rangle \xrightarrow{\varepsilon} \Omega \triangleright \mathtt{n}_2^{\sigma'}}\ (e - \pi_2)$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta \qquad \Delta = \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \text{☣}; \mathtt{n}), \Delta_2}{\Phi; \mathtt{t}'; \mathtt{m}; \Psi \triangleright \mathtt{x}^{\sigma}\ \mathtt{is}\ \text{☣} \xrightarrow{\varepsilon} \Phi; \mathtt{t}'; \mathtt{m}; \Psi \triangleright 0^{\text{🔓}}}\ (e - \mathtt{x\ is}\ \text{☣}-\text{yes})$$

$$\frac{\Psi = \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta \qquad \Delta = \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{t}; \square; \mathtt{n}), \Delta_2}{\Phi; \mathtt{t}'; \mathtt{m}; \Psi \triangleright \mathtt{x}^{\sigma}\ \mathtt{is}\ \text{☣} \xrightarrow{\varepsilon} \Phi; \mathtt{t}'; \mathtt{m}; \Psi \triangleright 1^{\text{🔓}}}\ (e - \mathtt{x\ is}\ \text{☣}-\text{no})$$

$$\frac{}{\Omega \triangleright \mathtt{n}^{\sigma}\ \mathtt{has}\ \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 0^{\text{🔓}}}\ (e - \mathtt{n\ has}\ \mathbb{N}) \qquad \frac{}{\Omega \triangleright \langle \mathtt{n}_1^{\sigma}, \mathtt{n}_2^{\sigma'} \rangle\ \mathtt{has}\ \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 1^{\text{🔓}}}\ (e - pair - \mathtt{has}\ \mathbb{N})$$

$$\frac{}{\Omega \triangleright \mathtt{x}^{\sigma}\ \mathtt{has}\ \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 1^{\text{🔓}}}\ (e - \mathtt{x\ has}\ \mathbb{N}) \qquad \frac{}{\Omega \triangleright \mathtt{n}^{\sigma}\ \mathtt{has}\ \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 1^{\text{🔓}}}\ (e - \mathtt{n\ has}\ \mathbb{N} \times \mathbb{N})$$

$$\frac{}{\Omega \triangleright \langle \mathtt{n}_1^{\sigma}, \mathtt{n}_2^{\sigma'} \rangle\ \mathtt{has}\ \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 0^{\text{🔓}}}\ (e - pair - \mathtt{has}\ \mathbb{N} \times \mathbb{N}) \qquad \frac{}{\Omega \triangleright \mathtt{x}^{\sigma}\ \mathtt{has}\ \mathbb{N} \times \mathbb{N} \xrightarrow{\varepsilon} \Omega \triangleright 1^{\text{🔓}}}\ (e - \mathtt{x\ has}\ \mathbb{N} \times \mathbb{N})$$

$$\frac{}{\Phi; \mathtt{t}; \mathtt{m}; \Psi \triangleright \mathtt{setDIT}\ \mathtt{n}^{\sigma} \xrightarrow{\varepsilon} \Phi; \mathtt{t}; \mathtt{n}; \Psi \triangleright \mathtt{n}^{\sigma}}\ (e - \mathtt{setDIT\ n})$$

$$\frac{}{\Phi; \mathtt{t}; \mathtt{n}; \Psi \triangleright \mathtt{getDIT}\ \mathtt{x}^{\sigma}\ \mathtt{in}\ \mathtt{e} \xrightarrow{\varepsilon} \Phi; \mathtt{t}; \mathtt{n}; \Psi \triangleright \mathtt{e}[\mathtt{n}^{\sigma}/\mathtt{x}]}\ (e - \mathtt{getDIT\ x\ in\ e})$$

Figure 72: Evaluation of $\mathrm{L}_{\mathrm{scct}}$ expressions, continued.

$$\boxed{\Omega \triangleright \mathtt{K}[\mathtt{e}] \xrightarrow{\mathtt{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{K}[\mathtt{e}']}$$

„Given an evaluation context K and an expression e, it evaluates under configuration Ω to e′ and new configuration Ω′ in context K, emitting event a."

$$\frac{\Omega \triangleright \mathtt{e} \xrightarrow{\mathtt{a}} \Omega' \triangleright \mathtt{e}'}{\Omega \triangleright \mathtt{K}[\mathtt{e}] \xrightarrow{\mathtt{a}}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{K}[\mathtt{e}']}\ (e-\mathsf{ctx}) \qquad \frac{\Omega \triangleright \mathtt{e} \xrightarrow{\lightning} \Omega' \triangleright \mathtt{stuck}}{\Omega \triangleright \mathtt{K}[\mathtt{e}] \xrightarrow{\lightning}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{stuck}}\ (e-\mathsf{ctx}-\mathtt{stuck})$$

$$\frac{\begin{array}{c}\Omega = \xi; \Xi; [\cdot]; \mathtt{comp}; 0; \Psi \qquad \Xi = \Xi_1, (\mathtt{let\ main\ x} : \tau_\lambda := \mathtt{e}), \Xi_2 \\ \Omega' = \xi; \Xi; \mathtt{K}^{\mathtt{main}}, [\cdot]; \mathtt{ctx}; 0; \Psi\end{array}}{\Omega \triangleright \mathtt{K}[\mathtt{call\ main}^{\sigma}\ \mathtt{v}^{\sigma'}] \xrightarrow{\mathtt{Start};\mathtt{comp};\sigma'}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{e}[\mathtt{v}^{\sigma'}/\mathtt{x}]}\ (e-\mathsf{ctx}-\mathtt{call}-\mathsf{main})$$

$$\frac{\begin{array}{c}\Omega = \xi; \Xi; \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi \qquad \Xi = \Xi_1, (\mathtt{let\ foo\ x} : \tau_\lambda := \mathtt{e}), \Xi_2 \\ \mathtt{foo} \in_{\neg\mathtt{t}} \xi \qquad \rho(\neg\mathtt{t}) = \mathbf{c} \qquad \Omega' = \xi; \Xi; \mathtt{K}^{\mathtt{foo}}, \overline{\mathtt{K}}; \neg\mathtt{t}; \mathtt{m}; \Psi\end{array}}{\Omega \triangleright \mathtt{K}[\mathtt{call\ foo}^{\sigma}\ \mathtt{v}^{\sigma'}] \xrightarrow{\mathtt{Call}\ \mathbf{c}\ \mathtt{foo\ v};\mathtt{t};\sigma'}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{e}[\mathtt{v}^{\sigma'}/\mathtt{x}]}\ (e-\mathsf{ctx}-\mathtt{call}-\mathsf{notsame})$$

$$\frac{\begin{array}{c}\Omega = \xi; \Xi; \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi \qquad \Xi = \Xi_1, (\mathtt{let\ foo\ x} : \tau_\lambda := \mathtt{e}), \Xi_2 \\ \mathtt{foo} \in_{\mathtt{t}} \xi \qquad \Omega' = \xi; \Xi; \mathtt{K}^{\mathtt{foo}}, \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi\end{array}}{\Omega \triangleright \mathtt{K}[\mathtt{call\ foo}^{\sigma}\ \mathtt{v}^{\sigma'}] \xrightarrow{\mathtt{Call}\ \varnothing\ \mathtt{foo\ v};\mathtt{t};\sigma'}_{\mathsf{ctx}} \Omega' \triangleright \mathtt{e}[\mathtt{v}^{\sigma'}/\mathtt{x}]}\ (e-\mathsf{ctx}-\mathtt{call}-\mathsf{same})$$

$$\frac{}{\xi; \Xi; [\cdot]^{\mathtt{main}}, [\cdot]; \mathtt{ctx}; \mathtt{m}; \Psi \triangleright \mathtt{K}'[\mathtt{return\ v}^{\sigma}] \xrightarrow{\mathtt{End\ v};\mathtt{t};\sigma}_{\mathsf{ctx}} \xi; \Xi; [\cdot]; \mathtt{comp}; 1; \Psi \triangleright \mathtt{v}^{\sigma}}\ (e-\mathsf{ctx}-\mathtt{return}-\mathsf{main})$$

$$\frac{\mathtt{foo} \in_{\neg\mathtt{t}} \xi \qquad \rho(\neg\mathtt{t}) = \mathbf{c}; \mathtt{t}}{\xi; \Xi; \mathtt{K}^{\mathtt{foo}}, \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi \triangleright \mathtt{K}'[\mathtt{return\ v}^{\sigma}] \xrightarrow{\mathtt{Ret}\ \mathbf{c}\ \mathtt{v};\mathtt{t};\sigma}_{\mathsf{ctx}} \xi; \Xi; \overline{\mathtt{K}}; \neg\mathtt{t}; \mathtt{m}; \Psi \triangleright \mathtt{K}[\mathtt{v}^{\sigma}]}\ (e-\mathsf{ctx}-\mathtt{return}-\mathsf{notsame})$$

$$\frac{\mathtt{foo} \in_{\mathtt{t}} \xi}{\xi; \Xi; \mathtt{K}^{\mathtt{foo}}, \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi \triangleright \mathtt{K}'[\mathtt{return\ v}^{\sigma}] \xrightarrow{\mathtt{Ret}\ \varnothing\ \mathtt{v};\mathtt{t};\sigma}_{\mathsf{ctx}} \xi; \Xi; \overline{\mathtt{K}}; \mathtt{t}; \mathtt{m}; \Psi \triangleright \mathtt{K}[\mathtt{v}^{\sigma}]}\ (e-\mathsf{ctx}-\mathtt{return}-\mathsf{same})$$

$$\boxed{\rho(\mathtt{t}) = \mathbf{c}}$$

„Returns either ? or ! depending on t."

$$\frac{}{\rho(\mathtt{ctx}) = \mathtt{?}}\ (\mathsf{comm\text{-}ctxtocomp}) \qquad \frac{}{\rho(\mathtt{comp}) = \mathtt{!}}\ (\mathsf{comm\text{-}comptoctx})$$

$$\boxed{\neg\mathtt{t} = \mathtt{t}'}$$

„Negation of t."

$$\frac{}{\neg\mathtt{ctx} = \mathtt{comp}}\ (\mathsf{neg\text{-}ctx}) \qquad \frac{}{\neg\mathtt{comp} = \mathtt{ctx}}\ (\mathsf{neg\text{-}comp})$$

Figure 73: Contextual Evaluation of $\mathrm{L}_{\mathrm{scct}}$ expressions.

$$\boxed{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}'}$$ „Expression e evaluates under configuration Ω to e' and new configuration Ω', emitting list of events $\bar{\mathsf{a}}$."

$$\frac{}{\Omega \triangleright \mathsf{f}_{\lightning} \xrightarrow{[\cdot]}{}^{*}_{\mathrm{ctx}} \Omega \triangleright \mathsf{f}_{\lightning}}\ (et-\mathsf{refl})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}' \quad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright \mathsf{e}'' \quad \mathsf{a} \neq \varepsilon}{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}\cdot\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (et-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}' \quad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright \mathsf{e}''}{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega'' \triangleright \mathsf{e}''}\ (et-\mathsf{trans{-}unimportant})$$

$$\boxed{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}{}^{n}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}'}$$ „Expression e evaluates in n steps under configuration Ω to e' and new configuration Ω', emitting list of events $\bar{\mathsf{a}}$."

$$\frac{}{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}_{\mathrm{ctx}}{}^{0}\ \Omega \triangleright \mathsf{e}}\ (en-\mathsf{refl})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}' \quad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}_{\mathrm{ctx}}{}^{n}\ \Omega'' \triangleright \mathsf{e}'' \quad \mathsf{a} \neq \varepsilon}{\Omega \triangleright \mathsf{e} \xrightarrow{\mathsf{a}\cdot\bar{\mathsf{a}}}_{\mathrm{ctx}}{}^{(n+1)}\ \Omega'' \triangleright \mathsf{e}''}\ (en-\mathsf{trans{-}important})$$

$$\frac{\Omega \triangleright \mathsf{e} \xrightarrow{\varepsilon}_{\mathrm{ctx}} \Omega' \triangleright \mathsf{e}' \quad \Omega' \triangleright \mathsf{e}' \xrightarrow{\bar{\mathsf{a}}}_{\mathrm{ctx}}{}^{n}\ \Omega'' \triangleright \mathsf{e}''}{\Omega \triangleright \mathsf{e} \xrightarrow{\bar{\mathsf{a}}}_{\mathrm{ctx}}{}^{(n+1)}\ \Omega'' \triangleright \mathsf{e}''}\ (en-\mathsf{trans{-}unimportant})$$

Figure 74: Trace prefix generation given a $\mathrm{L}_{\mathrm{scct}}$ program using the reflexive-transitive closure.

$$\boxed{\mathtt{prog}\ \Xi_{\mathtt{ctx}}\ \Xi_{\mathtt{comp}} \xRightarrow{\bar{\mathsf{a}}} \Omega \triangleright \mathsf{f}}$$ „Run $\mathrm{L}_{\mathrm{scct}}$ program $\mathtt{prog}\ \Xi_{\mathtt{ctx}}\ \Xi_{\mathtt{comp}}$, giving dynamic state Ω and emitting trace $\bar{\mathsf{a}}$."

$$\frac{\Xi = \Xi_{\mathtt{ctx}} \bowtie \Xi_{\mathtt{comp}} \quad \mathtt{main} \notin \xi \quad \xi = \mathrm{dom}\ \Xi_{\mathtt{comp}} \qquad \xi; \Xi; [\cdot]\,; \mathtt{t}; 1; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathtt{call\ main\ 0} \xrightarrow{\bar{\mathsf{a}}}{}^{*}_{\mathrm{ctx}} \Omega \triangleright \mathsf{f}_{\lightning}}{\mathtt{prog}\ \Xi_{\mathtt{ctx}}\ \Xi_{\mathtt{comp}} \xRightarrow{\bar{\mathsf{a}}} \Omega \triangleright \mathsf{f}_{\lightning}}\ (e-\mathsf{wprog})$$

$$\frac{\Xi = \Xi_{\mathtt{ctx}} \bowtie \Xi_{\mathtt{comp}} \quad \mathtt{main} \notin \xi \quad \xi = \mathrm{dom}\ \Xi_{\mathtt{comp}} \qquad \xi; \Xi; [\cdot]\,; \mathtt{t}; 1; [\cdot]\,; [\cdot]\,; [\cdot] \triangleright \mathtt{call\ main\ 0} \xrightarrow{\bar{\mathsf{a}}\cdot\lightning;\mathtt{t}}{}^{*}_{\mathrm{ctx}} \Omega \triangleright \mathsf{f}_{\lightning}}{\mathtt{prog}\ \Xi_{\mathtt{ctx}}\ \Xi_{\mathtt{comp}} \xRightarrow{\bar{\mathsf{a}}\cdot\lightning;\mathtt{t}} \Omega \triangleright \mathsf{f}_{\lightning}}\ (e-\mathsf{wprog}-\lightning)$$

Figure 75: Running a whole $\mathrm{L}_{\mathrm{scct}}$ program.

4.7.2 Translation to Specification Events

$\boxed{\delta_{MS}(\ell) = \underline{\ell}}$ „A map from $\mathrm{L_{scct}}$ memory locations ℓ to specification locations $\underline{\ell}$."

$\boxed{\theta_{\delta_{MS}}(\mathtt{a}) = a^{\mathrm{ms}}}$ „Project an $\mathrm{L_{scct}}$ event to specification events."

$$\text{(o-filter-context)}\ \frac{\mathtt{a_b} \neq \lightning}{\theta_{\delta_{MS}}(\mathtt{a_b}; \mathtt{ctx}; \sigma) = \underline{\varepsilon}} \qquad \text{(o-filter-comp-start)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{Start}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}$$

$$\text{(o-filter-comp-end)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{End\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}} \qquad \text{(o-filter-comp-alloc)}\ \frac{\delta_{MS}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{MS}}(\mathtt{Alloc}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\mathit{Alloc}\ \ell\ n}}$$

$$\text{(o-filter-comp-dealloc)}\ \frac{\delta_{MS}(\ell) = \underline{\ell}}{\theta_{\delta_{MS}}(\mathtt{Dealloc}\ \ell; \mathtt{comp}; \sigma) = \underline{\mathit{Dealloc}\ \ell}}$$

$$\text{(o-filter-comp-get-noleak)}\ \frac{\delta_{MS}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{MS}}(\mathtt{iGet}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n}}$$

$$\text{(o-filter-comp-set-noleak)}\ \frac{\delta_{MS}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{MS}}(\mathtt{iSet}\ \ell\ \mathtt{n\ v}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n}}$$

$$\text{(o-filter-comp-get-leak)}\ \frac{\delta_{MS}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{MS}}(\mathtt{Get}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n}}$$

$$\text{(o-filter-comp-set-leak)}\ \frac{\delta_{MS}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{MS}}(\mathtt{Set}\ \ell\ \mathtt{n\ v}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n}}$$

$$\text{(o-filter-comp-call)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{Call\ c\ foo\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}} \qquad \text{(o-filter-comp-ret)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{Ret\ c\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}$$

$$\text{(o-filter-comp-binop)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{Binop\ n_1\ n_2}; \mathtt{comp}; \sigma) = \underline{\varepsilon}} \qquad \text{(o-filter-comp-branch)}\ \frac{}{\theta_{\delta_{MS}}(\mathtt{Branch\ n}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}$$

$$\text{(o-filter-abort)}\ \frac{}{\theta_{\delta_{MS}}(\lightning; \mathtt{t}; \sigma) = \underline{\lightning}}$$

$\boxed{\theta^*_{\delta_{MS}}(\bar{\mathtt{a}}) = \overline{a^{\mathrm{ms}}}}$ „Project an $\mathrm{L_{scct}}$ trace to specification traces."

$$\text{(o-filter-empty)}\ \frac{}{\theta^*_{\delta_{MS}}([\cdot]) = \underline{[\cdot]}} \qquad \text{(o-filter-cons-relevant)}\ \frac{\theta_{\delta_{MS}}(\mathtt{a}) = \underline{a} \quad \theta^*_{\delta_{MS}}(\bar{\mathtt{a}}) = \overline{a^{\mathrm{ms}}} \quad \underline{a} \neq \underline{\varepsilon}}{\theta^*_{\delta_{MS}}(\mathtt{a} \cdot \bar{\mathtt{a}}) = \underline{a} \cdot \overline{a^{\mathrm{ms}}}}$$

$$\text{(o-filter-cons-relevant)}\ \frac{\theta_{\delta_{MS}}(\mathtt{a}) = \underline{\varepsilon} \quad \theta^*_{\delta_{MS}}(\bar{\mathtt{a}}) = \overline{a^{\mathrm{ms}}}}{\theta^*_{\delta_{MS}}(\mathtt{a} \cdot \bar{\mathtt{a}}) = \overline{a^{\mathrm{ms}}}}$$

Figure 76: Projection of $\mathrm{L_{scct}}$ events to specification events, but for memory safety. Since the security tag of specification events for memory safety doesn't play any role whatsoever, we leave them out for brevity.

$\boxed{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell}}$ „A map from $\mathrm{L}_{\mathrm{scct}}$ memory locations ℓ to specification locations $\underline{\ell}$.”

$\boxed{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{a}) = \underline{a}^{\mathrm{scct}}}$ „Project an $\mathrm{L}_{\mathrm{scct}}$ event to specification events.”

$$\frac{\mathtt{a_b} \neq \text{↯}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{a_b}; \mathtt{ctx}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-context)} \qquad \frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Start}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-start)}$$

$$\frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{End\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-end)} \qquad \frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Alloc}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\mathit{Alloc}\ \ell\ n; \text{🔓}}}\ \text{(o-scct-filter-comp-alloc)}$$

$$\frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Dealloc}\ \ell; \mathtt{comp}; \sigma) = \underline{\mathit{Dealloc}\ \ell; \text{🔓}}}\ \text{(o-scct-filter-comp-dealloc)}$$

$$\frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{iGet}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-get-noleak)} \qquad \frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{iSet}\ \ell\ \mathtt{n\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-set-noleak)}$$

$$\frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Get}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n; \sigma}}\ \text{(o-scct-filter-comp-get-leak)}$$

$$\frac{\delta_{\texttt{sCCT}}(\ell) = \underline{\ell} \quad \underline{n} = \mathtt{n}}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Set}\ \ell\ \mathtt{n\ v}; \mathtt{comp}; \sigma) = \underline{\mathit{Use}\ \ell\ n; \sigma}}\ \text{(o-scct-filter-comp-set-leak)}$$

$$\frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Call\ c\ foo\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-call)} \qquad \frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Ret\ c\ v}; \mathtt{comp}; \sigma) = \underline{\varepsilon}}\ \text{(o-scct-filter-comp-ret)}$$

$$\frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Binop\ n_1\ n_2}; \mathtt{comp}; \sigma) = \underline{\mathit{Binop}\ n_1\ n_2; \sigma}}\ \text{(o-scct-filter-comp-binop)}$$

$$\frac{}{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{Branch\ n}; \mathtt{comp}; \sigma) = \underline{\mathit{Branch}\ n; \sigma}}\ \text{(o-scct-filter-comp-branch)} \qquad \frac{}{\theta_{\delta_{\texttt{sCCT}}}(\text{↯}; \mathtt{t}; \sigma) = \underline{\text{↯}; \sigma}}\ \text{(o-scct-filter-abort)}$$

$$\frac{}{\theta_{\delta_{\texttt{sCCT}}}(\varepsilon) = \underline{\varepsilon}}\ \text{(o-scct-filter-emptyevent)}$$

$\boxed{\theta^*_{\delta_{\texttt{sCCT}}}(\overline{\mathtt{a}}) = \overline{\underline{a}^{\mathrm{scct}}}}$ „Project an $\mathrm{L}_{\mathrm{scct}}$ trace to specification traces.”

$$\frac{}{\theta^*_{\delta_{\texttt{sCCT}}}([\cdot]) = \underline{[\cdot]}}\ \text{(o-scct-filter-empty)} \qquad \frac{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{a}) = \underline{a} \quad \theta^*_{\delta_{\texttt{sCCT}}}(\overline{\mathtt{a}}) = \overline{\underline{a}^{\mathrm{scct}}} \quad \underline{a} \neq \underline{\varepsilon}}{\theta^*_{\delta_{\texttt{sCCT}}}(\mathtt{a} \cdot \overline{\mathtt{a}}) = \underline{a} \cdot \overline{\underline{a}^{\mathrm{scct}}}}\ \text{(o-scct-filter-cons-relevant)}$$

$$\frac{\theta_{\delta_{\texttt{sCCT}}}(\mathtt{a}) = \underline{\varepsilon} \quad \theta^*_{\delta_{\texttt{sCCT}}}(\overline{\mathtt{a}}) = \overline{\underline{a}^{\mathrm{scct}}}}{\theta^*_{\delta_{\texttt{sCCT}}}(\mathtt{a} \cdot \overline{\mathtt{a}}) = \overline{\underline{a}^{\mathrm{scct}}}}\ \text{(o-scct-filter-cons-relevant)}$$

Figure 77: Projection of $\mathrm{L}_{\mathrm{scct}}$ events to specification events.

$\boxed{\delta(\ell) = \ell}$ „Map from L locations ℓ to L_{scct} locations ℓ."

$\boxed{\overline{a} \approxeq^* \bar{a}}$ „The L trace $\overline{a}$ describes the same actions as L_{scct} trace $\bar{a}$."

$$\frac{}{[\cdot] \approxeq^* [\cdot]}\ \text{(scct-empty-trace-eq)} \qquad \frac{t = \mathtt{t} \quad a_b \approxeq_\delta \mathtt{a_b} \quad \overline{a} \approxeq^* \bar{\mathtt{a}}}{a_b; t \cdot \overline{a} \approxeq^* \mathtt{a_b}; \mathtt{t}; \sigma \cdot \bar{\mathtt{a}}}\ \text{(scct-cons-trace-eq)}$$

$\boxed{a_b \approxeq_\delta \mathtt{a_b}}$ „The L event a_b describes the same action as L event $\mathtt{a_b}$."

$$\frac{}{Start \approxeq_\delta \mathtt{Start}}\ \text{(scct-start-event-eq)} \qquad \frac{[\![v]\!]^{L \to L_{scct}} = \mathtt{v}}{End\ v \approxeq_\delta \mathtt{End\ v}}\ \text{(scct-end-event-eq)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad \delta(\ell) = \ell}{Alloc\ \ell\ n \approxeq_\delta \mathtt{Alloc}\ \ell\ \mathtt{n}}\ \text{(scct-alloc-event-eq)} \qquad \frac{\delta(\ell) = \ell}{Dealloc\ \ell \approxeq_\delta \mathtt{Dealloc}\ \ell}\ \text{(scct-dealloc-event-eq)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad \delta(\ell) = \ell}{Set\ \ell\ n\ v \approxeq_\delta \mathtt{iGet}\ \ell\ \mathtt{n}}\ \text{(scct-get-event-eq-noleak)} \qquad \frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad \delta(\ell) = \ell}{Get\ \ell\ n \approxeq_\delta \mathtt{Get}\ \ell\ \mathtt{n}}\ \text{(scct-get-event-eq-leak)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad [\![v]\!]^{L \to L_{scct}} = \mathtt{v} \quad \delta(\ell) = \ell}{Set\ \ell\ n\ v \approxeq_\delta \mathtt{Set}\ \ell\ \mathtt{n\ v}}\ \text{(scct-set-event-eq-leak)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad [\![v]\!]^{L \to L_{scct}} = \mathtt{v} \quad \delta(\ell) = \ell}{Set\ \ell\ n\ v \approxeq_\delta \mathtt{iSet}\ \ell\ \mathtt{n\ v}}\ \text{(scct-set-event-eq-noleak)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad [\![foo]\!]^{L \to L_{scct}} = \mathtt{foo} \quad \mathbf{c} = \mathbf{c}}{Call\ \mathbf{c}\ foo\ n \approxeq_\delta \mathtt{Call}\ \mathbf{c}\ \mathtt{foo\ n}}\ \text{(scct-call-event-eq)}$$

$$\frac{[\![n]\!]^{L \to L_{scct}} = \mathtt{n} \quad \mathbf{c} = \mathbf{c}}{Ret\ \mathbf{c}\ n \approxeq_\delta \mathtt{Ret}\ \mathbf{c}\ \mathtt{n}}\ \text{(scct-ret-event-eq)} \qquad \frac{}{\varepsilon \approxeq_\delta \varepsilon}\ \text{(scct-}\varepsilon\text{-event-eq)} \qquad \frac{}{\lightning \approxeq_\delta \lightning}\ \text{(scct-}\lightning\text{-event-eq)}$$

$$\frac{}{\varepsilon \approxeq_\delta \mathtt{Binop\ n_1\ n_2}}\ \text{(scct-binop-event-eq-leak)} \qquad \frac{}{\varepsilon \approxeq_\delta \mathtt{Branch\ n}}\ \text{(scct-branch-event-eq-leak)}$$

$\boxed{\mathbf{c} = \mathbf{c}}$ „Communications are equal."

$$\frac{}{? = ?}\ (? = ?) \qquad \frac{}{! = !}\ (! = !) \qquad \frac{}{\varnothing = \varnothing}\ (\varnothing = \varnothing)$$

Figure 78: Trace Relation from L to L_{scct}.

$\boxed{\delta(\ell) = \ell}$ „The L memory location ℓ corresponds to the $\mathrm{L_{scct}}$ memory location ℓ.”

$\boxed{\Omega \approx_{\delta_{scct}} \Omega}$ „The L state Ω agrees with L state Ω.”

(scct-state-eq)

$$\frac{\Omega = \Phi; t; \Psi \quad \Omega = \Phi; \mathtt{t}; \mathtt{n}; \Psi \quad \mathtt{n} \neq 0 \quad t = \mathtt{t} \quad \Phi \approx \Phi \quad \Psi \approx_{\delta_{scct}} \Psi}{\Omega \approx_{\delta_{scct}} \Omega}$$

(scct-abort-state-eq)

$$\frac{}{\lightning \approx_{\delta_{scct}} \lightning}$$

$\boxed{\Psi \approx_{\delta_{scct}} \Psi}$ „The L memory-state Ψ agrees with $\mathrm{L_{scct}}$ one Ψ. L contains locations introduced by the backtranslation wrapper.”

(scct-empty-memstate-eq)

$$\frac{}{[\cdot] ; [\cdot] ; [\cdot] \approx_{\delta_{scct}} [\cdot] ; [\cdot] ; [\cdot]}$$

(scct-comp-cons-memstate-eq)

$$\frac{\ell \notin L \quad n = \mathtt{n} \quad \rho = \rho \quad \delta(\ell) = \ell \quad \ell, n \vdash_\delta H^{comp} \approx \mathtt{H}^{\mathtt{comp}} \quad H^{ctx}; H^{comp}; \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \Delta_2}{H^{ctx}; H^{comp}; x \mapsto (\ell; comp; \rho; n), \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta_1, \mathtt{x} \mapsto (\ell; \mathtt{comp}; \rho; \mathtt{n}), \Delta_2}$$

(scct-ctx-cons-memstate-eq)

$$\frac{\ell \notin L \quad H^{ctx}; H^{comp}; \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta}{H^{ctx}; H^{comp}; x \mapsto (\ell; ctx; \rho; n), \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta}$$

(scct-whatever-cons-memstate-eq)

$$\frac{\ell \in L \quad H^{ctx}; H^{comp}; \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta}{H^{ctx}; H^{comp}; x \mapsto (\ell; t; \rho; n), \Delta \approx_{\delta_{scct}} \mathtt{H}^{\mathtt{ctx}}; \mathtt{H}^{\mathtt{comp}}; \Delta}$$

$\boxed{\ell, n \vdash_\delta H^{comp} \approx \mathtt{H}^{\mathtt{comp}}}$ „The heaps H^{comp} and $\mathtt{H}^{\mathtt{comp}}$ are related at ℓ for n memory cells.”

(scct-heap-related)

$$\frac{[\![n]\!]^{\mathrm{L}\to\mathrm{L_{scct}}} = \mathtt{n} \quad \delta(\ell) = \ell \quad \forall 0 \leq m < n,\ 0 \leq \mathtt{m} < \mathtt{n},\ H^{comp}(\ell + m) = \mathtt{H}^{\mathtt{comp}}(\ell + \mathtt{m})}{\ell, n \vdash_\delta H^{comp} \approx \mathtt{H}^{\mathtt{comp}}}$$

$\boxed{\Phi \approx \Phi}$ „The L control-flow-state Φ agrees with $\mathrm{L_{scct}}$ one Φ.”

(scct-cfstate-eq)

$$\frac{\Phi = \xi; \Xi; \overline{K} \quad \Phi = \xi; \Xi; \overline{\mathtt{K}} \quad \Xi \approx \Xi \quad \overline{K} \approx_\xi \overline{\mathtt{K}} \quad \xi = [\![\xi]\!]^{\mathrm{L}\to\mathrm{L_{scct}}}}{\Phi \approx \Phi}$$

$\boxed{\rho = \rho}$ „L poison equals $\mathrm{L_{scct}}$ one ρ.”

(☣-equal)

$$\frac{}{☣ = ☣}$$

(☐-equal)

$$\frac{}{\square = \square}$$

Figure 79: State Relation from L to $\mathrm{L_{scct}}$. This is meant to relate the states whenever we are „inside” a component. Note that the DOIT flag is required to be non-zero.

$\boxed{\Xi \approx \Xi}$ „The procedures of L agree with L_{scct} ones."

(scct-empty-commlib-lib-eq)
$$\frac{}{[\cdot] \approx \Xi}$$

(scct-cons-commlib-eq)
$$\frac{[\![\mathit{let\ foo\ x} : \tau_\lambda := e]\!]^{L \to L_{scct}} = \texttt{let foo x := e} \qquad \Xi \approx \Xi_1, \Xi_2}{(\mathit{let\ foo\ x} : \tau_\lambda := e), \Xi \approx \Xi_1, (\texttt{let foo x := e}), \Xi_2}$$

$\boxed{\overline{K} \approx_\xi \overline{\mathtt{K}}}$ „The stack of L continuations $\overline{K}$ agrees with L_{scct} one $\overline{\mathtt{K}}$. Only checks $foo \in \xi$"

(scct-empty-kontstack-eq)
$$\frac{}{[\cdot] \approx_\xi [\cdot]}$$

(scct-ign-cons-kontstack-eq)
$$\frac{foo \notin \xi \qquad \texttt{foo} = [\![foo]\!]^{L \to L_{scct}} \qquad \overline{K} \approx_\xi \overline{\mathtt{K}}}{\overline{K} \approx_\xi (\mathtt{K}; \texttt{foo}), \overline{\mathtt{K}}}$$

(cons-kontstack-eq)
$$\frac{foo \in \xi \qquad foo = \texttt{foo} \qquad \overline{K} \approx_\xi \overline{\mathtt{K}}}{(K; foo), \overline{K} \approx_\xi (\mathtt{K}; \texttt{foo}), \overline{\mathtt{K}}}$$

Figure 80: Memory-State Relations from L to L_{scct}.

$\boxed{\delta(\ell) = \ell}$ „The L memory location ℓ corresponds to the L_{scct} memory location ℓ."

$\boxed{\Omega \multimap_{\delta_{scct}} \Omega}$ „The L state Ω agrees with L_{scct} state Ω. L contains locations introduced by the backtranslation wrapper, which are subsequently ignored."

(state-qe)
$$\frac{\Omega = \Phi; t; \Psi \qquad \Omega = \Phi; \mathtt{t}; \mathtt{n}; \Psi \qquad \Phi \multimap \Phi \qquad \Psi \multimap_{\delta_{scct}} \Psi}{\Omega \multimap_{\delta_{scct}} \Omega}$$

(abort-state-qe)
$$\frac{}{\lightning \multimap_{\delta_{scct}} \lightning}$$

$\boxed{\Psi \multimap_{\delta_{scct}} \Psi}$ „The L memory-state Ψ agrees with L_{scct} one Ψ. L contains locations introduced by the backtranslation wrapper."

(memstate-qe)
$$\frac{\Psi \approx_{\delta_{scct}} \Psi}{\Psi \multimap_{\delta_{scct}} \Psi}$$

$\boxed{\Phi \multimap \Phi}$ „The L control-flow-state Φ agrees with L_{scct} one Φ."

(cfstate-qe)
$$\frac{\Phi \approx \Phi}{\Phi \multimap \Phi}$$

Figure 81: State Relation from L to L_{scct}. This is meant to relate the states whenever we are „inside" a context.

4.8 Robustly Preserving sCCT

4.8.1 Compiler

$\boxed{[\![e]\!]^{L \to L_{scct}} = \mathtt{e}'}$ „Compile L expression e to L_{scct} expression $\mathtt{e}'$."

$\boxed{[\![[v/x]]\!]^{L \to L_{scct}} = [\mathtt{v}/\mathtt{x}]}$ „Compile L substitution to L_{scct} substitution."

$\boxed{[\![\xi]\!]^{L \to L_{scct}} = \xi'}$ „Compile L component library to L_{scct} component library."

$$
\begin{aligned}
[\![f_{\lightning}]\!]^{L \to L_{scct}} &= \mathtt{f}_{\lightning} \\
[\![\mathit{call\ foo\ e}]\!]^{L \to L_{scct}} &= \mathtt{call}\ [\![\mathit{foo}]\!]^{L \to L_{scct}}\ [\![e]\!]^{L \to L_{scct}}; \mathtt{setDIT\ 1} \\
[\![\mathit{return\ e}]\!]^{L \to L_{scct}} &= \mathtt{return}\ [\![e]\!]^{L \to L_{scct}} \\
[\![e_1 \oplus e_2]\!]^{L \to L_{scct}} &= [\![e_1]\!]^{L \to L_{scct}} \oplus [\![e_2]\!]^{L \to L_{scct}} \\
[\![x[e]]\!]^{L \to L_{scct}} &= \mathtt{x}[[\![e]\!]^{L \to L_{scct}}] \\
[\![\mathit{let}\ x = e_1\ \mathit{in}\ e_2]\!]^{L \to L_{scct}} &= \mathtt{let\ x} = [\![e_1]\!]^{L \to L_{scct}}\ \mathtt{in}\ [\![e_2]\!]^{L \to L_{scct}} \\
[\![x[e_1] \leftarrow e_2]\!]^{L \to L_{scct}} &= \mathtt{x}[[\![e_1]\!]^{L \to L_{scct}}] \leftarrow [\![e_2]\!]^{L \to L_{scct}} \\
[\![\mathit{let}\ x = \mathit{new}\ e_1\ \mathit{in}\ e_2]\!]^{L \to L_{scct}} &= \mathtt{let\ x = new}\ [\![e_1]\!]^{L \to L_{scct}}\ \mathtt{in}\ [\![e_2]\!]^{L \to L_{scct}} \\
[\![\mathit{delete}\ x]\!]^{L \to L_{scct}} &= \mathtt{delete\ x} \\
[\![x\ \mathit{is}\ \biohazard]\!]^{L \to L_{scct}} &= \mathtt{x\ is}\ \biohazard \\
[\![(e_1; e_2)]\!]^{L \to L_{scct}} &= ([\![e_1]\!]^{L \to L_{scct}}; [\![e_2]\!]^{L \to L_{scct}}) \\
[\![\pi_1\ e]\!]^{L \to L_{scct}} &= \pi_1\ [\![e]\!]^{L \to L_{scct}} \\
[\![\pi_2\ e]\!]^{L \to L_{scct}} &= \pi_2\ [\![e]\!]^{L \to L_{scct}} \\
[\![e\ \mathit{has}\ \tau]\!]^{L \to L_{scct}} &= [\![e]\!]^{L \to L_{scct}}\ \mathtt{has}\ \tau \\
[\![\mathit{ifz}\ e_1\ \mathit{then}\ e_2\ \mathit{else}\ e_3]\!]^{L \to L_{scct}} &= \mathtt{ifz}\ [\![e_1]\!]^{L \to L_{scct}} \\
&\quad\ \mathtt{then}\ [\![e_2]\!]^{L \to L_{scct}} \\
&\quad\ \mathtt{else}\ [\![e_3]\!]^{L \to L_{scct}}
\end{aligned}
$$

$$
[\![[v/x]]\!]^{L \to L_{scct}} = [\,[\![v]\!]^{L \to L_{scct}} / [\![x]\!]^{L \to L_{scct}}\,]
$$

$$
\begin{aligned}
[\![[\cdot]]\!]^{L \to L_{scct}} &= [\cdot] \\
[\![\mathit{foo}, \xi]\!]^{L \to L_{scct}} &= \mathtt{foo}, [\![\xi]\!]^{L \to L_{scct}}
\end{aligned}
$$

Figure 82: Compiler from L to L_{scct}.

$\boxed{[\![F]\!]^{L \to L_{scct}} = \mathtt{F}}$ „Compile L procedures to L_{scct} procedures."

$$
[\![\mathit{let\ foo\ x} := e]\!]^{L \to L_{scct}} = \mathtt{let\ foo\ x} := \mathtt{setDIT\ 1}; [\![e]\!]^{L \to L_{scct}}
$$

$\boxed{[\![\Xi]\!]^{L \to L_{scct}} = \Xi}$ „Compile L libraries to L_{scct} libraries."

$$
\begin{aligned}
[\![[\cdot]]\!]^{L \to L_{scct}} &= [\cdot] \\
[\![F, \Xi]\!]^{L \to L_{scct}} &= [\![F]\!]^{L \to L_{scct}}, [\![\Xi]\!]^{L \to L_{scct}}
\end{aligned}
$$

Figure 83: Compiler from L components to L_{scct} components.

4.8.2 Proofs and Auxiliary Lemmas

Lemma 83 (Trace Equality transfer Safety)**.** *If*

(a) $\forall \ell\ \ell, \delta(\ell) = \ell \implies \delta_{scct}(\ell) = \delta_{scct}(\delta^{-1}(\ell))$

(b) $\overline{a} \approx^{*}_{\delta} \overline{\mathtt{a}}$

(c) $\theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\overline{\mathtt{a}}) \cong \overline{\boldsymbol{a}}$

(d) $\vdash T_{sCCT} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{sCCT}{}'$

Then

(i) $\theta_{\delta_{scct}}(\overline{a}) \cong \overline{\boldsymbol{a}}$

Proof. Induction on Assumption (b).

Case scct-empty-trace-eq**:** By Rule i-scct-filter-empty we know $\theta_{\delta_{scct}}([\cdot]) = \underline{[\cdot]}$ and similarily Rule o-scct-filter-empty we know $\theta_{\delta_{scct} \circ \delta^{-1}}([\cdot]) = \underline{[\cdot]}$. Invert Assumption (c) to know that $\overline{\boldsymbol{a}} = [\cdot]$. Rule scct-None resolves the goal.

Case scct-cons-trace-eq**:**

For readability, we repeat the proof context: If

(H_1) $\forall \ell\ \ell, \delta(\ell) = \ell \implies \delta_{scct}(\ell) = \delta_{scct}(\delta^{-1}(\ell))$

(H_2) $a_b; t \cdot \overline{a} \approx^{*}_{\delta} \mathtt{a_b}; \mathtt{t}; \sigma \cdot \overline{\mathtt{a}}$

(H_3) $\theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\mathtt{a_b}; \mathtt{t}; \sigma \cdot \overline{\mathtt{a}}) \cong \overline{\boldsymbol{a}}$

(H_4) $\vdash T_{\mathrm{sCCT}} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\mathrm{sCCT}}{}'$

(IH_1) Given $\overline{\boldsymbol{a}}_{IH}, T_{\mathrm{sCCT}\,IH}, T_{\mathrm{sCCT}}{}'_{IH}$,
if $\overline{a} \approx^{*}_{\delta} \overline{\mathtt{a}}$
and $\theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\overline{\mathtt{a}}) = \overline{\boldsymbol{a}}_{IH}$
and $\vdash T_{\mathrm{sCCT}\,IH} \overset{\overline{\boldsymbol{a}}_{IH}}{\rightsquigarrow}{}^{*} T_{\mathrm{sCCT}}{}'_{IH}$
then $\theta^{*}_{\delta_{scct}}(\overline{a}) = \overline{\boldsymbol{a}}_{IH}$

Then

(i) $\theta^{*}_{\delta_{scct}}(a_b; t \cdot \overline{a}) \cong \overline{\boldsymbol{a}}$

Invert Assumption (H_2) to get:

(H_5) $t = \mathtt{t}$

(H_6) $a_b \approx_{\delta} \mathtt{a_b}$

(H_7) $\overline{a} \approx^{*}_{\delta} \overline{\mathtt{a}}$

Note that $\theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\mathtt{a} \cdot \overline{\mathtt{a}}) = \theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\mathtt{a}) \doteq \theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\overline{\mathtt{a}})$ and rewrite Assumption (c) with that.

So, by inversion on Assumption (c) we know:

(H_8) $\overline{\boldsymbol{a}} = \boldsymbol{a} \cdot \overline{\boldsymbol{a}}'$

(H_9) $\theta^{*}_{\delta_{scct} \circ \delta^{-1}}(\mathtt{a}) \cong \boldsymbol{a}$

(H_{10}) $\theta^*_{\delta_{scct}\circ\delta^{-1}}(\overline{\mathtt{a}}) \cong^* \overline{\boldsymbol{a}}'$

Invert Assumption (H_4):

(H_{11}) $\vdash T_{\text{sCCT}} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{\text{sCCT}0}$

(H_{12}) $\vdash T_{\text{sCCT}0} \overset{\overline{\boldsymbol{a}}}{\rightsquigarrow}{}^{*} T_{\text{sCCT}}'$

Rewrite Goal (i), exploiting distributivity of the filter:

(i) $\theta^*_{\delta_{scct}}(a_b; t) \cong \boldsymbol{a}$

(ii) $\theta^*_{\delta_{scct}}(\overline{a}) \cong \overline{\boldsymbol{a}}'$

We are ready to solve Goal (ii) with Assumption (IH_1) using Assumptions (H_7), (H_{10}) and (H_{12}).

For Goal (i), perform case analysis on Assumption (H_6). We leave cases with $t = ctx$ out, they are easy by Rule i-scct-filter-context, noting that by inverting Assumption (H_9) would give us $\boldsymbol{a} = \boldsymbol{\varepsilon}$, which then allows us to apply Rule `sCCT`-None-Authentic. So, in the following, let $t = comp$ and $\mathtt{t} = \mathtt{comp}$.

Case scct-start-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-start, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-end-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-end, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-alloc-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{Any};$🔓. Note Rule i-scct-filter-comp-alloc, so Goal (i) follows from Rule `sCCT`-Public-Authentic.

Case scct-dealloc-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{Any};$🔓. Note Rule i-scct-filter-comp-dealloc, so Goal (i) follows from Rule `sCCT`-Public-Authentic.

Case scct-get-event-eq-noleak**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-get-noleak, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-set-event-eq-noleak**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-set-noleak, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-get-event-eq-leak**:** Case analysis on $\boldsymbol{\sigma}$.

Case $\boldsymbol{\sigma} =$ 🔓**:** Invert Assumption (H_9) to get:

(F_1) $\boldsymbol{a} = \boldsymbol{\varepsilon}$.

Note Rule i-scct-filter-comp-get-noleak, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case $\boldsymbol{\sigma} =$ 🔒**:** Inverting Assumption (H_{11}) immediately gives us a contradiction, because there is no matching rule that steps on 🔒.

Case scct-set-event-eq-leak**:** Case analysis on $\boldsymbol{\sigma}$.

Case $\boldsymbol{\sigma} =$ 🔓**:** Invert Assumption (H_9) to get:

(F_1) $\boldsymbol{a} = \boldsymbol{\varepsilon}$.

Note Rule i-scct-filter-comp-set-noleak, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case $\boldsymbol{\sigma} = 🔒$**:** Inverting Assumption (H_{11}) immediately gives us a contradiction, because there is no matching rule that steps on 🔒.

Case scct-call-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-call, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-ret-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-comp-ret, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-ε-event-eq**:** Invert Assumption (H_9) to get $\boldsymbol{a} = \boldsymbol{\varepsilon}$. Note Rule i-scct-filter-emptyevent, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case scct-$\lightning$-event-eq**:** Invert Assumption (H_9) to get:

(F_1) $\underline{\sigma} = \boldsymbol{\sigma}$

(F_2) $\boldsymbol{a} = \lightning\,;\boldsymbol{\sigma}$.

Case analysis on $\boldsymbol{\sigma}$.

Case $\boldsymbol{\sigma} = 🔓$**:** Note Rule i-scct-filter-abort, so Goal (i) follows from Rule `sCCT`-Abort-Authentic.

Case $\boldsymbol{\sigma} = 🔒$**:** Inverting Assumption (H_{11}) immediately gives us a contradiction, because there is no matching rule that steps on 🔒.

Case scct-binop-event-eq-leak**:** This case is very similar to the next case.

Case scct-branch-event-eq-leak**:** Case analysis on $\boldsymbol{\sigma}$.

Case $\boldsymbol{\sigma} = 🔓$**:** Invert Assumption (H_9) to get:

(F_1) $\boldsymbol{a} = \boldsymbol{\varepsilon}$.

Note Rule i-scct-filter-emptyevent, so Goal (i) follows from Rule `sCCT`-None-Authentic.

Case $\boldsymbol{\sigma} = 🔒$**:** Inverting Assumption (H_{11}) immediately gives us a contradiction, because there is no matching rule that steps on 🔒.

□

Lemma 84 (Primitive Forward Simulation)**.** *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a} \Omega' \triangleright e'\gamma'$

(b) $\vdash T_{sCCT} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{sCCT}'$

(c) $\theta_\delta(a) \cong \boldsymbol{a}$

(d) $\Omega \approx_{\delta_{scct}} \Omega$

Then $\exists \delta'_{scct}\ \mathtt{a}\ \Omega'$,

(i) $\Omega \triangleright [\![e]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}[\![\gamma]\!]^{\mathsf{L}\to\mathsf{L}_{scct}} \xrightarrow{\mathtt{a}} \Omega' \triangleright [\![e']\!]^{\mathsf{L}\to\mathsf{L}_{scct}}[\![\gamma']\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(ii) $a \approxeq_{\delta'_{scct}} \mathtt{a}$

(iii) $\Omega' \approx_{\delta'_{scct}} \Omega'$

(iv) $\theta_{\delta'_{scct}}(\mathtt{a}) \cong \boldsymbol{a}$

Proof. Case analysis on Assumption (a). □

Lemma 85 (Contextual Forward Simulation)**.** *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a}_{ctx} \Omega' \triangleright e'\gamma'$

(b) $\vdash T_{sCCT} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{sCCT}'$

(c) $\theta_{\delta}(a) \cong \boldsymbol{a}$

(d) $\Omega \approx_{\delta_{scct}} \Omega$

Then $\exists \delta'_{scct}\ \mathtt{a}\ \Omega'$,

(i) $\Omega \triangleright [\![e]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} \xrightarrow{\mathtt{a}}_{ctx} \Omega' \triangleright [\![e']\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma']\!]^{\mathrm{L}\to\mathrm{L}_{scct}}$

(ii) $a \approxeq_{\delta'_{scct}} \mathtt{a}$

(iii) $\Omega' \approx_{\delta'_{scct}} \Omega'$

(iv) $\theta_{\delta'_{scct}}(\mathtt{a}) \cong \boldsymbol{a}$

Proof. Case analysis on Assumption (a) and using Lemma 84 (Primitive Forward Simulation). □

Lemma 86 (Forward Simulation)**.** *If*

(a) $\Omega \triangleright e\gamma \xrightarrow{a}{}^{*}_{ctx} \Omega' \triangleright e'\gamma'$

(b) $\vdash T_{sCCT} \overset{\boldsymbol{a}}{\rightsquigarrow} T_{sCCT}'$

(c) $\theta_{\delta}(a) \cong \boldsymbol{a}$

(d) $\Omega \approx_{\delta_{scct}} \Omega$

Then $\exists \delta'_{scct}\ \mathtt{a}\ \Omega'$,

(i) $\Omega \triangleright [\![e]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} \xrightarrow{\mathtt{a}}{}^{*}_{ctx} \Omega' \triangleright [\![e']\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma']\!]^{\mathrm{L}\to\mathrm{L}_{scct}}$

(ii) $a \approxeq_{\delta'_{scct}} \mathtt{a}$

(iii) $\Omega' \approx_{\delta'_{scct}} \Omega'$

(iv) $\theta_{\delta'_{scct}}(\mathtt{a}) \cong \boldsymbol{a}$

Proof. Case analysis on Assumption (a) and using Lemma 85 (Contextual Forward Simulation). □

Lemma 87 (Backward Simulation)**.** *If*

(a) $\Omega \triangleright [\![e]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} \xrightarrow{\bar{\mathtt{a}}}{}^{*}_{ctx} \Omega' \triangleright [\![e']\!]^{\mathrm{L}\to\mathrm{L}_{scct}} [\![\gamma']\!]^{\mathrm{L}\to\mathrm{L}_{scct}}$

(b) $\Omega \approx_{\delta_{scct}} \Omega$

(c) $\vdash T_{sCCT} \overset{\boldsymbol{a}}{\rightsquigarrow}{}^{*} T_{sCCT}'$

(d) $\theta_\delta(a) \cong \boldsymbol{a}$

Then $\exists \delta'_{scct}\ \Omega'\ \overline{a}$

(i) $\delta_{scct} \subseteq \delta'_{scct}$

(ii) $\Omega \triangleright e\gamma \xrightarrow{\overline{a}}{}^{*}_{ctx} \Omega' \triangleright e'\gamma'$

(iii) $\Omega' \approx_{\delta'_{scct}} \Omega'$

(iv) $\overline{a} \approxeq^{*}_{\delta'_{scct}} \overline{\mathtt{a}}$

(v) $\theta_{\delta'_{scct}}(\overline{\mathtt{a}}) \cong \overline{\boldsymbol{a}}$

Proof. Follows from Lemma 86 and determinism. □

4.8.3 Backtranslation

$\boxed{\theta_{\bullet}\,(\mathtt{a}) = \mathtt{a}}$ „Filter an $\mathrm{L}_{\mathrm{scct}}$ event."

(obj-filter-context)
$$\frac{\begin{array}{c}\mathtt{a_b} = \mathtt{Alloc}\ \ell\ \mathtt{n} \lor \mathtt{a_b} = \mathtt{Dealloc}\ \ell \lor \\ \mathtt{a_b} = \mathtt{Get}\ \ell\ \mathtt{n} \lor \mathtt{a_b} = \mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{m} \lor \\ \mathtt{a_b} = \mathtt{Call}\ \emptyset\ \mathtt{foo}\ \mathtt{n} \lor \mathtt{a_b} = \mathtt{Ret}\ \emptyset\ \mathtt{n} \lor \\ \mathtt{a_b} = \mathtt{iGet}\ \ell\ \mathtt{n} \lor \mathtt{a_b} = \mathtt{iSet}\ \ell\ \mathtt{n}\ \mathtt{m} \lor \\ \mathtt{a_b} = \mathtt{Branch}\ \mathtt{n} \lor \mathtt{a_b} = \mathtt{Binop}\ \mathtt{n_1}\ \mathtt{n_2}\end{array}}{\theta_{\bullet}\,(\mathtt{a_b}; \mathtt{ctx}; \sigma) = \varepsilon}$$

(obj-filter-unimportant)
$$\frac{}{\theta_{\bullet}\,(\varepsilon) = \varepsilon}$$

(obj-filter-comp-start)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Start}; \mathtt{comp}; \sigma) = \varepsilon}$$

(obj-filter-comp-alloc)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Alloc}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \mathtt{Alloc}\ \ell\ \mathtt{n}}$$

(obj-filter-comp-dealloc)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Dealloc}\ \ell; \mathtt{comp}; \sigma) = \mathtt{Dealloc}\ \ell}$$

(obj-filter-comp-get-noleak)
$$\frac{}{\theta_{\bullet}\,(\mathtt{iGet}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \mathtt{iGet}\ \ell\ \mathtt{n}}$$

(obj-filter-comp-set-noleak)
$$\frac{}{\theta_{\bullet}\,(\mathtt{iSet}\ \ell\ \mathtt{n}\ \mathtt{v}; \mathtt{comp}; \sigma) = \mathtt{iSet}\ \ell\ \mathtt{n}\ \mathtt{v}}$$

(obj-filter-comp-get-leak)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Get}\ \ell\ \mathtt{n}; \mathtt{comp}; \sigma) = \mathtt{Get}\ \ell\ \mathtt{n}}$$

(obj-filter-comp-set-leak)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{v}; \mathtt{comp}; \sigma) = \mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{v}}$$

(obj-filter-comp-call)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Call}\ \mathbf{c}\ \mathtt{foo}\ \mathtt{v}; \mathtt{t}; \sigma) = \mathtt{Call}\ \mathbf{c}\ \mathtt{foo}\ \mathtt{v}; \mathtt{t}}$$

(obj-filter-comp-ret)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Ret}\ \mathbf{c}\ \mathtt{v}; \mathtt{t}; \sigma) = \mathtt{Ret}\ \mathbf{c}\ \mathtt{v}; \mathtt{t}}$$

(obj-filter-branch)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Branch}\ \mathtt{n}; \mathtt{comp}; \sigma) = \mathtt{Branch}\ \mathtt{n}}$$

(obj-filter-binop)
$$\frac{}{\theta_{\bullet}\,(\mathtt{Binop}\ \mathtt{n_1}\ \mathtt{n_2}; \mathtt{comp}; \sigma) = \mathtt{Binop}\ \mathtt{n_1}\ \mathtt{n_2}}$$

(obj-filter-abort)
$$\frac{}{\theta_{\bullet}\,(\lightning; \mathtt{t}; \sigma) = \lightning}$$

$\boxed{\theta^{*}_{\bullet}\,(\overline{\mathtt{a}}) = \overline{\mathtt{a}}'}$ „Filter an $\mathrm{L}_{\mathrm{scct}}$ trace."

(obj-filter-empty)
$$\frac{}{\theta^{*}_{\bullet}\,([\cdot]) = [\cdot]}$$

(obj-filter-cons-relevant)
$$\frac{\theta_{\bullet}\,(\mathtt{a}) = \mathtt{a} \neq \varepsilon \qquad \theta^{*}_{\bullet}\,(\overline{\mathtt{a}}) = \overline{\mathtt{a}}}{\theta^{*}_{\bullet}\,(\mathtt{a} \cdot \overline{\mathtt{a}}) = \mathtt{a} \cdot \overline{\mathtt{a}}}$$

(obj-filter-cons-relevant)
$$\frac{\theta_{\bullet}\,(\mathtt{a}) = \varepsilon \qquad \theta^{*}_{\bullet}\,(\overline{\mathtt{a}}) = \overline{\mathtt{a}}}{\theta^{*}_{\bullet}\,(\mathtt{a} \cdot \overline{\mathtt{a}}) = \overline{\mathtt{a}}}$$

Figure 84: Filtering of $\mathrm{L}_{\mathrm{scct}}$ events, getting rid of unimportant ones, for backtranslation.

$\boxed{\langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle^{\mathrm{L}\to\mathrm{L}_{\mathrm{scct}}} = v}$ „Map an $\mathrm{L}_{\mathrm{scct}}$ value to an L value."

(scct-backtrans-value)
$$\frac{}{\langle\!\langle\!\langle \mathtt{n} \rangle\!\rangle\!\rangle^{\mathrm{L}\to\mathrm{L}_{\mathrm{scct}}} = n}$$

Figure 85: Backtranslation of $\mathrm{L}_{\mathrm{scct}}$ values to L values.

$\boxed{\langle\!\langle \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}$ „Backtranslate a non-interface trace $\mathtt{\overline{a}}$ to an L expression."

$$\frac{}{\langle\!\langle [\cdot] \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \mathit{42}}\ \text{(scct-backtrans-empty)}$$

$$\frac{\langle\!\langle \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}{\langle\!\langle \mathtt{Alloc}\ \ell\ \mathtt{s}; \mathtt{t}; \sigma \cdot \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}\ \text{(scct-backtrans-alloc)}$$

$$\frac{\langle\!\langle \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}{\langle\!\langle \mathtt{Dealloc}\ \ell; \mathtt{t}; \sigma \cdot \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}\ \text{(scct-backtrans-dealloc)}$$

$$\frac{\langle\!\langle \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}{\langle\!\langle \mathtt{Set}\ \ell\ \mathtt{n}\ \mathtt{v}; \mathtt{t}; \sigma \cdot \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}\ \text{(scct-backtrans-set)}$$

$$\frac{\langle\!\langle \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}{\langle\!\langle \mathtt{Get}\ \ell\ \mathtt{n}; \mathtt{t}; \sigma \cdot \mathtt{\overline{a}} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}\ \text{(scct-backtrans-get)}$$

$$\frac{}{\langle\!\langle \lightning; \mathtt{t}; \sigma \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \mathit{abort}()}\ \text{(scct-backtrans-abort)}$$

Figure 86: Trace-Based Backtranslation from $\mathrm{L_{scct}}$ backtranslation-events to L terms.

$\boxed{{}^{!}\langle\!\langle \mathtt{a} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}}_{\circlearrowleft} = e}$ „Yield an L expression from target !-interaction-event $\mathtt{a}$."

$$\frac{}{{}^{!}\langle\!\langle \mathtt{Start} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \mathit{42}}\ \text{(scct-backtrans-start)}$$

$$\frac{}{{}^{!}\langle\!\langle \mathtt{Ret}\ !\ \mathtt{v} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \langle\!\langle \langle\!\langle \mathtt{v} \rangle\!\rangle \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}}}\ \text{(scct-backtrans-ret-comptoctx)}$$

$\boxed{{}^{?}\langle\!\langle \mathtt{a} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = e}$ „Create an $\mathrm{L_{scct}}$ expression from target ?-interaction-event $\mathtt{a}$."

$$\frac{}{{}^{?}\langle\!\langle \mathtt{Call}\ ?\ \mathtt{foo}\ \mathtt{v} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \langle\!\langle \langle\!\langle \mathtt{v} \rangle\!\rangle \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}}}\ \text{(scct-backtrans-call-ctxtocomp)}$$

$$\frac{}{{}^{?}\langle\!\langle \mathtt{End}\ \mathtt{v} \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}} = \langle\!\langle \langle\!\langle \mathtt{v} \rangle\!\rangle \rangle\!\rangle^{\mathrm{L}\to\mathrm{L_{scct}}}}\ \text{(scct-backtrans-end-ctxtocomp)}$$

Figure 87: scct-backtranslation of interaction-events from $\mathrm{L_{scct}}$ specification events to L terms.

$$\frac{}{\vdash \varepsilon\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\varepsilon\text{)}$$

$$\frac{}{\vdash \mathtt{Alloc}\ \ell\ \mathtt{n}; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Alloc}\text{)}$$

$$\frac{}{\vdash \mathtt{Dealloc}\ \ell; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Dealloc}\text{)}$$

$$\frac{}{\vdash \mathtt{Get}\ \ell\ \mathtt{n}; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Get}\text{)}$$

$$\frac{}{\vdash \mathtt{Set}\ \ell\ \mathtt{n}; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Set}\text{)}$$

$$\frac{}{\vdash \mathtt{Binop}\ \mathtt{n}_1\ \mathtt{n}_2; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Binop}\text{)}$$

$$\frac{}{\vdash \mathtt{Branch}\ \mathtt{n}; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\mathtt{Branch}\text{)}$$

$$\frac{}{\vdash \lightning; \mathtt{t}; \sigma\ \text{non-int-}\mathtt{a}}\ \text{(scct-non-interface-}\lightning\text{)}$$

$$\frac{}{\vdash [\cdot]\ \text{non-int-}\mathtt{\overline{a}}}\ \text{(scct-non-interface-trace-empty)}$$

$$\frac{\vdash \mathtt{a}\ \text{non-int-}\mathtt{\overline{a}} \quad \vdash \mathtt{\overline{a}}\ \text{non-int-}\mathtt{\overline{a}}}{\vdash \mathtt{a} \cdot \mathtt{\overline{a}}\ \text{non-int-}\mathtt{\overline{a}}}\ \text{(scct-non-interface-trace-cons)}$$

Figure 88: Non-Interfacing events.

$$\boxed{{}^{!?}\langle\!\langle \mathtt{a}_1 \cdot \overline{\mathtt{a}} \cdot \mathtt{a}_2 \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_1; e_2; e_3}$$ „Construct an L ensemble of expressions from target-trace $\overline{\mathtt{a}}$ which starts with a ? event and ends in a !."

(scct-subtoplevel-backtrans)

$$\frac{\vdash \overline{\mathtt{a}}\ \text{non-int-}\overline{\mathtt{a}} \qquad {}^{!}\langle\!\langle \mathtt{a}_1 \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_1 \qquad \langle\!\langle \overline{\mathtt{a}} \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_2 \qquad {}^{?}\langle\!\langle \mathtt{a}_2 \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_3}{{}^{!?}\langle\!\langle \mathtt{a}_1 \cdot \overline{\mathtt{a}} \cdot \mathtt{a}_2 \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_1; e_2; e_3}$$

Figure 89: Interaction-Trace-Based Backtranslation from L_{scct} specification events to L terms.

$$\boxed{\langle\!\langle\!\langle \overline{\mathtt{a}^{ms}} \rangle\!\rangle\!\rangle^{\mathsf{L}_{scct}\to\mathsf{L}} = \Xi}$$ „Top-Level backtranslation of $\overline{\mathtt{a}}$ to L library Ξ."

(scct-backtrans-top-level)

$$\frac{\begin{array}{c} \overline{\mathtt{a}} = \overline{\mathtt{a}_0} \cdot \overline{\mathtt{a}^{comp}} \cdot \overline{\mathtt{a}_1} \qquad \vdash \overline{\mathtt{a}^{comp}}\ \text{non-int-}\overline{\mathtt{a}} \\ \overline{\mathtt{a}_0} = \mathtt{Start} \cdot \overline{\mathtt{a}_0'} \cdot \mathtt{Call}\ ?\ \mathtt{foo}\ \mathtt{v}_1 \qquad \overline{\mathtt{a}_1} = \mathtt{Ret}\ !\ \mathtt{v}_2 \cdot \overline{\mathtt{a}_0'} \cdot \mathtt{End}\ \mathtt{v}_3 \\ {}^{!?}\langle\!\langle \theta^*_\bullet\ (\overline{\mathtt{a}_0}) \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_0; e_0'; e_0'' \\ {}^{!?}\langle\!\langle \theta^*_\bullet\ (\overline{\mathtt{a}_1}) \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = e_1; e_1'; e_1'' \\ e = e_0; e_0'; e_0''; e_1; e_1'; e_1'' \end{array}}{\langle\!\langle\!\langle \overline{\mathtt{a}} \rangle\!\rangle\!\rangle^{\mathsf{L}_{scct}\to\mathsf{L}} = let\ main\ x : \mathbb{N} \to \mathbb{N} := e, [\cdot]}$$

Figure 90: Top-Level trace-based Backtranslation from L_{scct} trace $\overline{\mathtt{a}}$ to L context Ξ.

Lemma 88 (Backtranslation Correctness of $\mathtt{Start}$)**.** *If*

(a) $\Xi = \Xi_{\mathtt{ctx}} \bowtie [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(b) $\xi = [\![\xi]\!]^{\mathsf{L}\to\mathsf{L}_{scct}} = \text{dom}\ [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(c) $\sigma'' = \sigma \sqcap \sigma'$

(d) $\xi; \Xi; [\cdot]; \mathtt{comp}; 1; [\cdot]; [\cdot]; [\cdot] \triangleright \mathtt{call\ main}^{\sigma}\ 0^{\sigma'} \xrightarrow{\mathtt{Start};\mathtt{comp};\sigma''}_{ctx} \xi; \Xi; ([\cdot]; \mathtt{main}), [\cdot]; \mathtt{ctx}; 1; [\cdot]; [\cdot]; [\cdot] \triangleright \mathtt{e}_{\mathtt{main}}[0^{\sigma'}/\mathtt{x}]$

(e) ${}^{!}\langle\!\langle \theta_\bullet\ (\mathtt{Start}; \mathtt{comp}; \sigma'') \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = 42$

(f) $\xi; \Xi; [\cdot]; comp; [\cdot]; [\cdot]; [\cdot] \approx_{\emptyset;[\cdot]} \xi; \Xi; [\cdot]; \mathtt{comp}; 1; [\cdot]; [\cdot]; [\cdot]$

(g) $\Xi = \{main \mapsto 42; e_{main}\} \bowtie \Xi_{comp}$

Then $\exists n$

(i) $\xi; \Xi; [\cdot]; comp; [\cdot]; [\cdot]; [\cdot] \triangleright call\ main\ 0 \xrightarrow{Start;comp}_{ctx}{}^{n}\ \xi; \Xi; ([\cdot]; main), [\cdot]; ctx; [\cdot]; [\cdot]; [\cdot] \triangleright e_{main}[0/x]$

(ii) $\xi; \Xi; ([\cdot]; main), [\cdot]; ctx; [\cdot]; [\cdot]; [\cdot] \multimap_{\emptyset} \xi; \Xi; ([\cdot]; \mathtt{main}), [\cdot]; \mathtt{ctx}; 1; [\cdot]; [\cdot]$

(iii) $Start \approxeq^*_{\emptyset} \mathtt{Start}$

Proof. With Rules $e-$ctx$-$**call**$-$main, $en-$refl and $en-$trans$-$important one can conclude:

(H_1) $\xi; \Xi; [\cdot]; comp; [\cdot]; [\cdot]; [\cdot] \triangleright call\ main\ 0 \xrightarrow{Start;comp}_{\mathtt{ctx}}{}^{1}\ \xi; \Xi; ([\cdot]; main), [\cdot]; ctx; [\cdot]; [\cdot]; [\cdot] \triangleright let\ \_ = 42\ in\ e_{main}[0/x]$

Now following with Rules $e-\textbf{let}-\textbf{f}$, en − refl and en − trans−unimportant:

(H_2) $\xi; \Xi; ([\cdot]; main), [\cdot]; comp; [\cdot]; [\cdot]; [\cdot] \triangleright let\ \_ = 42\ in\ e_{main}[0/x] \xrightarrow{\varepsilon}_{\text{ctx}}{}^{1}\ \xi; \Xi; ([\cdot]; main), [\cdot]; ctx; [\cdot]; [\cdot]; [\cdot] \triangleright e_{main}[0/x]$

Combine Assumptions (H_1) and (H_2) with Rules en−refl to en−trans−unimportant:

(H_3) $\xi; \Xi; [\cdot]; comp; [\cdot]; [\cdot]; [\cdot] \triangleright call\ main\ 0 \xrightarrow{Start;comp}_{\text{ctx}}{}^{2}\ \xi; \Xi; ([\cdot]; main), [\cdot]; ctx; [\cdot]; [\cdot]; [\cdot] \triangleright e_{main}[0/x]$

Instantiate $n = 2$ and Assumption (H_3) solves Goal (i).
Goal (ii) by Rules scct-empty-memstate-eq, scct-empty-kontstack-eq, state-qe and cfstate-qe and assumptions (b) and (f).
Goal (iii) by Rule scct-start-event-eq. ☐

Lemma 89 (Backtranslation Correctness of $\texttt{Ret}$)**.** *If*

(a) $\Omega = \xi; \Xi; (\mathtt{K}; \mathtt{foo}), \overline{\mathtt{K}}; \mathtt{comp}; 1; \Psi$

(b) $\Xi = \Xi_{\mathtt{ctx}} \bowtie [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(c) $\xi = [\![\xi]\!]^{\mathsf{L}\to\mathsf{L}_{scct}} = \text{dom}\ [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(d) $\Omega \triangleright \mathtt{K}_{\mathtt{component}}[\mathtt{return\ v}^{\sigma}] \xrightarrow{\mathtt{Ret\ !\ v;comp;}\sigma}_{ctx} \Omega' \triangleright \mathtt{K}[\mathtt{v}^{\sigma}]$

(e) $\mathtt{K}_{\mathtt{component}} = [\![K_{component}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(f) $\langle\!\langle\theta_{\bullet}\,(\mathtt{Ret\ !\ v;comp;}\sigma)\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = \langle\!\langle\mathtt{v}\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}$

(g) $\Omega \approx_{\delta_{scct}} \Omega$

(h) $\Omega = \xi; \Xi; (K, foo), \overline{K}; comp; \Psi$

(i) $\Xi = \{main \mapsto e_{main}\} \bowtie \Xi_{comp}$

then $\exists n\ \overline{a}$*,*

(i) $\Omega \triangleright K_{component}[return\ \langle\!\langle\mathtt{v}\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}] \xrightarrow{\overline{a}}{}^{n}_{ctx}\ \xi; \Xi; \overline{K}; ctx; \Psi \triangleright K[\langle\!\langle\mathtt{v}\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}]$

(ii) $\xi; \Xi; \overline{K}; ctx; \Psi \multimap_{\delta_{scct}} \Omega'$

(iii) $\overline{a} \approxeq^{*}_{\delta} \mathtt{Ret\ !\ v}$

Proof. Instantiate the existentials with $n = 1$ and $\overline{a} = Ret\ !\ \langle\!\langle\mathtt{v}\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}_{\eth}$. Let $v = \langle\!\langle\mathtt{v}\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}_{\eth}$. For Goal (i) use Rule e − ctx − **return** − notsame.

(H_1) $\xi; \Xi; (K; foo), \overline{K}; comp; \Psi \triangleright K_{component}[return\ v] \xrightarrow{Ret\ ?\ v}_{\text{ctx}}{}^{1}\ \xi; \Xi; \overline{K}; comp; \Psi \triangleright K[v]$

This solves Goal (i).
By inverting Assumption (d) we have $\Omega.\Psi = \Omega'.\Psi$. Invert Assumption (g):

(H_2) $1 \neq 0$

(H_3) $comp = \mathtt{comp}$

(H_4) $\Psi \approx_{\delta_{scct}} \Psi$

(H_5) $\xi; \Xi; (K; foo), \overline{K} \approx_{\delta_{scct}} \xi; \Xi; (\mathtt{K}; \mathtt{foo}), \overline{\mathtt{K}}$

From this, Goal (ii) follows easily.
Goal (iii) by Rule scct-ret-event-eq. □

Lemma 90 (Middle of Backtranslation Correctness)**.** *If*

(a) $\Omega = \xi; \Xi; \overline{\mathtt{K}}; \mathtt{ctx}; \mathtt{m}; \Psi$

(b) $\Xi = \Xi_{\mathtt{ctx}} \bowtie [\![\Xi_{comp}]\!]^{\mathtt{L}\to\mathtt{L}_{scct}}$

(c) $\xi = [\![\xi]\!]^{\mathtt{L}\to\mathtt{L}_{scct}} = \text{dom } [\![\Xi_{comp}]\!]^{\mathtt{L}\to\mathtt{L}_{scct}}$

(d) $\Omega \triangleright \mathtt{e}\gamma \xrightarrow{\overline{\mathtt{a}}}_{ctx}{}^{n} \Omega' \triangleright \mathtt{e}'\gamma'$

(e) $\langle\!\langle \theta^{*}_{\bullet} (\overline{\mathtt{a}}) \rangle\!\rangle^{\mathtt{L}\to\mathtt{L}_{scct}} = e$

(f) $\vdash \overline{\mathtt{a}}\,\text{non-int-}\,\overline{\mathtt{a}}$

(g) $\Omega \multimap_{\delta_{scct}} \Omega$

(h) $\Omega = \xi; \Xi; \overline{K}; ctx; \Psi$

(i) $\Xi = \Xi_{ctx} \bowtie \Xi_{comp}$

then $\exists n'\ \overline{a}$,

(i) $\Omega \triangleright K[e] \xrightarrow{\overline{a}}_{ctx}{}^{n'} \xi; \Xi; \overline{K}; \Psi' \triangleright K[42]$

(ii) $\Omega' \multimap_{\delta_{scct}} \xi; \Xi; \overline{K}; \Psi'$

(iii) $\overline{a} \approxeq^{*}_{\delta_{scct}} [\cdot]$

Proof. Similar to Lemma 71. □

Lemma 91 (Backtranslation Correctness of $\mathtt{Call}\ ?$)**.** *If*

(a) $\Omega = \xi; \Xi; \overline{\mathtt{K}}; \mathtt{ctx}; \mathtt{n}; \Psi$

(b) $\overline{\mathtt{K}} = (\mathtt{K}'; \mathtt{bar}), \overline{\mathtt{K}}'$

(c) $\Xi = \Xi_{\mathtt{ctx}} \bowtie [\![\Xi_{comp}]\!]^{\mathtt{L}\to\mathtt{L}_{scct}}$

(d) $\xi = [\![\xi]\!]^{\mathtt{L}\to\mathtt{L}_{scct}} = \text{dom } [\![\Xi_{comp}]\!]^{\mathtt{L}\to\mathtt{L}_{scct}}$

(e) $\Omega \triangleright \mathtt{K}\left[\mathtt{call\ foo}^{\sigma}\ \mathtt{v}^{\sigma'}\right] \xrightarrow{\mathtt{Call\ ?\ foo\ v;ctx;}\sigma'}_{ctx}{}^{2} \xi; \Xi; (\mathtt{K}; \mathtt{foo}), \overline{\mathtt{K}}; \mathtt{comp}; 1; \Psi \triangleright [\![e_{foo}]\!]^{\mathtt{L}\to\mathtt{L}_{scct}}[\mathtt{v}^{\sigma'}/\mathtt{y}]$

(f) ${}^{?}\langle\!\langle \theta_{\bullet} (\mathtt{Call\ ?\ foo\ v}) \rangle\!\rangle^{\mathtt{L}\to\mathtt{L}_{scct}} = call\ foo\ \langle\!\langle \mathtt{v} \rangle\!\rangle^{\mathtt{L}\to\mathtt{L}_{scct}}$

(g) $\Omega \multimap_{\delta} \Omega$

(h) $\Omega = \xi; \Xi; \overline{K}; ctx; \Psi$

(i) $let\ foo\ y := e_{foo} \in \Xi_{comp}$

(j) $\Xi = \Xi_{ctx} \bowtie \Xi_{comp}$

then $\exists n\ \overline{a}$,

(i) $\Omega \triangleright K[\mathit{call\ foo}\ \langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}] \xrightarrow{\overline{a}}_{ctx}{}^{n}\ \xi;\Xi;(K;foo),\overline{K};comp;\Psi \triangleright e_{foo}[v/y]$

(ii) $\xi;\Xi;(K,foo),\overline{K};comp;\Psi \approx_{\delta} \xi;\Xi;(\mathtt{K};\mathtt{foo}),\overline{\mathtt{K}};\mathtt{comp};\mathtt{1};\Psi$

(iii) $\overline{a} \approxeq^{*}_{\delta} \mathtt{Call\ ?\ foo\ v;ctx;}\sigma'$

Proof. Let $v = \langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle_{\mho}^{\mathsf{L}\to\mathsf{L}_{scct}}$. Instantiate the existentials of the goal $n = 1$ and $\overline{a} = \mathit{Call\ ?\ foo\ v;ctx},[\cdot]$.
Goal (i) follows immediately from Rule $e-\text{ctx}-$**call**$-$notsame.
Goal (ii) follows immediately from Rule scct-state-eq.
Goal (iii) follows immediately from Rules scct-cons-trace-eq and scct-call-event-eq. □

Lemma 92 (Backtranslation Correctness of $\mathtt{End}$). *If*

(a) $\Xi = \Xi_{\mathtt{ctx}} \bowtie [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(b) $\xi = [\![\xi]\!]^{\mathsf{L}\to\mathsf{L}_{scct}} = \text{dom}\ [\![\Xi_{comp}]\!]^{\mathsf{L}\to\mathsf{L}_{scct}}$

(c) $\xi;\Xi;([\cdot];\mathtt{main}),[\cdot];\mathtt{ctx};\mathtt{n};\Psi\triangleright\mathtt{K}[\mathtt{return\ v}^{\sigma}] \xrightarrow{\mathtt{End\ v;ctx;}\sigma}_{ctx}{}^{2} \xi;\Xi;[\cdot];\mathtt{comp};\mathtt{1};\Psi\triangleright \mathtt{v}^{\sigma}$

(d) ${}^{?}\langle\!\langle \theta_{\bullet}\ (\mathtt{End\ v;ctx;}\sigma) \rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}} = \mathit{return}\ \langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}$

(e) $\xi;\Xi;([\cdot];main),[\cdot];ctx;\Psi \multimap_{\delta} \xi;\Xi;([\cdot];\mathtt{main}),[\cdot];\mathtt{ctx};\mathtt{n};\Psi$

(f) $\Xi = \Xi_{ctx} \bowtie \Xi_{comp}$

then $\exists n\ \overline{a}$,

(i) $\Omega \triangleright K[\mathit{return}\ \langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}] \xrightarrow{\overline{a}}_{ctx}{}^{n}\ \xi;\Xi;[\cdot];comp;\Psi \triangleright \langle\!\langle\!\langle \mathtt{v} \rangle\!\rangle\!\rangle^{\mathsf{L}\to\mathsf{L}_{scct}}$

(ii) $\xi;\Xi;[\cdot];comp;\Psi \approx_{\delta'} \xi;\Xi;[\cdot];\mathtt{comp};\mathtt{1};\Psi$

(iii) $\overline{a} \approxeq^{*}_{\delta'} \mathtt{End\ v;ctx;}\sigma$

Proof. Instantiate the goal with $n = 1$ and $\overline{a} = \mathit{End}\ \ v;ctx$. Goal (i) by Rule $e-$ ctx $-$ **return**$-$main.
Goal (ii) by inversion on Assumption (e).
Goal (iii) by Rules scct-cons-trace-eq and scct-end-event-eq. □

Lemma 93 (Backtranslation Correctness). *If*

(a) $\Omega \triangleright \mathtt{call\ main\ 0} \xrightarrow{\overline{\mathtt{a}}}{}^{*}_{ctx}\ \Omega' \triangleright \mathtt{f}_{\lightning}$

(b) $\langle\!\langle\!\langle \overline{\mathtt{a}} \rangle\!\rangle\!\rangle^{\mathsf{L}_{scct}\to\mathsf{L}} = \Xi_{ctx}$

(c) $\Omega \approx_{\emptyset} \Omega$

(d) $\Omega = \mathtt{foo},[\cdot];\Xi;[\cdot];\mathtt{comp};\mathtt{1};[\cdot];[\cdot];[\cdot]$

(e) $\Omega = \xi;\Xi;\overline{K};comp;\Psi$

(f) $\Xi = \Xi_{ctx} \bowtie \Xi_{comp}$

then $\exists \delta_e\ \Omega'_e\ \overline{a}_e$,

(i) $\mathit{\Omega} \triangleright \mathit{call\ main\ 0} \xrightarrow{\overline{a}_e}{}^{*}_{ctx} \mathit{\Omega}'_e \triangleright \langle\langle\langle \mathtt{f}_{\lightning} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{scct}}$

(ii) $\mathit{\Omega}'_e \approx_{\delta_e} \Omega'$

(iii) $\overline{a}_e \approxeq^{*}_{\delta_e} \overline{\mathtt{a}}$

Proof. Inverting Assumption (b) gives:

(H_1) $\overline{\mathtt{a}} = \overline{\mathtt{a}_0} \cdot \overline{\mathtt{a}_{\mathtt{comp}}} \cdot \overline{\mathtt{a}_1}$

(H_2) $\overline{\mathtt{a}_0} = \mathtt{Start} \cdot \overline{\mathtt{a}'_0} \cdot \mathtt{Call\ ?\ foo\ v}$

(H_3) $\overline{\mathtt{a}_1} = \mathtt{Ret\ !\ v'} \cdot \overline{\mathtt{a}_1}' \cdot \mathtt{End\ v''}$

(H_4) $\vdash \overline{\mathtt{a}_{\mathtt{comp}}}$ non-int-$\overline{\mathtt{a}}$

(H_5) ${}^{!?}\langle\langle \theta^{*}_{\bullet}\left(\overline{\mathtt{a}_0}\right) \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e_0};\mathit{e'_0};\mathit{e''_0}$

(H_6) ${}^{!?}\langle\langle \theta^{*}_{\bullet}\left(\overline{\mathtt{a}_1}\right) \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e_1};\mathit{e'_1};\mathit{e''_1}$

By definition of $\theta_{\bullet}\left(\cdot\right)$, we have:

(H_7) $\theta^{*}_{\bullet}\left(\overline{\mathtt{a}_0}\right) = \mathtt{Start} \cdot \mathtt{Call\ ?\ foo\ v}$

(H_8) $\theta^{*}_{\bullet}\left(\overline{\mathtt{a}_1}\right) = \mathtt{Ret\ !\ v'} \cdot \mathtt{End\ v''}$

Invert Assumptions (H_5) and (H_6):

(H_9) ${}^{!}\langle\langle \mathtt{Start} \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e_0}$

(H_{10}) $\langle\langle [\cdot] \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e'_0}$

(H_{11}) ${}^{?}\langle\langle \mathtt{Call\ ?\ foo\ v} \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e''_0}$

(H_{12}) ${}^{!}\langle\langle \mathtt{Ret\ !\ v'} \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e_1}$

(H_{13}) $\langle\langle [\cdot] \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e'_1}$

(H_{14}) ${}^{?}\langle\langle \mathtt{End\ v''} \rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}} = \mathit{e''_1}$

Invert Assumptions (H_9) to (H_{14}):

(H_{15}) $\mathit{e_0} = \mathit{42}$

(H_{16}) $\mathit{e'_0} = \mathit{42}$

(H_{17}) $\mathit{e''_0} = \mathit{call\ foo} \langle\langle\langle \mathtt{v} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$

(H_{18}) $\mathit{e_1} = \langle\langle\langle \mathtt{v'} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$

(H_{19}) $\mathit{e'_1} = \mathit{42}$

(H_{20}) $\mathit{e''_1} = \langle\langle\langle \mathtt{v''} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$

Let $\mathit{v} = \langle\langle\langle \mathtt{v} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$ and $\mathit{v'} = \langle\langle\langle \mathtt{v'} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$ and $\mathit{v''} = \langle\langle\langle \mathtt{v''} \rangle\rangle\rangle^{\mathtt{L}\to\mathtt{L}_{\mathtt{scct}}}$.
By Rule $e -$ ctx $-$ **call**$-$main, we have

(H_{21}) $\mathit{\Omega} \triangleright \mathit{call\ main\ 0} \xrightarrow{\mathit{Start}}_{\text{ctx}} \mathit{foo}, [\cdot]; \mathit{\Xi}; (\mathit{main}; [\cdot]), [\cdot]; \mathit{comp}; \mathit{\Psi} \triangleright \mathit{42}; \mathit{42}; \mathit{call\ foo\ v}; \mathit{v'}; \mathit{42}; \mathit{v''}$

It's easy to see that this reduces further to:

(H_{22}) $\Omega \triangleright 42; 42; call\ foo\ v; v'; 42; v'' \xrightarrow{Call\ ?\ foo\ v}_{\mathrm{ctx}}{}^{3} foo, [\cdot]; \Xi; ([\cdot]; v'; 42; v''), (main; [\cdot]), [\cdot]; comp; \Psi \triangleright e_{foo}$

Now by backward simulation
then the other BT correctness lemmas $\square$

Theorem 17 (TMS Relation Correctness for $[\![\bullet]\!]^{\mathrm{L}\to\mathrm{L_{scct}}}$)**.**

$$\textit{if } \theta^{*}_{\delta_{MS}}\left(\sigma_{\theta^{*}_{\delta_{\mathtt{MS}}}(\bullet)}(\mathrm{tms})\right) = \mathrm{tms} \ \textit{ and } \sigma_{\approx^{*}_{\delta;\mathsf{x}}}(\sigma_{\theta^{*}_{\delta_{\mathtt{MS}}}(\bullet)}(\mathrm{tms})) \approx^{*}_{\delta;\mathsf{x}} \sigma_{\theta^{*}_{\delta_{\mathbf{MS}}}(\bullet)}(\mathrm{tms})$$
$$\textit{then } \theta^{*}_{\delta_{MS}}\left(\sigma_{\approx^{*}_{\delta;\mathsf{x}}}(\sigma_{\theta^{*}_{\delta_{MS}}(\bullet)}(\mathrm{tms}))\right) = \mathrm{tms}$$

Proof. unfolding. $\square$

Definition 44 (L Robust Satisfaction)**.** *We write* $\Xi_{comp} \vDash_R \pi$ *for If*

(a) $prog\ \Xi_{ctx}\ \Xi_{comp} \stackrel{\overline{a}}{\Longrightarrow} \Omega \triangleright f_{\lightning}$

Then $\exists \delta_{sCCT}$

(i) $\theta_{\delta_{sCCT}}(\overline{a}) \in \pi$

Definition 45 ($\mathrm{L_{scct}}$ Robust Satisfaction)**.** *We write* $\Xi_{\mathtt{comp}} \vDash_R \pi$ *for If*

(a) $\mathtt{prog}\ \Xi_{\mathtt{ctx}}\ \Xi_{\mathtt{comp}} \stackrel{\overline{\mathtt{a}}}{\Longrightarrow} \Omega \triangleright \mathtt{f}_{\lightning}$

Then $\exists \delta_{sCCT}$

(i) $\theta_{\delta_{sCCT}}(\overline{\mathtt{a}}) \in \pi$

Theorem 18 (Robust Strict Cryptographic Constant Time Preservation)**.**

(i) $\vdash [\![\bullet]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} : \lceil sCCT \rceil$

Proof. $\square$

Theorem 19 (Robust Memory Safety and Strict Cryptographic Constant Time Preservation)**.**

(i) $\vdash [\![[\![[\![\bullet]\!]^{\mathrm{L}_{tms}\to\mathrm{L}_{ms}}]\!]^{\mathrm{L}_{ms}\to\mathrm{L}}]\!]^{\mathrm{L}\to\mathrm{L}_{scct}} : \lceil \mathrm{msafe} \rceil \cap \lceil sCCT \rceil$

Proof. Note from **??** 12 (Robust Memory Safety Preservation) and **??** 18 (Robust Strict Cryptographic Constant Time Preservation):

(H_1) $\vdash [\![[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}} : \lceil \mathrm{msafe} \rceil$

(H_2) $\vdash [\![\bullet]\!]^{\mathrm{L}\to\mathrm{L_{scct}}} : \lceil \mathtt{sCCT} \rceil$

By Lemma 8 (Sequential Composition with RTP) using Assumptions (H_1) and (H_2) we have:

(H_3) $\vdash [\![[\![[\![\bullet]\!]^{\mathrm{L_{tms}}\to\mathrm{L_{ms}}}]\!]^{\mathrm{L_{ms}}\to\mathrm{L}}]\!]^{\mathrm{L}\to\mathrm{L_{scct}}} : \lceil \mathrm{msafe} \rceil \cap \lceil \mathtt{sCCT} \rceil$

This solves our goal. $\square$

4.9 Low-Level Language

The next language is „low-level", because it extends the previous one with speculation.

$$
\begin{array}{rl}
\textit{Final Result } \mathsf{f} ::= \mathsf{v}^{\sigma} \mid \mathsf{x}^{\sigma} & \textit{May be a Result } \mathsf{f}_{\not{\iota}} ::= \mathsf{f} \mid \mathsf{stuck} \\
\textit{Expressions } \mathsf{e} ::= & \mathsf{f}_{\not{\iota}} \mid \mathsf{e}_1 \oplus \mathsf{e}_2 \mid \mathsf{x}[\mathsf{e}] \mid \mathsf{getDIT}\ \mathsf{x}\ \mathsf{in}\ \mathsf{e} \mid \mathsf{setDIT}\ \mathsf{e} \\
& \mid \mathsf{let}\ \mathsf{x} = \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2 \mid \mathsf{x}[\mathsf{e}_1] \leftarrow \mathsf{e}_2 \\
& \mid \mathsf{let}\ \mathsf{x} = \mathsf{new}\ \mathsf{e}_1\ \mathsf{in}\ \mathsf{e}_2 \mid \mathsf{delete}\ \mathsf{x} \\
& \mid \mathsf{return}\ \mathsf{e} \mid \mathsf{call}\ \mathsf{foo}\ \mathsf{e} \mid \mathsf{ifz}\ \mathsf{e}_1\ \mathsf{then}\ \mathsf{e}_2\ \mathsf{else}\ \mathsf{e}_3 \\
& \mid \mathsf{abort}() \mid \mathsf{x}\ \mathsf{is}\ \text{☣} \\
& \mid \langle \mathsf{e}_1, \mathsf{e}_2 \rangle \mid \pi_1 \mathsf{e} \mid \pi_2 \mathsf{e} \mid \mathsf{e}\ \mathsf{has}\ \tau \\
& \text{where } \oplus \in \{+, -, \times, <, /\} \\
\textit{Functions } \mathsf{F} ::= \mathsf{let}\ \mathsf{foo}\ \mathsf{x} := \mathsf{e} & \textit{Types } \tau ::= \mathbb{N} \mid \mathbb{N} \times \mathbb{N} \\
\textit{Values } \mathsf{v} ::= \mathsf{n} \in \mathbb{N} & \textit{References } \ell \in \mathbb{N} \\
\textit{Eval.Ctx. } \mathsf{K} ::= & [\cdot] \mid \mathsf{K} \oplus \mathsf{e} \mid \mathsf{v} \oplus \mathsf{K} \mid \mathsf{x}[\mathsf{K}] \mid \mathsf{let}\ \mathsf{x} = \mathsf{K}\ \mathsf{in}\ \mathsf{e} \\
& \mid \mathsf{x}[\mathsf{K}] \leftarrow \mathsf{e} \mid \mathsf{x}[\mathsf{v}] \leftarrow \mathsf{K} \mid \mathsf{let}\ \mathsf{x} = \mathsf{new}\ \mathsf{K}\ \mathsf{in}\ \mathsf{e} \\
& \mid \mathsf{ifz}\ \mathsf{K}\ \mathsf{then}\ \mathsf{e}_1\ \mathsf{else}\ \mathsf{e}_2 \mid \mathsf{call}\ \mathsf{foo}\ \mathsf{K} \mid \mathsf{return}\ \mathsf{K} \\
& \mid \langle \mathsf{K}, \mathsf{e} \rangle \mid \langle \mathsf{v}, \mathsf{K} \rangle \mid \pi_1 \mathsf{K} \mid \pi_2 \mathsf{K} \mid \mathsf{K}\ \mathsf{has}\ \tau \mid \mathsf{barrier} \\
& \mid \mathsf{ifz}\ \mathsf{K}\ \mathsf{then}\ \mathsf{e}_1\ \mathsf{else}\ \mathsf{e}_2 \mid \mathsf{setDIT}\ \mathsf{K} \\
\textit{Variables } \mathsf{x} \mid \mathsf{y} \mid \mathsf{foo} \mid \ldots & \textit{Poison } \rho ::= \square \mid \text{☣} \\
\textit{Sandbox Tag } \mathsf{t} ::= \mathsf{ctx} \mid \mathsf{comp} & \\
\textit{Typing.Env. } \Gamma ::= [\cdot] \mid \Gamma, \mathsf{x} : \tau & \textit{Store } \Delta ::= [\cdot] \mid \mathsf{x} \mapsto (\ell; \mathsf{t}; \rho; \mathsf{n}), \Delta \\
\textit{Communication } \mathsf{c} ::= ? \mid\ ! \mid \varnothing & \textit{Heaps } \mathsf{H} ::= [\cdot] \mid \mathsf{H} :: \mathsf{n} \\
\textit{Cont. Stack } \overline{\mathsf{K}} ::= [\cdot] \mid (\mathsf{K}; \mathsf{foo}), \overline{\mathsf{K}} & \textit{Library } \Xi ::= [\cdot] \mid \mathsf{F}, \Xi \\
\textit{Relevant } \xi ::= [\cdot] \mid \mathsf{foo}, \xi & \textit{State } \Omega ::= \Phi; \mathsf{t}; \mathsf{n}; \Psi \\
\textit{Flow State } \Phi ::= \xi; \Xi; \overline{\mathsf{K}} & \textit{Memory State } \Psi ::= \mathsf{H}^{\mathsf{ctx}}; \mathsf{H}^{\mathsf{comp}}; \Delta \\
\textit{Programs } \mathsf{prog}\ \Xi_{\mathsf{ctx}}\ \Xi_{\mathsf{comp}} & \textit{Substitutions } \gamma ::= [\mathsf{v}/\mathsf{x}], \gamma \mid [\cdot] \\
\textit{Security Tag } \sigma ::= \text{🔒}_{\mathsf{vX}} \mid \text{🔓} & \\
\textit{Leak Tag } \mathsf{vX} ::= \mathsf{NONE} \mid \mathsf{PHT} &
\end{array}
$$

Figure 91: Syntax of $\mathrm{L}_{\text{👻}}$

The only difference between $\mathrm{L}_{\mathrm{scct}}$ and $\mathrm{L}_{\text{👻}}$ is speculative semantics.

4.9.1 Dynamic Semantics

$$\boxed{\mathrm{dom}\ \Xi = \mathsf{foo}, \ldots, \mathsf{bar}} \quad \text{„Collect function names.''}$$

$$\frac{}{\mathrm{dom}\ [\cdot] = [\cdot]}\ (\Xi\text{-dom-}[\cdot]) \qquad \frac{\mathrm{dom}\ \Xi = \mathsf{D}}{\mathrm{dom}\ (\mathsf{let\ foo\ x} : \tau_\lambda := \mathsf{e}), \Xi = \mathsf{foo}, \mathsf{D}}\ (\Xi\text{-dom-cons})$$

$$\boxed{\Xi \equiv \Xi_1 \cup \Xi_2} \quad \text{„Merging } \mathsf{L}_{\text{ghost}} \text{ libraries.''}$$

$$\frac{\Xi = \Xi_1}{\Xi \equiv \Xi_1 \cup [\cdot]}\ (\text{lib-merge-empty}) \qquad \frac{\Xi \equiv \Xi_1 \cup \Xi_2}{\mathsf{F}, \Xi \equiv \Xi_1 \cup \mathsf{F}, \Xi_2}\ (\text{lib-merge-cons})$$

$$\boxed{\Xi_1 \bowtie \Xi_2 = \Xi} \quad \text{„Syntactically linking } \mathsf{L}_{\text{ghost}} \text{ libraries.''}$$

$$\frac{\mathrm{dom}\ \Xi_1 \cap \mathrm{dom}\ \Xi_2 = \emptyset \qquad \Xi \equiv \Xi_1 \cup \Xi_2}{\Xi_1 \bowtie \Xi_2 = \Xi}\ (\text{syntactic-plugging})$$

Figure 92: $\mathsf{L}_{\text{ghost}}$ plugging of libraries and collecting of function names.

$$\begin{aligned}
\textit{Base Events}\ \mathsf{a_b} ::=\ & \mathsf{Alloc}\ \ell\ \mathsf{v} \mid \mathsf{Dealloc}\ \ell \mid \mathsf{Get}\ \ell\ \mathsf{v} \mid \mathsf{Set}\ \ell\ \mathsf{v}\ \mathsf{v}' \mid \lightning \\
& \mid \mathsf{Call}\ \mathbf{c}\ \mathsf{foo}\ \mathsf{v} \mid \mathsf{Ret}\ \mathbf{c}\ \mathsf{v} \mid \mathsf{Start} \mid \mathsf{End}\ \mathsf{v} \\
& \mid \mathsf{Branch}\ \mathsf{n} \mid \mathsf{Binop}\ \mathsf{n}\ \mathsf{m} \\
& \mid \mathsf{iGet}\ \ell\ \mathsf{v} \mid \mathsf{iSet}\ \ell\ \mathsf{v}\ \mathsf{v}' \\
& \mid \mathsf{Spec} \mid \mathsf{Rlb} \mid \mathsf{Barrier} \\
\textit{Events}\ \mathsf{a} ::=\ & \varepsilon \mid \mathsf{a_b}; \mathsf{t}; \sigma
\end{aligned}$$

Figure 93: Events of $\mathsf{L}_{\text{ghost}}$.

Rest similar to [3].

4.10 Robustly Preserving spec

4.10.1 Compiler

$\boxed{[\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = \mathsf{e}'}$ „Compile $\mathrm{L}_{\mathrm{scct}}$ expression e to $\mathrm{L}_{\text{ghost}}$ expression e'."

$\boxed{[\![[\mathsf{v}/\mathsf{x}]]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = [\mathsf{v}/\mathsf{x}]}$ „Compile $\mathrm{L}_{\mathrm{scct}}$ substitution to $\mathrm{L}_{\text{ghost}}$ substitution."

$\boxed{[\![\xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = \xi'}$ „Compile $\mathrm{L}_{\mathrm{scct}}$ component library to $\mathrm{L}_{\text{ghost}}$ component library."

$$
\begin{aligned}
[\![\mathsf{f}_{\lightning}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{f}_{\lightning}\\
[\![\mathtt{call\ foo\ e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{call}\ [\![\mathtt{foo}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\ [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathtt{return\ e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{return}\ [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathsf{e}_1 \oplus \mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= [\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} \oplus [\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathsf{x}[\mathsf{e}]]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{x}[[\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}]\\
[\![\mathtt{let\ x} = \mathsf{e}_1\ \mathtt{in}\ \mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{let\ x} = [\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\ \mathsf{in}\ [\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathsf{x}[\mathsf{e}_1] \leftarrow \mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{x}[[\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}] \leftarrow [\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathtt{let\ x} = \mathtt{new}\ \mathsf{e}_1\ \mathtt{in}\ \mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{let\ x} = \mathsf{new}\ [\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\ \mathsf{in}\ [\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathtt{delete\ x}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{delete\ x}\\
[\![\mathtt{x\ is}\ ☣]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{x\ is}\ ☣\\
[\![(\mathsf{e}_1;\mathsf{e}_2)]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= ([\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}};[\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}})\\
[\![\pi_1\ \mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \pi_1\ [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\pi_2\ \mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \pi_2\ [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
[\![\mathtt{e\ has}\ \tau]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\ \mathsf{has}\ \tau\\
[\![\mathtt{ifz}\ \mathsf{e}_1\ \mathtt{then}\ \mathsf{e}_2\ \mathtt{else}\ \mathsf{e}_3]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{ifz}\ [\![\mathsf{e}_1]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
&\quad\ \mathsf{then\ barrier};[\![\mathsf{e}_2]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\\
&\quad\ \mathsf{else\ barrier};[\![\mathsf{e}_3]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}
\end{aligned}
$$

$$
[\![[\mathsf{v}/\mathsf{x}]]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = [\,[\![\mathsf{v}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}/[\![\mathsf{x}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}\,]
$$

$$
\begin{aligned}
[\![[\cdot]]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= [\cdot]\\
[\![\mathtt{foo},\xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= \mathsf{foo},[\![\xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}
\end{aligned}
$$

Figure 94: Compiler from $\mathrm{L}_{\mathrm{scct}}$ to $\mathrm{L}_{\text{ghost}}$.

$\boxed{[\![\mathrm{F}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = \mathsf{F}}$ „Compile $\mathrm{L}_{\mathrm{scct}}$ procedures to $\mathrm{L}_{\text{ghost}}$ procedures."

$$
[\![\mathtt{let\ foo\ x} := \mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = [\![\mathsf{e}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}
$$

$\boxed{[\![\Xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} = \Xi}$ „Compile $\mathrm{L}_{\mathrm{scct}}$ libraries to $\mathrm{L}_{\text{ghost}}$ libraries."

$$
\begin{aligned}
[\![[\cdot]]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= [\cdot]\\
[\![\mathrm{F},\Xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}} &= [\![\mathrm{F}]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}},[\![\Xi]\!]^{\mathrm{L}_{\mathrm{scct}}\to\mathrm{L}_{\text{ghost}}}
\end{aligned}
$$

Figure 95: Compiler from $\mathrm{L}_{\mathrm{scct}}$ components to $\mathrm{L}_{\text{ghost}}$ components.

Theorem 20 (Robust Speculative Safety Preservation)**.**

(i) $\vdash [\![\bullet]\!]^{\mathtt{L}_{scct} \to \mathtt{L}_{👻}} : \lceil \text{spec} \rceil$

Proof. Analogous to **??** 18. $\square$

Theorem 21 (Robust Memory Safety, Strict Cryptographic Constant Time, and Speculation Safety Preservation)**.**

(i) $\vdash [\![[\![[\![[\![\bullet]\!]^{\mathtt{L}_{tms} \to \mathtt{L}_{ms}}]\!]^{\mathtt{L}_{ms} \to \mathtt{L}}]\!]^{\mathtt{L} \to \mathtt{L}_{scct}}]\!]^{\mathtt{L}_{scct} \to \mathtt{L}_{👻}} : \lceil \text{msafe} \rceil \cap \lceil \text{sCCT} \rceil \cap \lceil \text{spec} \rceil$

Proof. By Lemma 8 (Sequential Composition with RTP), **??** 12 (Robust Memory Safety Preservation), **??** 18 (Robust Strict Cryptographic Constant Time Preservation), and **??** 20 (Robust Speculative Safety Preservation). $\square$

References

[1] Gilles Barthe, Juan Manuel Crespo, and César Kunz. Relational verification using product programs. In Michael Butler and Wolfram Schulte, editors, *FM 2011: Formal Methods*, pages 200–214, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[2] Michael R. Clarkson and Fred B. Schneider. Hyperproperties. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium, CSF 2008, Pittsburgh, Pennsylvania, USA, 23-25 June 2008*, pages 51–65. IEEE Computer Society, 2008.

[3] Marco Guarnieri and Marco Patrignani. Exorcising spectres with secure compilers. *CoRR*, abs/1910.08607, 2019.