Secure Information Flow for Concurrent Programs Under Total Store Order

Supplemental technical material

Jeffrey A. Vaughan Todd Millstein

UCLA Compter Science Department

Technical Report #120007
May 9, 2012

Abstract

Modern multicore hardware and multithreaded programming languages expose *weak memory models* to programmers, which relax the intuitive sequential consistency (SC) memory model in order to support a variety of hardware and compiler optimizations. However, to our knowledge all prior work on secure information flow in a concurrent setting has assumed SC semantics. This paper investigates the impact of the Total Store Order (TSO) memory model, which is used by Intel x86 and Sun SPARC processors, on secure information flow, focusing on the natural security condition known as possibilistic noninterference. We show that possibilistic noninterference under SC and TSO are incomparable notions; neither property implies the other one. We define a simple type system for possibilistic noninterference under SC and demonstrate that it is not sound under TSO. We then provide two variants of this type system that are sound under TSO: one that requires only a small change to the original type system but is overly restrictive, and another that incorporates a form of flow sensitivity to safely retain desired expressiveness. Finally, we show that the original type system is in fact sound under TSO for programs that are free of data races. This report is a companion to "Secure Information Flow for Concurrent Programs Under Total Store Order" (Vaughan and Millstein, 2012).

1 Overview

This document presents the proofs and full definitions for "Secure Information Flow for Concurrent Programs Under Total Store Order" (Vaughan and Millstein, 2012).

The conference paper's type system only accepts *well-structured source* programs. For the proofs, it is convenient to present an expanded type system that accepts intermediate program states and to track the sourceness and well-structuredness properties separately. Additionally, results are numbered differently in the two documents. The chart below shows how these correspond.

Conference paper	This document
Theorem 3	Corollary 10
Theorem 4	Corollary 3
Theorem 6	Corollary 7
Theorem 7	Lemma 111
Theorem 8	Lemma 112
Theorem 15	Theorem 2

2 Language definition

2.1 Language syntax and main semantic sets

This syntax of the concurrent imperative language, and definitions of the main semantic objects are as follows.

Local variables	x, y, z	$\in$	$\mathbf{LocalVar}$	
Shared variables	X, Y, Z	$\in$	$\mathbf{HeapVar}$	
Variables, **Var**	v	$::=$	$\mathbf{LocalVar} \cup \mathbf{HeapVar}$	
Locks	ℓ	$\in$	$\mathbf{Lock}$	A finite set
Integer literals	i	$\in$	$\mathbb{Z}$	
Boolean literals	β	$\in$	$\mathbb{B} = \{\mathbf{true}, \mathbf{false}\}$	
Arithmetic Exp.	a	$::=$		
		$\mid$	x	Local var
		$\mid$	i	Integer literal
		$\mid$	$a \oplus a \mid (b \mathbin{\mathbf{?}} a \mathbin{\mathbf{:}} a) \mid \ldots$	Arithmetic ops.
Boolean Exp.	b	$::=$		
		$\mid$	β	Boolean literal
		$\mid$	$\mathbf{isZero}\ a$	Zero test
		$\mid$	$b \ovee b$	Boolean ops.
Command	c, d	$::=$		
		$\mid$	$x := X$	Load
		$\mid$	$X := x$	Store
		$\mid$	$x := a$	Expression evaluation
		$\mid$	$\mathbf{sync}\ \ell\ \mathbf{do}\ c$	Lock acquire
		$\mid$	$\mathbf{holding}\ \ell\ \mathbf{do}\ c$	Lock held (forbidden in source programs)
		$\mid$	$\mathbf{fence}$	Memory barrier
		$\mid$	$\mathbf{fork}\ c$	Thread creation
		$\mid$	$c_1; c_2 \mid \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2$	While language commands
		$\mid$	$\mathbf{while}\ b\ \mathbf{do}\ c \mid \mathbf{skip}$	
Write buffer, **WriteBuf**	W	$::=$		
		$\mid$	nil	Empty
		$\mid$	$(X := i)::W$	Write pending (ready to commit at head, newer writes in tail)
Local state	L	$\in$	$(\mathbf{LocalVar} \to \mathbb{Z}) \times \mathcal{P}(\mathbf{Lock})$ $\times\ \mathbf{WriteBuff}$	
Thread IDs	ι	$\in$	$\mathbf{TID}$	A finite set
Thread	s, t	$::=$	$\langle L, c \rangle_\iota$	
Process soup	P, Q	$::=$		
		$\mid$	$\mathfrak{o}$	Nil process
		$\mid$	$t \parallel P$	Parallel composition
Global state	G, H	$\in$	$(\mathbf{HeapVar} \to \mathbb{Z}) \times \mathcal{P}(\mathbf{Lock})$	
Configuration, **Config**	χ	$::=$	(G, P)	
Operation	op	$::=$	$eval \mid commit$	
Action, **Action**	α	$::=$	$op(i)$	
Action Set	A	$\in$	$\mathcal{P}(\mathbf{Action})$	

2.2 Notation

Suppose local state $L = (M, \lambda, W)$. We write $L.mem$, $L.locks$, and $L.wb$ for M, λ, and W respectively. If $x \in \mathbf{LocalVar}$ we write $L[x \mapsto i]$ for $(M[x \mapsto i], \lambda, W)$ and $L(x)$ for $M(x)$. Likewise we use $L \cup \kappa$ and $L \cap \kappa$ and $L \setminus \kappa$ and $\ell \in L$ for $(M, \lambda \cup \kappa)$ and $(M, \lambda \cap \kappa)$ and $(M, \lambda \setminus \kappa)$ and $\ell \in \lambda$. Finally we use $L +\!\!+ (X := i)$ for $(M, \lambda, W +\!\!+ (X := i)::nil)$ and $(X := i)::L$ for $(M, \lambda, (X := i)::W)$.

Symbol $L_\oslash$ represents the "empty" local state $((\lambda x.0), \emptyset, nil)$.

Suppose $G = (S, \lambda)$. We write $G.mem$ for S and $G.locks$ for λ.

Suppose $t = \langle L, c\rangle$. Then $t.cmd$, $t.ls$, $t.locks$, $t.wb$, and $t.mem$ mean c, L, $L.locks$, $L.wb$, and $L.mem$ respectively.

Metavariable S, for *Store*, ranges over global heaps and metavariable M, for *Memory*, ranges over local memories.

Thread ids are only important for the argument in Section 4 and are often suppressed to avoid clutter.

We will sometimes write singleton process $t \parallel \mathsf{o}$ simply as t. Additionally we define $Q +\!\!+ P$ to be the process soup formed by appending Q and P; that is $\mathsf{o} +\!\!+ P = P$ and $(t \parallel Q) +\!\!+ P = t \parallel (Q +\!\!+ P)$. We abuse notation and write $t \in P$ when $P \equiv t \parallel Q$ for some Q as well as $P \parallel Q$ for $P +\!\!+ Q$. Finally we write *nonempty* P when has form $t \parallel P_0$ and for such a *nonempty* process soup, $hd(P)$ is t.

2.3 Single Step Operations

$\boxed{(S; W)[X] \Downarrow i}$

$$\frac{}{(S; W +\!\!+ (X := i))[X] \Downarrow i} \qquad \frac{X \neq Y \qquad (S; W)[X] \Downarrow i}{(S; W +\!\!+ (Y := i_0))[X] \Downarrow i} \qquad \frac{}{(S; nil)[X] \Downarrow S(X)}$$

$\boxed{L[a] \Downarrow i}$

$$\frac{}{L[x] \Downarrow L(x)} \qquad \frac{}{L[i] \Downarrow i} \qquad \frac{L[a_1] \Downarrow i_1 \qquad L[a_2] \Downarrow i_2 \qquad i = i_1 \llbracket \oplus \rrbracket i_2}{L[a_1 \oplus a_2] \Downarrow i} \qquad \frac{L[b] \Downarrow \mathbf{true} \qquad L[a_1] \Downarrow i}{L[(b \mathbin{\mathbf{?}} a_1 \mathbin{\mathbf{:}} a_2)] \Downarrow i}$$

$$\frac{L[b] \Downarrow \mathbf{false} \qquad L[a_2] \Downarrow i}{L[(b \mathbin{\mathbf{?}} a_1 \mathbin{\mathbf{:}} a_2)] \Downarrow i}$$

$\boxed{L[b] \Downarrow \beta}$

$$\frac{}{L[\beta] \Downarrow \beta} \qquad \frac{L[a] \Downarrow 0}{L[\mathbf{isZero}\ a] \Downarrow \mathbf{true}} \qquad \frac{L[a] \Downarrow i \qquad i \neq 0}{L[\mathbf{isZero}\ a] \Downarrow \mathbf{false}} \qquad \frac{L[b_1] \Downarrow \beta_1 \qquad L[b_2] \Downarrow \beta_2 \qquad \beta = \beta_1 \llbracket \otimes \rrbracket \beta_2}{L[b_1 \otimes b_2] \Downarrow \beta}$$

$\boxed{(G, t) \longrightarrow^{commit} (G', P)}$

$$\frac{}{(G, \langle (X := i)::L, c\rangle) \longrightarrow^{commit} (G[X \mapsto i], \langle L, c\rangle)}$$

$$\boxed{(G, t) \longrightarrow^{eval} (G', P)}$$

$$\frac{}{(G, \langle L, X := x\rangle) \longrightarrow^{eval} (G, \langle L +\!\!+ (X := L(x)), \mathbf{skip}\rangle)}\ \textsc{ec-Store}$$

$$\frac{(G.mem; L.wb)[X] \Downarrow i}{(G, \langle L, x := X\rangle) \longrightarrow^{eval} (G, \langle L[x \mapsto i], \mathbf{skip}\rangle)}\ \textsc{ec-Load}$$

$$\frac{L[a] \Downarrow i}{(G, \langle L, x := a\rangle) \longrightarrow^{eval} (G, \langle L[x \mapsto i], \mathbf{skip}\rangle)}\ \textsc{ec-EvalExp}$$

$$\frac{\ell \in G \qquad L.wb = nil}{(G, \langle L, \mathbf{sync}\ \ell\ \mathbf{do}\ c\rangle) \longrightarrow^{eval} (G \setminus \{\ell\}, \langle L \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c\rangle)}\ \textsc{ec-SyncAcquire}$$

$$\frac{\ell \in L}{(G, \langle L, \mathbf{sync}\ \ell\ \mathbf{do}\ c\rangle) \longrightarrow^{eval} (G, \langle L, \mathbf{fence};\ (c;\ \mathbf{fence})\rangle)}\ \textsc{ec-SyncReenter}$$

$$\frac{\ell \in L \qquad (G, \langle L, c\rangle_\iota) \longrightarrow^{eval} (G', \langle L', c'\rangle_\iota \parallel P)}{(G, \langle L, \mathbf{holding}\ \ell\ \mathbf{do}\ c\rangle_\iota) \longrightarrow^{eval} (G', \langle L', \mathbf{holding}\ \ell\ \mathbf{do}\ c'\rangle_\iota \parallel P)}\ \textsc{ec-HoldStep}$$

$$\frac{\ell \in L \qquad L.wb = nil}{(G, \langle L, \mathbf{holding}\ \ell\ \mathbf{do}\ \mathbf{skip}\rangle) \longrightarrow^{eval} (G \cup \{\ell\}, \langle L \setminus \{\ell\}, \mathbf{skip}\rangle)}\ \textsc{ec-HoldRelease}$$

$$\frac{L.wb = nil}{(G, \langle L, \mathbf{fence}\rangle) \longrightarrow^{eval} (G, \langle L, \mathbf{skip}\rangle)}\ \textsc{ec-Fence}$$

$$\frac{L.wb = nil \qquad \iota'\ fresh}{(G, \langle L, \mathbf{fork}\ c\rangle_\iota) \longrightarrow^{eval} (G, \langle L, \mathbf{skip}\rangle_\iota \parallel \langle L_\oslash, c\rangle_{\iota'})}\ \textsc{ec-Fork}$$

$$\frac{(G, \langle L, c_1\rangle_\iota) \longrightarrow^{eval} (G', \langle L', c_1'\rangle_\iota \parallel P)}{(G, \langle L, c_1;\ c_2\rangle_\iota) \longrightarrow^{eval} (G', \langle L', c_1';\ c_2\rangle_\iota \parallel P)}\ \textsc{ec-SeqStruct}$$

$$\frac{}{(G, \langle L, \mathbf{skip};\ c\rangle_\iota) \longrightarrow^{eval} (G, \langle L, c\rangle_\iota)}\ \textsc{ec-SeqSkip}$$

$$\frac{L[b] \Downarrow \mathbf{true}}{(G, \langle L, \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2\rangle) \longrightarrow^{eval} (G, \langle L, c_1\rangle)}\ \textsc{ec-IfTrue}$$

$$\frac{L[b] \Downarrow \mathbf{false}}{(G, \langle L, \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2\rangle) \longrightarrow^{eval} (G, \langle L, c_2\rangle)}\ \textsc{ec-IfFalse}$$

$$\frac{L[b] \Downarrow \mathbf{true}}{(G, \langle L, \mathbf{while}\ b\ \mathbf{do}\ c\rangle_\iota) \longrightarrow^{eval} (G, \langle L, c;\ \mathbf{while}\ b\ \mathbf{do}\ c\rangle_\iota)}\ \textsc{ec-WhileTrue}$$

$$\frac{L[b] \Downarrow \mathbf{false}}{(G, \langle L, \mathbf{while}\ b\ \mathbf{do}\ c\rangle_\iota) \longrightarrow^{eval} (G, \langle L, \mathbf{skip}\rangle_\iota)}\ \textsc{ec-WhileFalse} \qquad \frac{L.wb = nil \qquad L.locks = \emptyset}{(G, \langle L, \mathbf{skip}\rangle) \longrightarrow^{eval} (G, \mathfrak{o})}\ \textsc{ec-Reap}$$

2.4 Possibilistic evaluation

$$op(i) \in \mathrm{Ready}(G, t_1 \cdots t_i \cdots t_n) \quad \text{iff} \quad \text{exists } \chi \text{ such that } (G, t_i) \longrightarrow^{op} \chi$$

$$\mathrm{ReadySC}(\chi) = \begin{cases} \mathit{commits} & \mathit{commits} \neq \emptyset \\ \mathrm{Ready}(\chi) & \text{otherwise} \end{cases}$$

$$\text{where } \mathit{commits} = \{\mathit{commit}(i) \mid \mathit{commit}(i) \in \mathrm{Ready}(\chi)\}$$

$\boxed{(G, P) \Longrightarrow^{\mathsf{sc}} (G', P')}$

$$\frac{P = t_1 \ldots t_{i-1} \parallel t_i \parallel t_{i+1} \ldots t_n \qquad \begin{array}{c} op(i) \in \mathrm{ReadySC}(\chi) \\ (G, t_i) \longrightarrow^{op} (G', Q) \end{array} \qquad P' = t_1 \ldots t_{i-1} \parallel Q \parallel t_{i+1} \ldots t_n}{(G, P) \Longrightarrow^{\mathsf{sc}} (G', P')}$$

$\boxed{(G, P) \Longrightarrow^{\mathsf{tso}} (G', P')}$

$$\frac{P = t_1 \ldots t_{i-1} \parallel t_i \parallel t_{i+1} \ldots t_n \qquad (G, t_i) \longrightarrow^{op} (G', Q) \qquad P' = t_1 \ldots t_{i-1} \parallel Q \parallel t_{i+1} \ldots t_n}{(G, P) \Longrightarrow^{\mathsf{tso}} (G', P')}$$

Define $\Longrightarrow^{\mathsf{mm}*}$ as the reflexive, transitive closure of the $\Longrightarrow^{\mathsf{mm}}$ relation where mm is either sc or tso.

3 A simple type system for possibilistic flows

This is a minimal delta from (Smith and Volpano, 1998). Typing uses the following syntactic classes.

Syntactic Security Level	τ	::=	*low* | *high*
Static Security Context	Γ	::=	$\mathbf{HeapVar} \cup \mathbf{LocalVar} \cup \mathbf{Lock} \rightarrow \tau$

We define lattice operators for syntactic security levels: least upper bound $\sqcup$, greatest lower bound $\sqcap$, and partial order $\sqsubseteq$. These respect the ordering $low \sqsubseteq high$.

3.1 Types and basic properties

$\boxed{\Gamma \vdash a : \tau}$

$$\frac{\Gamma(x) \sqsubseteq \tau}{\Gamma \vdash x : \tau} \qquad \frac{}{\Gamma \vdash i : \tau} \qquad \frac{\Gamma \vdash a_1 : \tau \qquad \Gamma \vdash a_2 : \tau}{\Gamma \vdash a_1 \oplus a_2 : \tau} \qquad \frac{\Gamma \vdash b : \tau \qquad \Gamma \vdash a_1 : \tau \qquad \Gamma \vdash a_2 : \tau}{\Gamma \vdash (b \mathbin{\mathbf{?}} a_1 \mathbin{\mathbf{:}} a_2) : \tau}$$

$\boxed{\Gamma \vdash b : \tau}$

$$\frac{}{\Gamma \vdash \beta : \tau} \qquad \frac{\Gamma \vdash a : \tau}{\Gamma \vdash \mathbf{isZero}\ a : \tau} \qquad \frac{\Gamma \vdash b_1 : \tau \qquad \Gamma \vdash b_2 : \tau}{\Gamma \vdash b_1 \otimes b_2 : \tau}$$

$\boxed{pc; \Gamma \vdash^{\text{tso}} c}$

$$\frac{pc \sqcup \Gamma(Y) \sqsubseteq \Gamma(x)}{pc; \Gamma \vdash^{\text{tso}} x := Y}\ \textsc{tso-Load} \qquad \frac{pc \sqcup \Gamma(y) \sqsubseteq \Gamma(X)}{pc; \Gamma \vdash^{\text{tso}} X := y}\ \textsc{tso-Store} \qquad \frac{\Gamma \vdash a : \tau \qquad pc \sqcup \tau \sqsubseteq \Gamma(x)}{pc; \Gamma \vdash^{\text{tso}} x := a}\ \textsc{tso-Eval}$$

$$\frac{\Gamma(\ell); \Gamma \vdash^{\text{tso}} c}{low; \Gamma \vdash^{\text{tso}} \textbf{sync}\ \ell\ \textbf{do}\ c}\ \textsc{tso-Sync} \qquad \frac{\Gamma(\ell); \Gamma \vdash^{\text{tso}} c}{low; \Gamma \vdash^{\text{tso}} \textbf{holding}\ \ell\ \textbf{do}\ c}\ \textsc{tso-Hold}$$

$$\frac{}{low; \Gamma \vdash^{\text{tso}} \textbf{fence}}\ \textsc{tso-Fence} \qquad \frac{pc; \Gamma \vdash^{\text{tso}} c}{low; \Gamma \vdash^{\text{tso}} \textbf{fork}\ c}\ \textsc{tso-Fork} \qquad \frac{pc; \Gamma \vdash^{\text{tso}} c_1 \qquad pc; \Gamma \vdash^{\text{tso}} c_2}{pc; \Gamma \vdash^{\text{tso}} c_1;\ c_2}\ \textsc{tso-Seq}$$

$$\frac{\Gamma \vdash b : \tau \qquad pc \sqcup \tau; \Gamma \vdash^{\text{tso}} c_1 \qquad pc \sqcup \tau; \Gamma \vdash^{\text{tso}} c_2}{pc; \Gamma \vdash^{\text{tso}} \textbf{if}\ b\ \textbf{do}\ c_1\ \textbf{else}\ c_2}\ \textsc{tso-If} \qquad \frac{\Gamma \vdash b : low \qquad pc; \Gamma \vdash^{\text{tso}} c}{low; \Gamma \vdash^{\text{tso}} \textbf{while}\ b\ \textbf{do}\ c}\ \textsc{tso-While}$$

$$\frac{}{pc; \Gamma \vdash^{\text{tso}} \textbf{skip}}\ \textsc{tso-Skip}$$

$\boxed{pc; \Gamma \vdash^{\text{tso}} \lambda}$

$$\frac{}{pc; \Gamma \vdash^{\text{tso}} \{\}} \qquad \frac{}{low; \Gamma \vdash^{\text{tso}} \lambda}$$

$\boxed{pc; \Gamma \vdash^{\text{tso}} W}$

$$\frac{}{pc; \Gamma \vdash^{\text{tso}} nil} \qquad \frac{pc \sqsubseteq \Gamma(X) \qquad pc; \Gamma \vdash^{\text{tso}} W}{pc; \Gamma \vdash^{\text{tso}} (X := i)::W}$$

$\boxed{pc; \Gamma \vdash^{\text{tso}} t}$

$$\frac{pc; \Gamma \vdash^{\text{tso}} \lambda \qquad pc; \Gamma \vdash^{\text{tso}} W \qquad pc; \Gamma \vdash^{\text{tso}} c}{pc; \Gamma \vdash^{\text{tso}} \langle (M, \lambda, W), c \rangle_\iota}$$

$\boxed{\overline{pc}; \Gamma \vdash^{\text{tso}} P}$

$$\frac{}{\cdot; \Gamma \vdash^{\text{tso}} \mathfrak{o}} \qquad \frac{pc; \Gamma \vdash^{\text{tso}} t \qquad \overline{pc}; \Gamma \vdash^{\text{tso}} P}{pc, \overline{pc}; \Gamma \vdash^{\text{tso}} t \parallel P}$$

$\boxed{pc \sqsubseteq \overline{pc}}$

$$\frac{}{pc \sqsubseteq \cdot} \qquad \frac{pc \sqsubseteq pc_0 \qquad pc \sqsubseteq \overline{pc}}{pc \sqsubseteq pc_0, \overline{pc}}$$

3.2 Properties of syntax and evaluation

Definition 1 (*size*).

$$\begin{aligned}
\mathit{size}\ \mathbf{skip} &= 1\\
\mathit{size}\ (x := a) &= 2\\
\mathit{size}\ (X := x) &= 3\\
\mathit{size}\ (x := X) &= 2\\
\mathit{size}\ \mathbf{fence} &= 2\\
\mathit{size}\ (c_1;\ c_2) &= 1 + \mathit{size}\ c_1 + \mathit{size}\ c_2\\
\mathit{size}\ (\mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2) &= 1 + \mathit{size}\ c_1 + \mathit{size}\ c_2\\
\mathit{size}\ (\mathbf{while}\ b\ \mathbf{do}\ c) &= 1 + \mathit{size}\ c\\
\mathit{size}\ (\mathbf{holding}\ \ell\ \mathbf{do}\ c) &= 1 + \mathit{size}\ c\\
\mathit{size}\ (\mathbf{sync}\ \ell\ \mathbf{do}\ c) &= 7 + \mathit{size}\ c\\
\mathit{size}\ (\mathbf{fork}\ c) &= 1 + \mathit{size}\ c
\end{aligned}$$

$$\begin{aligned}
\mathit{size}\ \mathit{nil} &= 0\\
\mathit{size}\ (X := i)::W &= 1 + \mathit{size}\ W
\end{aligned}$$

3.3 Evaluation contexts

$$\begin{array}{llll}
\text{Command Context} & \mathcal{C} & ::= & [\cdot] \mid \mathcal{C};\ c \mid \mathbf{holding}\ \ell\ \mathbf{do}\ \mathcal{C}\\
\text{Evaluation Context} & \mathcal{E} & = & (\lambda, \mathcal{C})
\end{array}$$

Notation $\mathcal{C}[c]$ has the usual meaning and when $\mathcal{E} = (\lambda, \mathcal{C})$ write $\mathcal{E}[W \mid \langle L, c\rangle_\iota]$ for $\langle W +\!\!+ L \cup \lambda, \mathcal{C}[c]\rangle_\iota$. Notation $\mathcal{E}_\emptyset$ means $([\cdot], \emptyset)$.

We define $\mathcal{C}.\mathit{locks}$ as follows:

$$\begin{aligned}
[\cdot].\mathit{locks} &= \emptyset\\
(\mathcal{C};\ c).\mathit{locks} &= \mathcal{C}.\mathit{locks}\\
(\mathbf{holding}\ \ell\ \mathbf{do}\ \mathcal{C}).\mathit{locks} &= \{\ell\} \cup \mathcal{C}.\mathit{locks}.
\end{aligned}$$

Definition 2 (Active evaluation context)**.** *Evaluation context $(\lambda, \mathcal{C})$ is active, written active $(\lambda, \mathcal{C})$, when $\mathcal{C}.\mathit{locks} \subseteq \lambda$.*

Lemma 1. *If $(G, \mathcal{E}[W \mid t]) \longrightarrow^{eval} (G', P')$ then active $\mathcal{E}$.*

Proof. Assume for a contradiction that $\mathcal{E} = (\lambda, \mathcal{C})$ is not active. Then there is a stuck **holding** command in $\mathcal{C}$ which contracts the assumption that $\mathcal{E}[W \mid t]$ takes an *eval*-step. ☐

Definition 3 (*canEval*)**.** *If there exist G, G', and P' such that $(G, t) \longrightarrow^{eval} (G', P')$, then we say canEval t.*

Lemma 2. *Suppose canEval $(\mathcal{E}[t])$. Then there exists $\mathcal{E}_0$ such that active $\mathcal{E}_0$ and $\mathcal{E}_0[t] = \mathcal{E}[t]$.*

Proof. Let $\mathcal{E} = (\lambda, \mathcal{C})$ and define $\mathcal{E}_0 = (\lambda \cup \mathcal{C}.\mathit{locks}, \mathcal{C})$. From *canEval* $\mathcal{E}[t]$ we know that for some G, G', and P' it is the case that $(G, t) \longrightarrow^{eval} (G', P')$. To show $\mathcal{E}_0[t] = \mathcal{E}[t]$ if suffices to show that $\mathcal{C}.\mathit{locks} \subseteq t.\mathit{ls}.\mathit{locks}$, which follows from a simple induction on the $\longrightarrow^{eval}$ relation. ☐

Lemma 3. *If $(G, P) \Longrightarrow^{\mathsf{tso}*} (G', P')$, then $(G, P \parallel P_0) \Longrightarrow^{\mathsf{tso}*} (G', P' \parallel P_0)$.*

Definition 4 (*hasEmptyWBs*(P))**.** *We say P has empty write buffers, written hasEmptyWBs(P), if for all t such that $P = P_1 \parallel t \parallel P_2$, it is the case that $t.\mathit{wb} = \mathit{nil}$.*

Lemma 4. *If $(G, P) \Longrightarrow^{\mathsf{sc}*} (G', P')$, and hasEmptyWBs$(P_0)$ then $(G, P \parallel P_0) \Longrightarrow^{\mathsf{sc}*} (G', P' \parallel P_0)$.*

Definition 5. *Source programs, and well structured contexts and commands*

$\boxed{\mathit{src}\ c}$

$$\frac{}{\mathit{src}\ (x := X)} \qquad \frac{}{\mathit{src}\ (X := x)} \qquad \frac{}{\mathit{src}\ (x := X)} \qquad \frac{\mathit{src}\ c}{\mathit{src}\ (\mathbf{sync}\ \ell\ \mathbf{do}\ c)} \qquad \frac{}{\mathbf{fence}} \qquad \frac{\mathit{src}\ c}{\mathit{src}\ \mathbf{fork}\ c}$$

$$\frac{\mathit{src}\ c_1 \qquad \mathit{src}\ c_2}{\mathit{src}\ c_1;\ c_2} \qquad \frac{\mathit{src}\ c_1 \qquad \mathit{src}\ c_2}{\mathit{src}\ \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2} \qquad \frac{\mathit{src}\ c}{\mathit{src}\ \mathbf{while}\ b\ \mathbf{do}\ c} \qquad \frac{}{\mathit{src}\ \mathbf{skip}}$$

$\boxed{\textit{wellStruct } \mathcal{C}}$

$$\frac{}{\textit{wellStruct } [\cdot]} \qquad \frac{\textit{wellStruct } \mathcal{C} \qquad \textit{src } c}{\textit{wellStruct } \mathcal{C};\, c} \qquad \frac{\textit{wellStruct } \mathcal{C} \qquad \ell \notin \mathcal{C}.\textit{locks}}{\textit{wellStruct } \textbf{holding } \ell \textbf{ do } \mathcal{C}}$$

$\boxed{\textit{wellStruct } c}$

$$\frac{\textit{wellStruct } \mathcal{C} \qquad \textit{src } c}{\textit{wellStruct } \mathcal{C}[c]}$$

$\boxed{\textit{wellStruct } t}$

$$\frac{\textit{wellStruct } c}{\textit{wellStruct } \langle L, c\rangle}$$

$\boxed{\textit{wellStruct } P}$

$$\frac{}{\textit{wellStruct } \circ} \qquad \frac{\textit{wellStruct } t \qquad \textit{wellStruct } P}{\textit{wellStruct } t \parallel P}$$

There is one interesting difference between the abridged definitions give in the conference submission and the full versions in this this document. The type systems presented in the paper only apply to source commands. In this document we extend typing to include threads, locksets, write buffers, and non-source commands that occur during evaluation. Thus the premises of some theorems in this document contain additional hypotheses stating that c is either a source command or is well structured, but these hypotheses are not needed in the submission as they are implied by typing.

Lemma 5. *Suppose $(G, P) \Longrightarrow^{\mathsf{tso}*} (G', P')$. If wellStruct P then wellStruct P'.*

Proof. by nested induction on the length of the $\Longrightarrow^{\mathsf{tso}*}$ derivation then induction on each $\longrightarrow^{eval}$ derivation or trivial consideration of $\longrightarrow^{commit}$ s. □

Lemma 6. *Suppose wellStruct c and $c = \mathcal{C}[c_0]$, then wellStruct c_0 and wellStruct $\mathcal{C}$.*

Proof. by structural induction on c. □

Definition 6 ($c.\textit{locks}$)**.**

$$\begin{aligned}
(\textbf{holding } \ell \textbf{ do } c).\textit{locks} &= \{\ell\} \cup c.\textit{locks} \\
(\textbf{sync } \ell \textbf{ do } c).\textit{locks} &= \{\ell\} \cup c.\textit{locks} \\
(c_1;\, c_2).\textit{locks} &= c_1.\textit{locks} \cup c_2.\textit{locks} \\
(\textbf{skip}).\textit{locks} &= \emptyset \\
&\vdots
\end{aligned}$$

Lemma 7. *Suppose $\mathcal{E} = (\lambda, \textbf{holding } \ell \textbf{ do } [\cdot])$ and wellStruct $\mathcal{E}[t]$. If $(G, \mathcal{E}[t]) \longrightarrow^{eval} (G', \mathcal{E}[t'] \parallel P')$ then $\ell \in t'$.*

Proof. Use induction on the *wellStruct* derivation to show that t does not contain redexes of the form **holding** ℓ **do** _, then continue by induction on the $\longrightarrow^{eval}$ derivation. □

Lemma 8. *Suppose wellStruct c and $c.\textit{locks} = \emptyset$. Then for any ℓ it is the case that wellStruct* **holding** ℓ **do** c.

Proof. by induction on the derivation of *wellStruct* c. $\square$

Lemma 9. *Suppose* $(G, t_1 \parallel \ldots t_n \parallel \mathfrak{o}) \implies^{\mathsf{mm}*} (G', t'_1 \parallel \ldots t'_m \parallel \mathfrak{o})$. *Then*

$$G.locks \cup t_1.ls.locks \cup \ldots \cup t_n.ls.locks = G'.locks \cup t'_1.ls.locks \cup \ldots \cup t'_m.ls.locks.$$

Proof. by easy induction. $\square$

3.4 Typing properties

Definition 7 ($tailOf(W_0, W)$)**.** *Write buffer* W_0 *is the tail of* W*, written* $tailOf(W_0, W)$*, when* $W = (X := i)::W_0$ *for some* X *and* i*.*

Lemma 10 (Step preservation)**.** *Suppose* $pc; \Gamma \vdash^{\text{tso}} t$ *and* $(G, \mathcal{E}[W \mid t]) \longrightarrow^{op} (G', \mathcal{E}[W' \mid t'] \parallel P')$*. Further suppose either* $W = W'$*, or both* $op = commit$ *and* $tailOf(W', W)$*. Then* $pc; \Gamma \vdash^{\text{tso}} t'$ *and* $\overline{pc}; \Gamma \vdash^{\text{tso}} P'$ *with* $pc \sqsubseteq \overline{pc}$*.*

Proof. By induction on the typing relation. $\square$

Lemma 11 (Subtyping)**.** *Suppose* $\tau_2 \sqsubseteq \tau_1$*. The following implications hold:*

- $\Gamma \vdash a : \tau_2$ *implies* $\Gamma \vdash a : \tau_1$
- $\Gamma \vdash b : \tau_2$ *implies* $\Gamma \vdash b : \tau_1$
- $\tau_1; \Gamma \vdash^{\text{tso}} c$ *implies* $\tau_2; \Gamma \vdash^{\text{tso}} c$
- $\tau_1; \Gamma \vdash^{\text{tso}} \lambda$ *implies* $\tau_2; \Gamma \vdash^{\text{tso}} \lambda$
- $\tau_1; \Gamma \vdash^{\text{tso}} W$ *implies* $\tau_2; \Gamma \vdash^{\text{tso}} W$
- $\tau_1; \Gamma \vdash^{\text{tso}} t$ *implies* $\tau_2; \Gamma \vdash^{\text{tso}} t$

Proof. By induction. $\square$

Lemma 12 (Eval preservation)**.** *Suppose* $\overline{pc}; \Gamma \vdash^{\text{tso}} P$ *where* $pc \sqsubseteq \overline{pc}$*. If* $(G, P) \implies^{m*} (G', P')$ *then* $\overline{pc}'; \Gamma \vdash^{\text{tso}} P'$ *where* $pc \sqsubseteq \overline{pc}'$*.*

Proof. by induction on the number of evaluation steps. The lemma holds trivially when there are zero steps. Instead suppose the trace contains $n+1$ steps. We have

$$(G, P) = (G, P_1 \parallel t \parallel P_2) \implies^{m} (H, P_1 \parallel Q \parallel P_2) \implies^{m*} (G', P')$$

where $(G, t) \longrightarrow^{op} (H, Q)$. Using the induction hypothesis and the definition of $\vdash^{\text{tso}}$, it suffices to show that $\overline{pc}_Q; \Gamma \vdash^{\text{tso}} Q$ for some $\overline{pc}_Q$ where $pc \sqsubseteq \overline{pc}_Q$. If $Q = \mathfrak{o}$ this is immediate. If instead $Q = t' \parallel Q_0$ then for $\mathcal{E} = (\emptyset, [\cdot])$ we have $(G, \mathcal{E}[nil \mid t]) \longrightarrow^{op} (H, \mathcal{E}[nil \mid t'] \parallel Q_0)$. Inverting the definition of $\vdash^{\text{tso}}$ yields $pc_t; \Gamma \vdash^{\text{tso}} t$ where $pc \sqsubseteq pc_t$. By Lemma 10, both $pc_t; \Gamma \vdash^{\text{tso}} t'$ and $\overline{pc}_{Q_0}; \Gamma \vdash^{\text{tso}} Q_0$ where $pc_t \sqsubseteq \overline{pc}_{Q_0}$. Conclude using the definition of $\vdash^{\text{tso}}$ and the transitivity of $\sqsubseteq$.

$\square$

Lemma 13 (Total write-buffer typing)**.** *For all* W*,* $low; \Gamma \vdash^{\text{tso}} W$*.*

Proof. By inspection. $\square$

3.5 Trace properties

Definition 8 (Front-Reap–Freedom, Commit-Freedom and Simple Traces)**.** *Let $\mathcal{T}$ range over non-empty sequences of of configurations.*

Write $\mathcal{T} :: (G_1, P_1) \Longrightarrow^{\mathsf{tso}*} (G_n, P_n)$ when $\mathcal{T}$ has form $(G_1, P_1), (G_2, P_2) \ldots (G_n, P_n)$ and for each $i \in \{1 \ldots n-1\}$ it is the case that $(G_i, P_i) \Longrightarrow^{\mathsf{tso}} (G_{i+1}, P_{i+1})$.

We say FrontReapFree $\mathcal{T}$ when for each such i, there exist thread pools P, Q, R and thread t such that

$$\begin{aligned} P_i &= P \parallel t \parallel Q \\ P_{i+1} &= P \parallel R \parallel Q \end{aligned}$$

where either P is non-empty or $(G_i, t) \longrightarrow^{op} (G_{i+1}, R)$ by a rule other than EC-REAP.

We say FrontCommitFree $\mathcal{T}$ when for each i,

$$\begin{aligned} P_i &= P \parallel t \parallel Q \\ P_{i+1} &= P \parallel R \parallel Q \end{aligned}$$

either P is non-empty or $(G_i, t) \longrightarrow^{op} (G_{i+1}, R)$ by a rule other than EC-COMMIT.

Finally Simple $\mathcal{T}$ when both FrontReapFree $\mathcal{T}$ and FrontCommitFree $\mathcal{T}$.

Lemma 14. *Suppose $high; \Gamma \vdash^{\mathrm{tso}} t$ and active $\mathcal{E}$ then $\mathcal{T} :: (G, \mathcal{E}[W \mid t] \parallel \circ) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[W \mid \langle L', \mathbf{skip} \rangle_\iota] \parallel \circ)$ for some G', L' and Simple $\mathcal{T}$.*

Proof. Let $\langle L, c \rangle = t$ and proceed by an easy strong induction on *size* c. Because c is *high*-typed it does not contain any occurrences of **while**, so if it takes a step other than EC-REAP, EC-COMMIT, or EC-FORK the size of c decreases and we can conclude by the induction hypothesis.

If $c = \mathbf{skip}$ we're done. Otherwise, observe that typing ensures c contains no occurrences of **fork**, **fence**, **sync**, or **holding**. Therefore c is not stuck and can take an *eval*-step that is not EC-REAP or EC-FORK. Conclude noting that c steps to a smaller command.

□

Lemma 15 (Contextual compatibility for *eval* steps)**.** *Suppose active $\mathcal{E}$ and $c.locks \cap \lambda \subseteq L.locks$ where $\mathcal{E} = (\lambda, \mathcal{C})$. Also assume that for all $\ell \in \lambda$ it is the case that wellStruct* **holding** *ℓ* **do** *c. If $(G, \mathcal{E}_\emptyset[W \mid \langle L, c \rangle]) \longrightarrow^{eval} (G', \mathcal{E}_\emptyset[W \mid t'] \parallel P')$ then $(G, \mathcal{E}[W \mid \langle L, c \rangle]) \longrightarrow^{eval} (G', \mathcal{E}[W \mid t'] \parallel P')$ by a step rule other than* EC-REAP.

Proof. by induction. □

Lemma 16 (Contextual compatibility for arbitrary steps)**.** *Suppose active $\mathcal{E}$ and $c.locks \cap \lambda \subseteq L.locks$ where $\mathcal{E} = (\lambda, \mathcal{C})$. Also assume that for all $\ell \in \lambda$ it is the case that wellStruct* **holding** *ℓ* **do** *c. If $(G, \langle L, c \rangle_\iota) \longrightarrow^{op} (G', \langle L', c' \rangle_\iota \parallel P')$, then it is the case that $(G, \mathcal{E}[nil \mid \langle L, c \rangle_\iota]) \longrightarrow^{op} (G', \mathcal{E}[nil \mid \langle L', c' \rangle_\iota] \parallel P')$ by a step rule other than* EC-REAP.

Proof. *Commit*-steps are trivial; use Lemma 15 for *eval*-steps. □

Lemma 17 (Contextual compatibility for evaluation)**.** *Suppose $\mathcal{T} :: (G, \langle L, c \rangle_\iota \parallel P) \Longrightarrow^{\mathsf{tso}*} (G', \langle L', c' \rangle_\iota \parallel P')$ and FrontReapFree $\mathcal{T}$. Assume for $\mathcal{E} = (\lambda, \mathcal{C})$ and both active $\mathcal{E}$ and $c.locks \cap \lambda \subseteq L.locks$. Also assume that for all $\ell \in \lambda$ it is the case that wellStruct* **holding** *ℓ* **do** *c. Then it is the case that $\mathcal{T}' :: (G, \mathcal{E}[nil \mid \langle L, c \rangle_\iota] \parallel P) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[nil \mid \langle L', c' \rangle_\iota] \parallel P')$ where FrontReapFree $\mathcal{T}'$.*

Proof. By an easy induction on the size of $\mathcal{T}$, using Lemma 16. □

3.6 Equivalences

We define several forms of low equivalence. The various $\sim$ relations are used with all three systems—tso, sc, and wb—introduced in this document, while the $\sim^{\mathrm{tso}}$ relations are specialized for the Smith-Volpano–style system.

Definition 9 ($\sim_\Gamma$)**.**

1. *$M_1 \sim_\Gamma M_2$ iff for all x such that $\Gamma(x) = low$ it is the case that $M_1(x) = M_2(x)$.*

2. *$W_1 \sim_\Gamma W_2$ is defined as the least fixed point of the following implications.*

(a) $nil \sim_\Gamma nil$

(b) $(X := i)::W_1 \sim_\Gamma (X := i)::W_2$ *when* $W_1 \sim_\Gamma W_2$

(c) $(X := i)::W_1 \sim_\Gamma W_2$ *when* $W_1 \sim_\Gamma W_2$ *and* $\Gamma(X) = high$

(d) $W_1 \sim_\Gamma (X := i)::W_2$ *when* $W_1 \sim_\Gamma W_2$ *and* $\Gamma(X) = high$

3. $S_1 \sim_\Gamma S_2$ *iff for all* X *such that* $\Gamma(X) = low$ *it is the case that* $S_1(X) = S_2(X)$.

Definition 10 ($\sim_\Gamma^{\mathrm{tso}}$).

1. $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$ *iff each of the following holds:*

   (a) $L_1.wb \sim_\Gamma L_2.wb$

   (b) $L_1.mem \sim_\Gamma L_2.mem$

   (c) $L_1.locks = L_2.locks$

2. $t_1 \sim_\Gamma^{\mathrm{tso}} t_2$ *is defined by the following introduction rules*

   (a) $\langle L_1, c\rangle_{\iota_1} \sim_\Gamma^{\mathrm{tso}} \langle L_2, c\rangle_{\iota_2}$ *when* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$

   (b) $\mathcal{E}[W_1 \,|\, \langle L_1, c_1\rangle_{\iota_1}] \sim_\Gamma^{\mathrm{tso}} \mathcal{E}[W_2 \,|\, \langle L_2, c_2\rangle_{\iota_2}]$ *when* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$ *and* $W_1 \sim_\Gamma W_2$ *and both* $high; \Gamma \vdash^{\mathrm{tso}} \langle L_1, c_1\rangle_{\iota_1}$ *and* $high; \Gamma \vdash^{\mathrm{tso}} \langle L_2, c_2\rangle_{\iota_2}$.

3. $G_1 \sim_\Gamma^{\mathrm{tso}} G_2$ *iff* $G_1.mem \sim_\Gamma G_2.mem$ *and* $G_1.locks = G_2.locks$.

4. $P_1 \sim_\Gamma^{\mathrm{tso}} P_2$ *is defined by the least fixed point of the following implications.*

   (a) $\mathfrak{o} \sim_\Gamma^{\mathrm{tso}} \mathfrak{o}$, *always*

   (b) $t \parallel P_1 \sim_\Gamma^{\mathrm{tso}} P_2$ *when* $high; \Gamma \vdash^{\mathrm{tso}} t$ *and* $P_1 \sim_\Gamma^{\mathrm{tso}} P_2$

   (c) $P_1 \sim_\Gamma^{\mathrm{tso}} t \parallel P_2$ *when* $high; \Gamma \vdash^{\mathrm{tso}} t$ *and* $P_1 \sim_\Gamma^{\mathrm{tso}} P_2$

   (d) $t_1 \parallel P_1 \sim_\Gamma^{\mathrm{tso}} t_2 \parallel P_2$ *when* $t_1 \sim_\Gamma^{\mathrm{tso}} t_2$ *and* $P_1 \sim_\Gamma^{\mathrm{tso}} P_2$

5. $(G_1, P_1) \sim_\Gamma^{\mathrm{tso}} (G_2, P_2)$ *when* $G_1 \sim_\Gamma^{\mathrm{tso}} G_2$ *and* $P_1 \sim_\Gamma^{\mathrm{tso}} P_2$.

Lemma 18. *Each* $\sim_\Gamma$ *and* $\sim^{\mathrm{tso}}$ *relation is an equivalence relation.*

Proof. By inspection. ☐

Lemma 19. *If* $P_{11} \parallel t_1 \parallel P_{12} \sim_\Gamma P_2$ *then* $P_2 = P_{21} \parallel P_2^* \parallel P_{22}$ *where the following hold:*

$$\begin{array}{rcl} P_{21} & \sim_\Gamma & P_{11} \\ P_2^* & \sim_\Gamma & t_1 \\ P_{22} & \sim_\Gamma & P_{12} \\ P_2^* & \in & \{\mathfrak{o}, t_2 \parallel \mathfrak{o}\} \text{ for some } t_2 \end{array}$$

Lemma 20. *Suppose* $G_1 \sim_\Gamma^{\mathrm{tso}} G_2$. *Then* $G_1 \cup \{\ell\} \sim_\Gamma^{\mathrm{tso}} G_2 \cup \{\ell\}$.

Lemma 21. *Suppose* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$. *Then* $L_1 \cup \lambda \sim_\Gamma^{\mathrm{tso}} L_2 \cup \lambda$.

Lemma 22. *Suppose* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$. *Then* $L_1 +\!\!+ (X := i) \sim_\Gamma^{\mathrm{tso}} L_2 +\!\!+ (X := i)$.

Lemma 23. *Suppose* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$ *and* $\Gamma(X) = high$. *Then* $L_1 +\!\!+ (X := i) \sim_\Gamma^{\mathrm{tso}} L_2 +\!\!+ (X := j)$.

Lemma 24. *Suppose* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$. *Then* $L_1[x \mapsto i] \sim_\Gamma^{\mathrm{tso}} L_2[x \mapsto i]$.

Lemma 25. *Suppose* $L_1 \sim_\Gamma^{\mathrm{tso}} L_2$ *and* $\Gamma(x) = high$. *Then* $L_1[x \mapsto i_1] \sim_\Gamma^{\mathrm{tso}} L_2[x \mapsto i_2]$.

Lemma 26. *Suppose* $G_1.mem \sim_\Gamma G_2.mem$ *and* $L_1.wb \sim_\Gamma L_2.wb$ *and* $\Gamma(X) = low$. *If* $(G_1.mem; L_1.wb)[X] \Downarrow i_1$ *and* $(G_2.mem; L_2.wb)[X] \Downarrow i_2$ *then* $i_1 = i_2$.

Lemma 27. $G_1 \sim_\Gamma^{\mathrm{tso}} G_2$ *implies* $G_1[X \mapsto i] \sim_\Gamma^{\mathrm{tso}} G_2[X \mapsto i]$

Lemma 28. *If* $t_1 \parallel \mathfrak{o} \sim_\Gamma^{\mathrm{tso}} t_2 \parallel \mathfrak{o}$ *and* $W_1 \sim_\Gamma W_2$ *then* $\mathcal{E}[W_1 \,|\, t_1] \parallel \mathfrak{o} \sim_\Gamma^{\mathrm{tso}} \mathcal{E}[W_2 \,|\, t_2] \parallel \mathfrak{o}$.

Lemma 29. $\langle L_1, c_1\rangle \sim_\Gamma^{\mathrm{tso}} \langle L_2, c_2\rangle$ *implies* $L_1.wb \sim_\Gamma^{\mathrm{tso}} L_2.wb$.

Proof. Suppose we have $L_1 \sim_\Gamma^{\text{tso}} L_2$, then we're done by definition. Otherwise L_1 and L_2 have write buffers with equivalent prefixes and suffixes. (The suffixes are both *high*-typed). These are also equivalent. □

Lemma 30. *Suppose $\langle L_1, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle L_2, c_2\rangle_{\iota_2}$ and $L_1^* \sim_\Gamma^{\text{tso}} L_1$. Then $\langle L_1^*, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle L_2, c_2\rangle_{\iota_2}$.*

Proof. We proceed by considering two cases. Suppose that $c_1 = c_2$ and $L_1 \sim_\Gamma^{\text{tso}} L_2$; then we conclude using Lemma 18.

Suppose instead that

$$\langle L_1, c_1\rangle_{\iota_1} = \mathcal{E}[W_1 \,|\, \langle L_{10}, c_{10}\rangle_{\iota_1}]$$
$$\langle L_2, c_2\rangle_{\iota_2} = \mathcal{E}[W_2 \,|\, \langle L_{20}, c_{20}\rangle_{\iota_2}]$$

with $high; \Gamma \vdash^{\text{tso}} \langle L_{10}, c_{10}\rangle_{\iota_1}$ and $high; \Gamma \vdash^{\text{tso}} \langle L_{20}, c_{20}\rangle_{\iota_2}$ and both $L_{10} \sim_\Gamma^{\text{tso}} L_{20}$ and $W_1 \sim_\Gamma W_2$. Let $(\lambda, \mathcal{C}) = \mathcal{E}$ and $W_1^* = L_1^*.wb$ and define $L_{10}^* = (L_1^*.mem, L_1^*.locks \setminus \lambda, nil)$. We want to find

$$\langle L_1^*, c_1\rangle_{\iota_1} = \mathcal{E}[W_1^* \,|\, \langle L_{10}^*, c_{10}\rangle_{\iota_1}] \sim_\Gamma^{\text{tso}} \mathcal{E}[W_2 \,|\, \langle L_{20}, c_{20}\rangle_{\iota_2}],$$

which follows from three interesting properties.

- To establish $high; \Gamma \vdash^{\text{tso}} \langle L_{10}^*, c_{10}\rangle_{\iota_1}$, it is necessary to show $L_{10}^*.locks = \emptyset$:

$$\begin{array}{rcll} L_{10}^*.locks & = & L_1^* \setminus \lambda & \\ & = & L_1 \setminus \lambda & \text{by defn. of } \sim_\Gamma^{\text{tso}} \\ & = & L_{10} \setminus \lambda & \text{by defn of } \mathcal{E}\text{-substitution} \\ & = & \emptyset \setminus \lambda & \text{by typing} \end{array}$$

- To show $W_1^* \sim_\Gamma W_2$ we use Lemma 29 to find $W_1^* = L_1^*.wb \sim_\Gamma L_1.wb \sim_\Gamma L_2.wb = W_2 +\!\!+ L_{20}.wb$. Because L_{20} has *high* type, it follows that $W_1^* \sim_\Gamma W_2 +\!\!+ L_{20}.wb \sim_\Gamma W_2$.
- And $L_{10}^* \sim_\Gamma^{\text{tso}} L_{20}$ follows from Lemma 18 and the definition of $\sim_\Gamma^{\text{tso}}$. □

Lemma 31. *If $\langle (X := i) +\!\!+ L_1, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle (X := i) +\!\!+ L_2, c_2\rangle_{\iota_2}$ then $\langle L_1, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle L_2, c_2\rangle_{\iota_2}$.*

Lemma 32. *Suppose $L_1 \sim_\Gamma^{\text{tso}} L_2$ and both $high; \Gamma \vdash^{\text{tso}} c_1$ and $high; \Gamma \vdash^{\text{tso}} c_2$. Then $\langle L_1, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle L_2, c_2\rangle_{\iota_2}$*

Proof. Let $L_1' = (L_1.mem, \emptyset, nil)$ and $L_2' = (L_2.mem, \emptyset, nil)$. From $L_1 \sim_\Gamma^{\text{tso}} L_2$, it follows that $W_1 = L_1.wb \sim_\Gamma L_2.wb = W_2$ and there is some $\lambda = L_1.locks = L_2.locks$. Conclude by defining $\mathcal{E} = (\lambda, [\cdot])$ and observing $\langle L_1, c_1\rangle_{\iota_1} = \mathcal{E}[W_1 \,|\, \langle L_1', c_1\rangle_{\iota_1}] \sim_\Gamma^{\text{tso}} \mathcal{E}[W_2 \,|\, \langle L_2', c_2\rangle_{\iota_2}] = \langle L_2, c_2\rangle_{\iota_2}$. □

3.7 Possibilistic Noninterference

Definition 11 (Possibilistic noninterference)**.** *We say that command c is* possibilistically noninterfering *(or* possibilistically secure*) under memory model* mm *and policy Γ if for all S_1, S_2 such that $S_1 \sim_\Gamma S_2$, if $((S_1, \mathbf{Lock}), \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{mm}*} (G_1', \mathfrak{o})$ then there exists G_2' such that $((S_2, \mathbf{Lock}), \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{mm}*} (G_2', \mathfrak{o})$ and $G_1'.mem \sim_\Gamma G_2'.mem$.*

3.8 Security

Lemma 33. *If $\mathcal{E} = (\lambda, \mathcal{C})$ and $(G, \mathcal{E}[W \,|\, \langle L, c\rangle]) \longrightarrow^{eval} (G', P')$ and $L.locks \supseteq \lambda \cap c.locks$, then either $c = \mathbf{skip}$ or both $P' = \mathcal{E}[W \,|\, \langle L', c'\rangle] \parallel P_0'$ and $(G, \mathcal{E}_\emptyset[W \,|\, \langle L, c\rangle]) \longrightarrow^{eval} (G', \mathcal{E}_\emptyset[W \,|\, \langle L', c'\rangle] \parallel P_0')$.*

Proof. by simple induction. The interesting cases occur when c has form **sync** or **holding**; these cases work because we know that if c references a lock in $\mathcal{E}$, that lock also occurs in L. □

Lemma 34. *Whenever $high; \Gamma \vdash^{\text{tso}} c$ it is the case that $c.locks = \emptyset$.*

Proof. by induction. □

Lemma 35 (Global confinement)**.** *Suppose $high; \Gamma \vdash^{\text{tso}} t$ and $(G, \mathcal{E}[W \,|\, t]) \longrightarrow^{op} (G', P')$. If $P' = \mathfrak{o}$ or $P' = \mathcal{E}[W \,|\, t'] \parallel P_0'$ then $(G, \mathcal{E}[W \,|\, t] \parallel \mathfrak{o}) \sim_\Gamma^{\text{tso}} (G', P')$.*

Proof. Assume $P' = \circ$ or $P' = \mathcal{E}[W \mid t'] \parallel P'_0$. Observe that either $op \neq commit$ or $W = nil$.

First we demonstrate $G \sim^{\text{tso}}_{\Gamma} G'$. It's necessary to show $G.locks = G'.locks$, which follows from inverting the typing relation finitely many times and observing that $\longrightarrow^{op}$ cannot contain a (nested) use of EC-SYNCACQUIRE or EC-HOLDRELEASE. Consider an arbitrary global variable X; it remains to show that $G(X) = G'(X)$ whenever $\Gamma(X) = low$. Suppose that $op \neq commit$; then $G.mem = G'.mem$ and we're done. Instead suppose that $op = commit$ and, consequently, $W = nil$. Then $G' = G[Y \mapsto i]$ where $t.wb = (Y := i)::W_0$. Inverting the typing of $t.wb$ yields $high \sqsubseteq \Gamma(Y)$ so $X \neq Y$ and $G(X) = G'(X)$.

Second we must show $\mathcal{E}[W \mid t] \parallel \circ \sim^{\text{tso}}_{\Gamma} P'$. If $P' = \circ$ then the process stepped by EC-REAP, so $\mathcal{E} = (\emptyset, [\cdot])$ and $\mathcal{E}[W \mid t] = t$. Thus we can conclude by noting $high; \Gamma \vdash^{\text{tso}} t$ implies $\mathcal{E}[W \mid t] \parallel \circ = t \parallel \circ \sim^{\text{tso}}_{\Gamma} \circ = P'$. If instead $P' = \mathcal{E}[W \mid t'] \parallel P'_0$ then it suffices to show $high; \Gamma \vdash^{\text{tso}} t'$ and $\overline{pc}; \Gamma \vdash^{\text{tso}} P'_0$ for $high \sqsubseteq \overline{pc}$, which follow from Lemma 10. ☐

Lemma 36 (Local Confinement)**.** *Suppose $high; \Gamma \vdash^{\text{tso}} t$ and $(G, \mathcal{E}[W \mid t]) \longrightarrow^{op} (G', \mathcal{E}[W \mid t'] \parallel P')$. Then $\mathcal{E}[W \mid t] \sim^{\text{tso}}_{\Gamma} \mathcal{E}[W \mid t']$.*

Proof. Let $t = \langle L, c\rangle$ and $t' = \langle L', c'\rangle$. As in the proof of Lemma 35, a low write, a low commit, a lock, or an unlock would contradict c's *high* type. Therefore $L \sim^{\text{tso}}_{\Gamma} L'$. It remains remains to show that $high; \Gamma \vdash^{\text{tso}} \langle L', c'\rangle$ and $\overline{pc}; \Gamma \vdash^{\text{tso}} P'$ for $high \sqsubseteq \overline{pc}$, which follow from Lemma 10. ☐

Lemma 37 (Contextual confinement, fork free)**.** *Suppose $\mathcal{T} :: (G, \mathcal{E}[W \mid t] \parallel \circ) \Longrightarrow^{\text{tso}*} (G', \mathcal{E}[W' \mid s'] \parallel \circ)$ and Simple $\mathcal{T}$. Further suppose $high; \Gamma \vdash^{\text{tso}} t$. Then there is some t' such that following hold:*

$$\begin{array}{rcl} \mathcal{E}[W \mid t'] & = & \mathcal{E}[W' \mid s'] \\ \mathcal{E}[W \mid t] & \sim^{\text{tso}}_{\Gamma} & \mathcal{E}[W \mid t'] \\ G & \sim^{\text{tso}}_{\Gamma} & G' \end{array}$$

Proof. By induction on the length of $\mathcal{T}$. If $\mathcal{T}$ contains a single element then $G' = G$, $\mathcal{E}[W \mid t] = \mathcal{E}[W' \mid s']$ and we conclude using that $\sim^{\text{tso}}_{\Gamma}$ is an equivalence relation.

Suppose $\mathcal{T}$ contains $n > 1$ elements. Because *high* commands cannot contain **fork**s, $\mathcal{T}$ witnesses an evaluation sequence of the following form:

$$(G, \mathcal{E}[W \mid t]) \Longrightarrow^{\text{tso}} (G_1, \mathcal{E}[W_1 \mid s_1]) \Longrightarrow^{\text{tso}*} (G', \mathcal{E}[W' \mid s'])$$

Because *Simple* $\mathcal{T}$ the first step is not a commit, so $W_1 +\!\!+ s.wb = W +\!\!+ t.wb$ or—if the step is by a write—$W_1 +\!\!+ s_1.wb = W +\!\!+ t.wb +\!\!+ (X := i)::nil$. In either case we can define W_{t_1} such that $W_1 +\!\!+ s_1.wb = W +\!\!+ W_{t_1}$. Define $t_1 = \langle (s_1.mem, s_1.locks, W_{t_1}), s_1.cmd\rangle$ and observe that $\mathcal{T}$ also witnesses the following:

$$(G, \mathcal{E}[W \mid t]) \Longrightarrow^{\text{tso}} (G_1, \mathcal{E}[W \mid t_1]) = (G_1, \mathcal{E}[W_1 \mid s_1]) \Longrightarrow^{\text{tso}*} (G', \mathcal{E}[W' \mid s'])$$

By the induction hypothesis $(G', \mathcal{E}[W' \mid s']) = (G', \mathcal{E}[W \mid t'])$ for some t'. Additionally $G_1 \sim^{\text{tso}}_{\Gamma} G'$ and $\mathcal{E}[W \mid t_1] \sim^{\text{tso}}_{\Gamma} \mathcal{E}[W \mid t'] = \mathcal{E}[W' \mid s']$. Using the transitivity of $\sim^{\text{tso}}_{\Gamma}$, it remains to show $G \sim^{\text{tso}}_{\Gamma} G_1$ and $\mathcal{E}[W \mid t] \sim^{\text{tso}}_{\Gamma} \mathcal{E}[W \mid t_1]$. These are consequences of Lemmas 35 and 36, respectively. ☐

Lemma 38 (SV Expression Confinement)**.** *Suppose $L_1 \sim^{\text{tso}}_{\Gamma} L_2$. Then $low; \Gamma \vdash^{\text{tso}} a$ implies $i_1 = i_2$ when $L_1[a] \Downarrow i_1$ and $L_2[a] \Downarrow i_2$. Likewise, $low; \Gamma \vdash^{\text{tso}} b$ implies $\beta_1 = \beta_2$ when $L_1[b] \Downarrow \beta_1$ and $L_2[b] \Downarrow \beta_2$.*

Proof. by induction ☐

Lemma 39 (SV Commit step security)**.** *Suppose $(G_1, \langle L_1, c_1\rangle_{\iota_1}) \longrightarrow^{commit} (G'_1, P'_1)$ and $pc; \Gamma \vdash^{\text{tso}} \langle L_1, c_1\rangle_{\iota_1}$. Further assume both $\langle L_1, c_1\rangle_{\iota_1} \sim^{\text{tso}}_{\Gamma} t_2$ and $G_1 \sim^{\text{tso}}_{\Gamma} G_2$. Then there exist L'_1, G'_2, and t'_2 such that $P'_1 = \langle L'_1, c_1\rangle_{\iota_1} \parallel \circ$ and $G'_1 \sim^{\text{tso}}_{\Gamma} G'_2$ and $\langle L'_1, c_1\rangle_{\iota_1} \sim^{\text{tso}}_{\Gamma} t'_2$ and $(G_2, t_2 \parallel \circ) \Longrightarrow^{\text{tso}*} (G'_2, t'_2 \parallel \circ)$.*

Proof. Let $\langle L_2, c_2\rangle_{\iota_2} = t_2$. Proof is by induction on the structure of $L_2.wb$. Inverting the evaluation relation shows for some X, i, and L_{10}, and that

$$\begin{aligned} L_1 &= (X := i) +\!\!+ L_{10} \\ G'_1 &= G_1[X \mapsto i] \\ P'_1 &= \langle L_{10}, c_1\rangle \parallel \circ. \end{aligned}$$

Let L_{10} be the witness to L'_1.

If $\Gamma(X) = \mathit{high}$ then by definition $G_1' \sim_\Gamma^{\text{tso}} G_1 \sim_\Gamma^{\text{tso}} G_2$ and $L_1 \sim_\Gamma^{\text{tso}} L_{10}$. So taking $G_2' = G_2$ and $t_2' = t_2$, it suffices to show $\langle L_{10}, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} \langle L_2, c_2\rangle_{\iota_2}$, a consequence of Lemma 30.

Suppose instead that $\Gamma(X) = \mathit{low}$. Were $L_2.\mathit{wb}$ empty this would contradict the assumption that $t_1 \sim_\Gamma^{\text{tso}} t_2$, making the conclusion trivial. If $L_2.\mathit{wb} = (X := j) +\!\!+ L_{20}$ then, by the definition of $\sim_\Gamma$, $i = j$ and $L_{10}.\mathit{wb} \sim_\Gamma L_{20}.\mathit{wb}$. Because $(G_2, t_2) \longrightarrow^{\mathit{commit}} (G_2[X \mapsto i], \langle L_{20}, c_2\rangle_{\iota_2})$ it suffices to show $G_1[X \mapsto i] \sim_\Gamma^{\text{tso}} G_2[X \mapsto i]$, which follows from by Lemma 27, and $\langle L_{10}, c_1\rangle \sim_\Gamma^{\text{tso}} \langle L_{20}, c_2\rangle_{\iota_2}$, which follows from Lemma 31.

Finally if $L_2.\mathit{wb} = (Y := j) +\!\!+ L_{20}$ with $Y \neq X$ the definition of $\sim_\Gamma$ requires that $\Gamma(Y) = \mathit{high}$. So $(G_2, t_2) \Longrightarrow^{\mathsf{tso}} (G_2[Y \mapsto j], \langle L_{20}, c_2\rangle)$ where $G_1 \sim_\Gamma^{\text{tso}} G_2[Y \mapsto j]$ and $\langle L_1, c_1\rangle \sim_\Gamma^{\text{tso}} \langle L_{20}, c_2\rangle$. The induction hypothesis yields

$$(G_2[Y \mapsto j], \langle L_{20}, c_2\rangle) \Longrightarrow^{\mathsf{tso}*} (G_2', t_2' \parallel \mathsf{o})$$

where $G_1' \sim_\Gamma^{\text{tso}} G_2'$ and $\langle L_{10}, c_1\rangle_{\iota_1} \sim_\Gamma^{\text{tso}} t_2'$. Conclude by deriving that $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', t_2' \parallel \mathsf{o})$. □

Lemma 40 (SV Eval step security)**.** *Suppose $(G_1, t_1) \longrightarrow^{\mathit{eval}} (G_1', P_1')$ and $pc; \Gamma \vdash^{\text{tso}} t_1$ and wellStruct t_1. Further assume both $t_1 \sim_\Gamma^{\text{tso}} t_2$ and $G_1 \sim_\Gamma^{\text{tso}} G_2$. Then there exist G_2' and P_2' such that $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$ and $(G_1', P_1') \sim_\Gamma^{\text{tso}} (G_2', P_2')$.*

Proof. Let $t_1 = \langle L_1, c_1\rangle_\iota$. We strengthen the induction hypothesis by also requiring P_2' satisfy the following property: When $P_1' = t_{10}' \parallel P_{10}'$ then $P_2' = t_{20}' \parallel P_{20}'$ and *FrontReapFree* $\mathcal{T}$ where $\mathcal{T} :: (G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$, and both $t_{10}' \sim_\Gamma^{\text{tso}} t_{20}'$ and $P_{10}' \sim_\Gamma^{\text{tso}} P_{20}'$. Proceed by induction on the structure of c_1, grouping cases in a non-standard way.

To avoid pointless textual copying we will let several cases refer to other parts of the argument. There is no circularity and the dependencies are as follows. In Case 2, subcases ec-HoldStep and ec-SeqStruct refer to Case 1 and no other (sub)cases. Case 1 refers to Case 2, subcases ec-HoldRelease, ec-Reap, ec-SeqSkip, and no other (sub)cases.

Consider the two principle cases arising from $t_1 \sim_\Gamma^{\text{tso}} t_2$.

- Case 1: $t_1 = \mathcal{E}[W_1 \mid \langle L_{10}, c_{10}\rangle]$ and $t_2 = \mathcal{E}[W_2 \mid \langle L_{20}, c_{20}\rangle]$ where $\mathit{high}; \Gamma \vdash^{\text{tso}} \langle L_{10}, c_{10}\rangle$ and $\mathit{high}; \Gamma \vdash^{\text{tso}} \langle L_{20}, c_{20}\rangle$ and both $L_{10} \sim_\Gamma^{\text{tso}} L_{20}$ and $W_1 \sim_\Gamma W_2$. Lemmas 33 and 34 show we must examine two situations:

  - Suppose $c_{10} = \mathbf{skip}$. We will show that t_2 evaluates to a state such that we can finish by copying reasoning from the ec-HoldStep, ec-SeqSkip, or ec-Reap cases below. Lemma 14 gives a simple derivation of

    $$(G_2, \mathcal{E}_2[W_1 \mid \langle L_{20}, c_{20}\rangle]) \Longrightarrow^{\mathsf{tso}*} (G_{20}', \mathcal{E}[W_2 \mid \langle L_{20}', \mathbf{skip}\rangle] \parallel \mathsf{o}).$$

    (Note that *active* $\mathcal{E}$ because t_1 steps and $L_{10}.\mathit{locks} = \emptyset$.) It now suffices to find witnesses G_2' and P_2' such that $(G_{20}', \mathcal{E}[W_2 \mid \langle L_{20}', \mathbf{skip}\rangle] \parallel \mathsf{o}) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$ and $G_1' \sim_\Gamma^{\text{tso}} G_2'$ and P_2' satisfies the appropriate properties (above). Copying from case 2 can provide such witnesses, but requires we first demonstrate two equivalences. $G_1 \sim_\Gamma^{\text{tso}} G_2 \sim_\Gamma^{\text{tso}} G_{20}'$ and $t_1 \sim_\Gamma^{\text{tso}} \mathcal{E}[W_2 \mid \langle L_{20}, c_{20}\rangle] \sim_\Gamma^{\text{tso}} \mathcal{E}[W_2 \mid \langle L_{20}', c_{20}'\rangle]$ follow from Lemma 37 and the transitivity of $\sim_\Gamma^{\text{tso}}$.
  - Suppose instead that, for some t_{10}', it is the case that $P_1' = \mathcal{E}[W \mid t_{10}'] \parallel P_{10}'$ and $(G_1, \mathcal{E}_\emptyset[W_1 \mid \langle L_{10}, c_{10}\rangle]) \longrightarrow^{\mathit{eval}} (G_1', \mathcal{E}_\emptyset[W_1 \mid t_{10}'] \parallel P_{10}')$. Take witnesses G_2' and P_2' to be G_2 and P_2. It suffices to show that $G_1 \sim_\Gamma^{\text{tso}} G_1'$ and $t_1 \parallel \mathsf{o} \sim_\Gamma^{\text{tso}} P_1'$. The former is a consequence of Lemma 35. Toward the latter we use Lemma 12 to establish that $\overline{pc}; \Gamma \vdash^{\text{tso}} P_1'$ where $\mathit{high} \sqsubseteq \overline{pc}$ so $P_{10}' \sim_\Gamma^{\text{tso}} \mathsf{o}$. It remains to show that $t_1 \sim_\Gamma^{\text{tso}} \mathcal{E}[W \mid t_{10}']$, which follows from Lemmas 1, 6, 8, 15, 34, and 36. .

- Case 2: $t_2 = \langle c_2, L_2\rangle$ where $c_1 = c_2$ and $L_1 \sim_\Gamma^{\text{tso}} L_2$. Continue by inverting the $\longrightarrow^{\mathit{eval}}$ relation.

  ec-Store: Here $c_1 = X := x$. If $\Gamma(x) = \mathit{low}$, the definition of $\sim_\Gamma^{\text{tso}}$ shows $L_1(x) = L_2(x)$ and we conclude using Lemma 22. Otherwise $\Gamma(x) = \mathit{high}$, inverting typing rule tso-Store gives $\Gamma(X) = \mathit{high}$, and the result follows from Lemma 23.

  ec-Load: Here $c_1 = x := Y$. If $\Gamma(Y) = \mathit{low}$ then, by Lemma 26, $(G_1.\mathit{mem}; L_1.\mathit{wb})[Y] \Downarrow i$ and $(G_2.\mathit{mem}; L_2.\mathit{wb})[Y] \Downarrow i$ for some i. Conclude using Lemma 24. Suppose instead $\Gamma(Y) = \mathit{high}$. Inverting typing rule tso-Load shows $\Gamma(x) = \mathit{high}$ and we conclude via Lemma 25.

  ec-EvalExp: Here $c_1 = x := a$. Suppose $\mathit{low} \vdash \Gamma : a$ then by Lemma 38 there exists a unique i such that $L_1[a] \Downarrow i$ and $L_2[a] \Downarrow i$, and we can conclude via Lemma 24. Suppose instead $\mathit{high} \vdash \Gamma : a$, then by inverting typing tso-EvalExp we find $\Gamma(x) = \mathit{high}$. Conclude via Lemma 25.

EC-SYNCACQUIRE: Here $c_1 = \textbf{sync}\ \ell\ \textbf{do}\ c$ with $\ell \in G_1$ and $L_1.wb = nil$. As $L_1 \sim^{\text{tso}}_{\Gamma} L_2$, any store $(X := i)$ in $L_2.wb$ must have $\Gamma(X) = high$. As in the $op = commit$ case use an inner induction on the structure of $L_2.wb$ to find G_{20} and L_{20} where $(G_2, t_2 \parallel \mathsf{o}) \Longrightarrow^{\mathsf{tso}*} (G_{20}, \langle L_{20}, \textbf{sync}\ \ell\ \textbf{do}\ c\rangle \parallel \mathsf{o})$ and both $G_{20} \sim^{\text{tso}}_{\Gamma} G_2 \sim^{\text{tso}}_{\Gamma} G_1$ and $L_{20} \sim^{\text{tso}}_{\Gamma} L_2 \sim^{\text{tso}}_{\Gamma} L_1$. Via the first equivalence, $\ell \in G_{20}$ so using EC-SYNCACQUIRE we find $(G_2, t_2 \parallel \mathsf{o}) \Longrightarrow^{\mathsf{tso}*} (G_{20}, \langle L_{20}, \textbf{holding}\ \ell\ \textbf{do}\ c\rangle \parallel \mathsf{o})$.

EC-HOLDRELEASE, EC-FENCE, EC-FORK: Similar to the EC-SYNCACQUIRE case.

EC-SYNCREENTER: Trivial after noting $L_1 \sim^{\text{tso}}_{\Gamma} L_2$ gives $L_1.locks = L_2.locks$.

EC-HOLDSTEP: Here, $c_1 = \textbf{holding}\ \ell\ \textbf{do}\ c$ and $P'_1 = \langle L'_1, \textbf{holding}\ \ell\ \textbf{do}\ c'_{10}\rangle_\iota \parallel P'_{10}$.

Inversion and the induction hypothesis give

$$\ell \in L_1$$

$$(G_1, \langle L_1, c\rangle_\iota) \longrightarrow^{eval} (G'_1, \langle L'_1, c'_{10}\rangle_\iota \parallel P'_{10})$$
$$\mathcal{T} :: (G_2, \langle L_2, c\rangle_\iota) \Longrightarrow^{\mathsf{tso}*} (G'_2, \langle L'_2, c'_{20}\rangle_\iota \parallel P'_{20}).$$
$$\textit{FrontReapFree}\ \mathcal{T}$$

From $L_1 \sim^{\text{tso}}_{\Gamma} L_2$ it follows that $\ell \in L_2$ and we can define $\mathcal{E} = (\{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ [\cdot])$ where $active\ \mathcal{E}$ and $t_2 = \mathcal{E}[nil \,|\, \langle L_2, c\rangle_\iota]$. By Lemma 17,

$$\begin{array}{rcl} \mathcal{T}' :: (G_2, t_2 \parallel \mathsf{o}) & = & (G_2, \mathcal{E}[nil \,|\, \langle L_2, c\rangle_\iota]) \\ & \Longrightarrow^{\mathsf{tso}*} & (G'_2, \mathcal{E}[nil \,|\, \langle L'_2, c'_{20}\rangle_\iota] \parallel P'_{20}) \end{array}$$

where $\textit{FrontReapFree}\ \mathcal{T}'$.

It is necessary to show the following:

$$\begin{array}{rcl} G'_1 & \sim^{\text{tso}}_{\Gamma} & G'_2 \\ \mathcal{E}[nil \,|\, \langle L'_1, c'_{10}\rangle_\iota] & \sim^{\text{tso}}_{\Gamma} & \mathcal{E}[nil \,|\, \langle L'_2, c'_{20}\rangle_\iota] \\ P'_{10} & \sim^{\text{tso}}_{\Gamma} & P'_{20} \end{array}$$

These are implied by the above use of induction hypothesis and Lemma 28.

EC-SEQSTRUCT: Similar to EC-HOLDSTEP.

EC-SEQSKIP: Immediate.

EC-IFTRUE: Here $c_1 = c_2 = \textbf{if}\ b\ \textbf{do}\ c_t\ \textbf{else}\ c_f$ and inversion shows that $L_1[b] \Downarrow \textbf{true}$ and $P'_1 = \langle L_1, c_t\rangle$. Suppose that it's not the case that $\Gamma \vdash b : low$. Then inverting the typing relation shows both $high; \Gamma \vdash^{\text{tso}} c_t$ and $high; \Gamma \vdash^{\text{tso}} c_f$. Thread t_2 could potentially step to configuration $(G_2, \langle L_2, c_t\rangle \parallel \mathsf{o})$ or to configuration $(G_2, \langle L_2, c_f\rangle \parallel \mathsf{o})$. Without loss of generality assume the latter. Use Lemma 32 to establish $\langle L_1, c_t\rangle \sim^{\text{tso}}_{\Gamma} \langle L_2, c_f\rangle$. All other goals are immediate.

Suppose instead that $\Gamma \vdash b : low$. Then Lemma 38 shows $L_2[b] \Downarrow \textbf{true}$ and we have a reap-free trace showing $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2, \langle L_2, c_t\rangle \parallel \mathsf{o})$.

EC-IFFALSE, EC-WHILETRUE, EC-WHILEFALSE: Similar to, or simpler than, the EC-IFTRUE case.

EC-REAP: Immediate, noting that the trace we construct need not be *FrontReapFree*. □

Theorem 1 (SV Security)**.** *Suppose* $(G_1, P_1) \sim^{\text{tso}}_{\Gamma} (G_2, P_2)$ *and* $\overline{pc}; \Gamma \vdash^{\text{tso}} P_1$ *and wellStruct* P_1. *Suppose also that* $(G_1, P_1) \Longrightarrow^{\mathsf{tso}} (G'_1, P'_1)$. *Then there exists* G'_2, P'_2 *such that* $(G'_1, P'_1) \sim^{\text{tso}}_{\Gamma} (G'_2, P'_2)$ *and* $(G_2, P_2) \Longrightarrow^{\mathsf{tso}*} (G'_2, P'_2)$.

Proof. Inverting the tso-evaluation relation and appealing to Lemma 19 gives

$$P_1 = P_{11} \parallel t_1 \parallel P_{12}$$
$$P_2 = P_{21} \parallel P^*_2 \parallel P_{22}$$
$$P'_1 = P_{11} \parallel Q_1 \parallel P_{12}$$

where P^*_2 contains at most one thread (i.e., $P^*_2 \in \{\mathsf{o}, t_2 \parallel \mathsf{o}\}$ for some t_2) and the following hold:

$$(G, t_1) \longrightarrow^{op} (G'_1, Q_1)$$

$$P_{11} \sim_{\Gamma}^{\text{tso}} P_{21}$$
$$t \parallel \circ \sim_{\Gamma}^{\text{tso}} P_2^*$$
$$P_{12} \sim_{\Gamma}^{\text{tso}} P_{22}$$

It suffices to show that there exists G_2' and Q_2 such that $(G_1', Q_1) \sim_{\Gamma}^{\text{tso}} (G_2', Q_2)$ and $(G_2, P_2^*) \implies^{\mathsf{tso}*} (G_2', Q_2)$. (Observe that while we could rename threads in Q_2, we do not need to; thread names are only really relevant for the data-race freedom argument.) Inspecting the definition of $\sim_{\Gamma}^{\text{tso}}$ shows there are only three ways in which to find $t \parallel \circ \sim_{\Gamma}^{\text{tso}} P_2^*$. Proceed by case analysis.

First suppose that that the equivalence arises from Definition 10, clause 4b. Here $\mathit{high}; \Gamma \vdash^{\text{tso}} t_1$ and via Lemma 35, $(G_1, t \parallel \circ) \sim_{\Gamma}^{\text{tso}} (G_1', Q_1)$. Note that to apply Lemma 35 we take $\mathcal{E} = (\emptyset, [\cdot])$ and $W = \mathit{nil}$. Conclude using Lemma 18 and taking G_2 and P_2^* as existential witnesses G_2' and Q_2.

Second suppose that that the equivalence arises from Definition 10, clause 4c. Here $P_2^* = t_2 \parallel \circ$ where $\mathit{high}; \Gamma \vdash^{\text{tso}} t_2$ and $t_1 \sim_{\Gamma}^{\text{tso}} \circ$. From $t_1 \sim_{\Gamma}^{\text{tso}} \circ$ it follows that $\mathit{high}; \Gamma \vdash^{\text{tso}} t_1$. Again taking G_2 and P_2^* to be witnesses G_2' and Q' conclude with the following equational reasoning:

$$\begin{array}{rcll}
(G', Q_1) & \sim_{\Gamma}^{\text{tso}} & (G_1, t_1 \parallel \circ) & \text{by Lemma 35} \\
& \sim_{\Gamma}^{\text{tso}} & (G_1, \circ) & \\
& \sim_{\Gamma}^{\text{tso}} & (G_2, \circ) & \text{by assumption} \\
& \sim_{\Gamma}^{\text{tso}} & (G_2, t_2 \parallel \circ) & \\
& = & (G_2, P_2^*) &
\end{array}$$

Third suppose that that the equivalence arises from Definition 10, clause 4d. Here $P_2^* = t_2 \parallel \circ$ for some t_2 with $t_1 \sim_{\Gamma}^{\text{tso}} t_2$. Finitely many inversions of the typing relation show $pc; \Gamma \vdash^{\text{tso}} t_1$ for some pc. Similarly, *wellStruct* t_1. Conclude via Lemmas 39 and 40. ☐

Corollary 1. *Suppose* $(G_1, P_1) \sim_{\Gamma}^{\text{tso}} (G_2, P_2)$ *and* $\overline{pc}; \Gamma \vdash^{\text{tso}} P_1$ *and wellStruct* P_1. *Suppose also that* $(G_1, P_1) \implies^{\mathsf{tso}*} (G_1', P_1')$. *Then there exists* G_2', P_2' *such that* $(G_1', P_1') \sim_{\Gamma}^{\text{tso}} (G_2', P_2')$ *and* $(G_2, P_2) \implies^{\mathsf{tso}*} (G_2', P_2')$.

Proof. By finitely many application of Lemmas 5 and 12 and Theorem 1. ☐

Corollary 2. *Suppose* $G_1 \sim_{\Gamma}^{\text{tso}} G_2$ *and* $pc; \Gamma \vdash^{\text{tso}} t$ *and src* $t.cmd$. *If* $(G_1, t) \implies^{\mathsf{tso}*} (G_1', \circ)$ *then* $(G_2, t) \implies^{\mathsf{tso}*} (G_2', \circ)$ *for some* G_2' *where* $G_1' \sim_{\Gamma}^{\text{tso}} G_2'$.

Proof. An instantiation of the first corollary of Theorem 1 shows that (G_2, t) reduces to a configuration where we may conclude with several applications of Lemma 14 and applications of EC-REAP. ☐

Corollary 3 (TSO Simple possibilistic noninterference)**.** *Suppose* $pc; \Gamma \vdash^{\text{tso}} c$ *and src* c. *Then* c *is possibilistically noninterfering under* tso *and* Γ.

Proof. Immediate from the second corollary of Theorem 1. ☐

4 Data-Race Freedom

We want to show that our TSO and SC machines are equivalent for data-race free programs.

4.1 SC Executions are a Subset of TSO Executions

First, it is clear that every possibilistic SC execution is also a possibilistic TSO execution. We formalize this with the following lemma:

Lemma 41 (SC-Eval Implies TSO-Eval)**.** *If* $(G, P) \implies^{\mathsf{sc}} (G', P')$ *then* $(G, P) \implies^{\mathsf{tso}} (G', P')$.

Proof. Since $(G, P) \implies^{\mathsf{sc}} (G', P')$ we have that $P = t_1 \ldots t_{i-1} \parallel t_i \parallel t_{i+1} \ldots t_n$ and $(G, t_i) \longrightarrow^{op} (G', Q)$ and $P' = t_1 \ldots t_{i-1} \parallel Q \parallel t_{i+1} \ldots t_n$. Therefore also $(G, P) \implies^{\mathsf{tso}} (G', P')$. ☐

Corollary 4 (SC-Eval* Implies TSO-Eval*)**.** *If* $(G, P) \implies^{\mathsf{sc}*} (G', P')$ *then* $(G, P) \implies^{\mathsf{tso}*} (G', P')$.

4.2 Definition of Data Race Freedom

Definition 12 (Reads Next)**.** *Thread* $\langle L, c\rangle$ reads X next *if one of the following conditions holds:*

- *c has the form $x := X$*
- *c has the form $c_1; c_2$ and $\langle L, c_1\rangle$ reads X next*
- *c has the form* **holding** ℓ **do** c' *and $\ell \in L$ and $\langle L, c'\rangle$ reads X next*

Definition 13 (Writes Next)**.** *Thread* $\langle L, c\rangle$ writes X next *if one of the following conditions holds:*

- *c has the form $X := x$*
- *c has the form $c_1; c_2$ and $\langle L, c_1\rangle$ writes X next*
- *c has the form* **holding** ℓ **do** c' *and $\ell \in L$ and $\langle L, c'\rangle$ writes X next*

Definition 14 (Accesses Next)**.** *Thread t* accesses X next *if either t reads X next or t writes X next.*

Definition 15 (Conflicting Threads)**.** *Threads s and t* conflict *if there exists a variable X such that each thread accesses X next and at least one thread writes X next.*

Definition 16 (Race-Exhibiting Process Soup)**.** *Process soup P* exhibits a race *if it contains two distinct threads that conflict.*

Definition 17 (Race-Free Configuration)**.** *Configuration (G, P) is* race-free *if for all G' and P' such that $(G, P) \Longrightarrow^{\mathsf{sc}*} (G', P')$, it is not the case that P' exhibits a race.*

4.3 TSO Executions are a Subset of SC Executions for DRF Programs

4.3.1 Definitions

It will be convenient to associate each step of computation with its *action* – the kind of step taken (commit or eval) and the thread that takes the step. Accordingly we annotate our evaluation steps as follows:

$$\frac{P = t_1 \dots t_{i-1} \parallel t_i \parallel t_{i+1} \dots t_n \qquad \begin{array}{c} op(i) \in \mathrm{ReadySC}(\chi) \\ (G, t_i) \longrightarrow^{op} (G', Q) \end{array} \qquad P' = t_1 \dots t_{i-1} \parallel Q \parallel t_{i+1} \dots t_n}{(G, P) \Longrightarrow^{\mathsf{sc}}_{op(i)} (G', P')}$$

$$\frac{P = t_1 \dots t_{i-1} \parallel t_i \parallel t_{i+1} \dots t_n \qquad (G, t_i) \longrightarrow^{op} (G', Q) \qquad P' = t_1 \dots t_{i-1} \parallel Q \parallel t_{i+1} \dots t_n}{(G, P) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P')}$$

Next we define an invariant on any configuration in a valid TSO execution of a program:

Definition 18. *Consider configuration $G, t_1 \parallel \dots \parallel t_n$ Such a configuration is* well-locked *if lock sets $G.locks$, $t_1.locks, \dots, t_n.locks$ are pairwise disjoint.*

Now we define an invariant on any configuration in the TSO execution of a race-free program:

Definition 19 (Well-Behaved Configuration)**.** *A configuration (G, P) is* well behaved *if the following conditions hold, where $P = \langle L_1, c_1\rangle \parallel \dots \parallel \langle L_n, c_n\rangle$:*

- *P does not exhibit a race*
- *(G, P) is well locked*
- *If $L_i.wb$ contains a write (possibly multiple) to some variable X, then for each $j \neq i$, $L_j.wb$ does not contain a write to X and c_j does not access X next.*

Next we need to formalize the relationship between a TSO execution and a corresponding SC execution. The two executions will not stay in lockstep but their configurations will always have a strong relationship (if the original program is data-race-free). Let $\hat{L}$ denote the "cleared" version of L — the local state identical to L but with an empty write buffer. That is, $\hat{L}.mem = L.mem$ and $\hat{L}.locks = L.locks$ and $\hat{L}.wb = nil$.

Definition 20. *Let* $P = \langle L_1, c_1 \rangle_{\iota_1} \| \ldots \| \langle L_n, c_n \rangle_{\iota_n}$. *We say that* (H, Q) *is the commit closure of* (G, P) *if the following conditions hold:*

1. $Q = \langle \hat{L}_1, c_1 \rangle_{\iota_1} \| \ldots \| \langle \hat{L}_n, c_n \rangle_{\iota_n}$
2. $H.locks = G.locks$
3. *For each variable* X *that does not appear in any write buffer in* P*, we have* $H.mem\ (X) = G.mem\ (X)$.
4. *For each variable* X *that appears in the write buffer of some thread* i *in* P*, there exists a* k *such that* $H.mem\ (X) = k$ *and* $(G.mem; L_i.wb)[X] \Downarrow k$.

Intuitively, if (H, Q) is the commit closure (G, P) and (G, P) is well behaved then (H, Q) is the unique configuration that results from executing commit operations in any order from (G, P) until all write buffers are empty.

4.3.2 Simple Lemmas

Notation: We denote by $P.ls(i)$ the local state in thread i of P and by $P.cmd(i)$ the command in thread i of P. We denote by $P[i \mapsto t]$ the process soup identical to P but with the ith thread replaced by t. We use $P[i \mapsto L]$ as shorthand for $P[i \mapsto \langle L, P.cmd(i) \rangle]$ and $P[i \mapsto c]$ as shorthand for $P[i \mapsto \langle P.ls(i), c \rangle]$. For all of these notations, we also allow the thread to be indexed by its TID rather than its position. Finally, we use $P[i \mapsto Q]$ to denote the process soup $t_1 \ldots t_{i-1} \| Q \| t_{i+1} \ldots t_n$, where $P = t_1 \ldots t_{i-1} \| t_i \| t_{i+1} \ldots t_n$.

Lemma 42 (Preservation of well-lockedness)**.** *Suppose* (G, P) *is well-locked and* $(G, P) \Longrightarrow^{\mathsf{tso}} (G', P')$. *Then* (G', P') *is also well locked.*

Lemma 43. *If* $(S; W)[Y] \Downarrow j$ *and* $X \neq Y$*, then* $(S[X \mapsto k]; W)[Y] \Downarrow j$.

Lemma 44. *If* $(S; (X := k)::W)[Y] \Downarrow j$ *and* $X \neq Y$*, then* $(S; W)[Y] \Downarrow j$.

Lemma 45. *If* $(S; W)[Y] \Downarrow j$ *and* $X \neq Y$*, then* $(S; W +\!\!+ (X := k))[Y] \Downarrow j$.

Lemma 46. *If* $(S; (X := k)::W)[X] \Downarrow j$ *then* $(S[X \mapsto k].mem; W)[X] \Downarrow j$.

Lemma 47. *If* $(S; W)[X] \Downarrow k$ *and* X *does not appear in* W*, then* $S(X) = k$.

Lemma 48. *If* (G, P) *is well behaved and* (H, Q) *is the commit closure of* (G, P) *and* $P.cmd(i)$ *accesses* X *next and* $(G.mem; P.ls(i).wb)[X] \Downarrow k$*, then* $H.mem(X) = k$.

Lemma 49. $\langle L, c_1; c_2 \rangle$ *reads [writes]* X *next if and only if* $\langle L, c_1 \rangle$ *reads [writes]* X *next.*

Lemma 50. *If* $\langle L, \textbf{holding}\ \ell\ \textbf{do}\ c \rangle$ *reads [writes]* X *next then* $\langle L, c \rangle$ *reads [writes]* X *next. If* $\langle L, c \rangle$ *reads [writes]* X *next then* $\langle L \cup \{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ c \rangle$ *reads [writes]* X *next.*

Lemma 51. *Suppose* (G, P) *is well behaved, let* $\langle L, c \rangle$ *be the ith thread of* P*, and let* c' *be a command. If the set of variables that* $\langle L, c' \rangle$ *reads next are a subset of those that* $\langle L, c \rangle$ *accesses reads next, and similarly for writes, then* $(G, P[i \mapsto c'])$ *is well behaved.*

Lemma 52. *If* (H, Q) *is the commit closure of* (G, P)*, then* $(H, Q[i \mapsto c])$ *is the commit closure of* $(G, P[i \mapsto c])$.

Lemma 53. *If* P *exhibits a race and* (H, Q) *is the commit closure of* (G, P)*, then* Q *exhibits a race.*

Lemma 54. *If* $(G, \langle L, c \rangle) \longrightarrow^{eval} (G', P')$ *and* c *has the form* $\mathcal{C}[X := x]$ *then* $\langle L, c \rangle$ *writes* X *next.*

Notation: We denote by P/ι the process soup identical to P but with thread ι removed. We denote by G/X the global state identical to G but with the value of X set to 0.

4.3.3 Key Lemmas

Lemma 55. *Suppose (G, P) is well behaved and $(G, P) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P')$ and (H, Q) is the commit closure of (G, P).*

1. *If op = commit then (H, Q) is the commit closure of (G', P').*
2. *If op = eval and thread t_i writes some variable X next, then there exist (H_0, Q_o) and (H', Q') such that $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0)$ and $(H_0, Q_0) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q')$, where (H', Q') is the commit closure of (G', P').*
3. *If op = eval and thread t_i does not write any variable next, then there exists (H', Q') such that $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where (H', Q') is the commit closure of (G', P').*

Proof. So we have $P = t_1 \dots t_{i-1} \parallel t_i \parallel t_{i+1} \dots t_n$ and $(G, t_i) \longrightarrow^{op} (G', P_0)$ and $P' = t_1 \dots t_{i-1} \parallel P_0 \parallel t_{i+1} \dots t_n$. Let $t_i = \langle L_i, c_i \rangle$.

1. Then L_i has the form $(X := k)::L_0$ and $P_0 = \langle L_0, c_i \rangle$ and $G' = G[X \mapsto k]$. Then $\hat{L}_i = \hat{L}_0$, so since thread i is the only one that is modified from P to P', condition 1 of the commit closure holds. Since locks are unchanged from G to G', condition 2 holds as well.

   Consider a variable $Y \neq X$. If Y does not appear in any write buffer of P then we have $H.mem\ (Y) = G.mem\ (Y)$. By definition also Y is not in any write buffer of P', and $G'.mem\ (Y) = G.mem\ (Y)$, so condition 3 is satisfied for Y. If Y appears in the write buffer of some thread j in P, then $H.mem\ (Y) = n$ and $(G.mem; L_j.wb)[Y] \Downarrow n$ for some value n. By definition Y also appears in the write buffer of thread j in P'. If $j \neq i$ then by Lemma 43 we have $(G'.mem; L_j.wb)[Y] \Downarrow n$, satisfying condition 4. If $j = i$ then by Lemmas 44 and 43 we have $(G'.mem; L_0.wb)[Y] \Downarrow n$, satisfying condition 4.

   Finally consider variable X. Since X is in the write buffer of L_i we know that $H.mem\ (X) = n$ and $(G.mem; L_i.wb)[X] \Downarrow n$ for some n. Further since (G, P) is well behaved we know that X is not in the write buffer of any thread other than i in P and therefore also in P'. By Lemma 46 we have $(G'.mem; L_0.wb)[X] \Downarrow n$. If X is in the write buffer of L_0 then we have satisfied condition 4. Otherwise X is not in any write buffer in P'. By Lemma 47 we have that $G'.mem\ (X) = n$, so condition 3 is satisfied for X.

2. We proceed by induction on the derivation of $(G, t_i) \longrightarrow^{op} (G', P_0)$. Case analysis on the form of c_i, which we are given writes variable X next:

   $X := x$ Then rule EC-STORE is used to step, so $L'_i = L_i +\!\!+ (X := L_i(x))$ and $P_0 = \langle L'_i, \textbf{skip} \rangle$ and $G' = G$.

   I claim that $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0)$ and $(H_0, Q_0) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q')$ for some H_0, Q_0, H', and Q'. Since (H, Q) is the commit closure of (G, P) we know that the ith thread of (H, Q) is $\langle \hat{L}_i, c_i \rangle$. Further we know that all write buffers in Q are empty, so no commit actions are ready on any thread. Therefore, the first step follows by EC-STORE, which then enables the commit step. By definition $\hat{L}_i.mem = L_i.mem$, so $\hat{L}_i(x) = L_i(x)$, so $H' = H[X \mapsto L_i(x)]$. Also, Q' is identical to Q except the ith thread's command is **skip**.

   We now prove that (H', Q') is the commit closure of (G', P'). Condition 1 holds for all threads other than i, since these threads are unchanged from Q and P and we have that (H, Q) is the commit closure of (G, P). For thread i the result follows by definition of these threads in P' and Q' and the fact that $\hat{L}_i$ is also the "cleared" version of $L_i +\!\!+ (X := L_i(x))$. Condition 2 follows from the fact that locks are unchanged from G to G' and H to H'.

   For Condition 3, let S be the set of variables that do not appear in any write buffer in P. Then $S - \{X\}$ is the set of variables that do not appear in any write buffer in P'. For each such variable Y, $G(Y) = G'(Y)$ and $H(Y) = H'(Y)$ by the definition of G' and H'. Then the result follows from the fact that $G(Y) = H(Y)$.

   For Condition 4, first consider a variable $Y \neq X$ that appears in the write buffer of some thread j in P'. Then it also appears in the write buffer of thread j in P, so we know that $H.mem\ (Y) = k$ and $(G.mem; L_j.wb)[Y] \Downarrow k$, where L_j is the local state of thread j in P. Then by definition of H' also $H'.mem\ (Y) = k$, and since $G' = G$ we have $(G'.mem; L_j.wb)[Y] \Downarrow k$. If $j \neq i$ we are done; otherwise by Lemma 45 we have $(G'.mem; L'_i.wb)[Y] \Downarrow k$.

Finally consider variable X. Since t_i writes X next and (G, P) is well behaved, we know that X is not in the write buffer of any thread other than i in P, and hence also in P'. By the definition of L'_i and the lookup procedure we have that $(G'.mem; L'_i.wb)[X] \Downarrow L_i(x)$. By definition of H' we have $H'(X) = L_i(x)$ so the result follows.

$c_{i1}; c_{i2}$ Since $\langle L_i, c_{i1}\rangle$ writes X next we know that c_{i1} is not **skip**, so the step must occur with rule EC-SEQSTRUCT. Therefore we have $(G, \langle L_i, c_{i1}\rangle) \longrightarrow^{eval} (G', \langle L'_i, c'_{i1}\rangle \parallel P'_0)$ and $P_0 = \langle L'_i, c'_{i1}; c_{i2}\rangle \parallel P'_0$. Therefore we have $(G, P[i \mapsto c_{i1}]) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P'[i \mapsto c'_{i1}])$. By Lemma 49 the set of variables that $\langle L_i, c_{i1}\rangle$ reads [writes] next are a subset of those that $\langle L_i, c_{i1}; c_{i2}\rangle$ reads [writes] next. Therefore by Lemma 51 we have that $(G, P[i \mapsto c_{i1}])$ is well behaved. By Lemma 52 $(H, Q[i \mapsto c_{i1}])$ is the commit closure of $(G, P[i \mapsto c_{i1}])$. Finally by Lemma 49 we know that $\langle L_i, c_{i1}\rangle$ writes X next.

Therefore by induction there exist (H_0, Q_o) and (H', Q') such that $(H, Q[i \mapsto c_{i1}]) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0)$ and $(H_0, Q_0) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q')$, where (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i1}])$. Then by rule EC-SEQSTRUCT it follows that

$$(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0[i \mapsto Q_0.cmd(i); c_{i2}])$$

and by the commit rule

$$(H_0, Q_0[i \mapsto Q_0.cmd(i); c_{i2}]) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q'[i \mapsto Q_0.cmd(i); c_{i2}])$$

Since (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i1}])$ we have that $Q_0.cmd(i) = Q'.cmd(i) = c'_{i1}$, so by Lemma 52 we have that $(H', Q'[i \mapsto Q_0.cmd(i); c_{i2}])$ is the commit closure of (G', P').

holding ℓ **do** c_{i0} Since $\langle L_i, c_i\rangle$ writes X next in P we have that $\ell \in L_i$ and $\langle L_i, c_{i0}\rangle$ writes X next. Therefore we know that c_{i0} is not **skip**, so the step cannot occur with rule EC-HOLDRELEASE. Therefore the step must occur with rule EC-HOLDSTEP. Therefore we have $(G, \langle L_i, c_{i0}\rangle) \longrightarrow^{eval} (G', \langle L'_i, c'_{i0}\rangle \parallel P'_0)$ and $P_0 = \langle L'_i, \textbf{holding } \ell \textbf{ do } c'_{i0}\rangle \parallel P'_0$. Therefore we have

$$(G, P[i \mapsto c_{i0}]) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P'[i \mapsto c'_{i0}])$$

By Lemma 50 the set of variables that $\langle L_i, c_{i0}\rangle$ reads [writes] next are a subset of those that $\langle L_i, \textbf{holding } \ell \textbf{ do } c_{i0}\rangle$ reads [writes] next. Therefore by Lemma 51 we have that $(G, P[i \mapsto c_{i0}])$ is well behaved. By Lemma 52 $(H, Q[i \mapsto c_{i0}])$ is the commit closure of $(G, P[i \mapsto c_{i0}])$. Finally by Lemma 50 we know that $\langle L_i, c_{i0}\rangle$ accesses X next.

Therefore by induction there exist (H_0, Q_o) and (H', Q') such that $(H, Q[i \mapsto c_{i0}]) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0)$ and $(H_0, Q_0) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q')$, where (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i0}])$. Since $\ell \in L_i$ also $\ell \in \hat{L}_i$, so by rule EC-HOLDSTEP it follows that

$$(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H_0, Q_0[i \mapsto \textbf{holding } \ell \textbf{ do } Q_0.cmd(i)])$$

and by the commit rule

$$(H_0, Q_0[i \mapsto \textbf{holding } \ell \textbf{ do } Q_0.cmd(i)]) \Longrightarrow^{\mathsf{sc}}_{commit(i)} (H', Q'[i \mapsto \textbf{holding } \ell \textbf{ do } Q_0.cmd(i)])$$

Since (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i0}])$, we have $Q_0.cmd(i) = Q'.cmd(i) = c'_{i0}$, so by Lemma 52 we have that $(H', Q'[i \mapsto \textbf{holding } \ell \textbf{ do } Q_0.cmd(i)])$ is the commit closure of (G', P').

3. We proceed by induction on the derivation of $(G, t_i) \longrightarrow^{op} (G', P_0)$. Case analysis of the rule used to take this step:

EC-LOAD Then c_i has the form $x := X$ and $(G.mem; L_i.wb)[X] \Downarrow n$ and $P_0 = \langle L_i[x \mapsto n], \textbf{skip}\rangle$ and $G' = G$. Since c_i accesses X next, by Lemma 48 we have $H.mem(X) = n$, and since $\hat{L}_i$ has an empty write buffer also $(H.mem; \hat{L}_i.wb)[X] \Downarrow n$. Then by EC-LOAD we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i[x \mapsto n], \textbf{skip}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$.

Now we argue that (H', Q') is the commit closure of (G', P'). Since $\hat{L}_i[x \mapsto n]$ is the "cleared" version of $L_i[x \mapsto n]$, condition 1 holds for thread i, and it continues to hold for all other threads, which are unchanged. Condition 2 continues to hold since locks are not changed. Since $G' = G$ and $H' = H$ and no write buffers are modified, conditions 3 and 4 continue to hold.

EC-EVALEXP Then c_i has the form $x := a$ and $L_i[a] \Downarrow n$ and $P_0 = \langle L_i[x \mapsto n], \mathbf{skip}\rangle$ and $G' = G$. By definition of $\hat{L}_i$ also $\hat{L}_i[a] \Downarrow n$, so by EC-EVALEXP we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i[x \mapsto n], \mathbf{skip}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$. Finally, we can argue that (H', Q') is the commit closure of (G', P') as in the case for EC-LOAD above.

EC-SYNCACQUIRE Then c_i has the form $\mathbf{sync}\ \ell\ \mathbf{do}\ c_{i0}$ and $\ell \in G$ and $L_i.wb = nil$ and $G' = G \setminus \{\ell\}$ and $P_0 = \langle L_i \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c_{i0}\rangle$. Since (H, Q) is the commit closure of (G, P), also $\ell \in H$, and by definition $\hat{L}_i.wb = nil$. Therefore by EC-SYNCACQUIRE we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H', Q_0)$, where $H' = H \setminus \{\ell\}$ and $Q_0 = \langle \hat{L}_i \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c_{i0}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $Q' = Q[i \mapsto Q_0]$.

Now we argue that (H', Q') is the commit closure of (G', P'). Since $\hat{L}_i \cup \{\ell\}$ is also the "cleared" version of $L_i \cup \{\ell\}$, condition 1 holds for thread i, and it continues to hold for all other threads, which are unchanged. Condition 2 continues to hold since both G and H get ℓ removed from their locksets. Since $G'.mem = G.mem$ and $H'.mem = H.mem$ and no write buffers are modified, conditions 3 and 4 continue to hold.

EC-SYNCREENTER Then c_i has the form $\mathbf{sync}\ \ell\ \mathbf{do}\ c_{i0}$ and $\ell \in L_i$ and $G' = G$ and $P_0 = \langle L_i, c_{i0}\rangle$. Therefore also $\ell \in \hat{L}_i$, so by EC-SYNCREENTER we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i, \mathbf{fence}; (c_{i0}; \mathbf{fence})\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto \mathbf{fence}; (c_{i0}; \mathbf{fence})]$. Then by Lemma 52 we have that (H', Q') is the commit closure of (G', P').

EC-HOLDSTEP Then c_i has the form $\mathbf{holding}\ \ell\ \mathbf{do}\ c_{i0}$ and $\ell \in L_i$ and $(G, \langle L_i, c_{i0}\rangle) \longrightarrow^{eval} (G', \langle L'_i, c'_{i0}\rangle \parallel P'_0)$ and $P_0 = \langle L'_i, \mathbf{holding}\ \ell\ \mathbf{do}\ c'_{i0}\rangle \parallel P'_0$. Therefore we have $(G, P[i \mapsto c_{i0}]) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P'[i \mapsto c'_{i0}])$. By Lemma 50 the set of variables that $\langle L_i, c_{i0}\rangle$ reads [writes] next are a subset of those that $\langle L_i, \mathbf{holding}\ \ell\ \mathbf{do}\ c_{i0}\rangle$ reads [writes] next. Therefore by Lemma 51 we have that $(G, P[i \mapsto c_{i0}])$ is well behaved. Then by Lemma 52 $(H, Q[i \mapsto c_{i0}])$ is the commit closure of $(G, P[i \mapsto c_{i0}])$. Finally by Lemma 50 we know that $\langle L_i, c_{i0}\rangle$ does not write any variables next.

Therefore by induction there exist (H', Q') such that $(H, Q[i \mapsto c_{i0}]) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i0}])$. Since $\ell \in L_i$ also $\ell \in \hat{L}_i$. Then by rule EC-HOLDSTEP it follows that

$$(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q'[i \mapsto \mathbf{holding}\ \ell\ \mathbf{do}\ Q'.cmd(i)])$$

Since (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i0}])$ we have that $Q'.cmd(i) = c'_{i0}$, so by Lemma 52 we have that $(H', Q'[i \mapsto \mathbf{holding}\ \ell\ \mathbf{do}\ Q'.cmd(i)])$ is the commit closure of (G', P').

EC-HOLDRELEASE Then c_i has the form $\mathbf{holding}\ \ell\ \mathbf{do}\ \mathbf{skip}$ and $\ell \in L_i$ and $L_i.wb = nil$ and $G' = G \cup \{\ell\}$ and $P_0 = \langle L_i \setminus \{\ell\}, \mathbf{skip}\rangle$. Then by definition $\ell \in \hat{L}_i$ and $\hat{L}_i.wb = nil$. Therefore by EC-HOLDRELEASE we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H', Q_0)$, where $H' = H \cup \{\ell\}$ and $Q_0 = \langle \hat{L}_i \setminus \{\ell\}, \mathbf{skip}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $Q' = Q[i \mapsto Q_0]$.

Now we argue that (H', Q') is the commit closure of (G', P'). Since $\hat{L}_i \setminus \{\ell\}$ is also the "cleared" version of $L_i \setminus \{\ell\}$, condition 1 holds for thread i, and it continues to hold for all other threads, which are unchanged. Condition 2 continues to hold since both G and H get ℓ added to their locksets. Since $G'.mem = G.mem$ and $H'.mem = H.mem$ and no write buffers are modified, conditions 3 and 4 continue to hold.

EC-FENCE Then c_i has the form $\mathbf{fence}$ and $L_i.wb = nil$ and $G' = G$ and $P_0 = \langle L_i, \mathbf{skip}\rangle$. By definition $\hat{L}_i.wb = nil$. Therefore by EC-FENCE we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i, \mathbf{skip}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$. Then by Lemma 52 we have that (H', Q') is the commit closure of (G', P').

EC-FORK Then c_i has the form $\mathbf{fork}\ c_{i0}$ and $L_i.wb = nil$ and $G' = G$ and $P_0 = \langle L_i, \mathbf{skip}\rangle \parallel \langle L_\oslash, c_{i0}\rangle$. By definition $\hat{L}_i.wb = nil$. Therefore by EC-FORK we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i, \mathbf{skip}\rangle \parallel \langle L_\oslash, c_{i0}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$.

Now we argue that (H', Q') is the commit closure of (G', P'). Since all local states are unchanged and $L_\oslash$ is its own "cleared" version by definition, condition 1 continues to hold. Condition 2 continues to hold since $G = G'$ and $H = H'$. Finally, since $G'.mem = G.mem$ and $H'.mem = H.mem$, no write buffers are modified, and $L_\oslash$ has an empty write buffer, conditions 3 and 4 continue to hold.

EC-SEQSTRUCT Then c_i has the form $c_{i1}; c_{i2}$ and $(G, \langle L_i, c_{i1}\rangle) \longrightarrow^{eval} (G', \langle L'_i, c'_{i1}\rangle \parallel P'_0)$ and $P_0 = \langle L'_i, c'_{i1}; c_{i2}\rangle \parallel P'_0$. Therefore we have $(G, P[i \mapsto c_{i1}]) \Longrightarrow^{\mathsf{tso}}_{op(i)} (G', P'[i \mapsto c'_{i1}])$. By Lemma 49 the set of variables that $\langle L_i, c_{i1}\rangle$ reads [writes] next are a subset of those that $\langle L_i, c_{i1}; c_{i2}\rangle$ reads [writes] next. Therefore by Lemma 51 we have that $(G, P[i \mapsto c_{i1}])$ is well behaved. By Lemma 52 $(H, Q[i \mapsto c_{i1}])$ is the commit closure of $(G, P[i \mapsto c_{i1}])$. Finally by Lemma 49 we know that $\langle L_i, c_{i1}\rangle$ does not write any variables next.

Therefore by induction there exist (H', Q') such that $(H, Q[i \mapsto c_{i1}]) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i1}])$. Then by rule EC-SEQSTRUCT it follows that

$$(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q'[i \mapsto Q'.cmd(i); c_{i2}])$$

Since (H', Q') is the commit closure of $(G', P'[i \mapsto c'_{i1}])$ we have that $Q'.cmd(i) = c'_{i1}$, so by Lemma 52 we have that $(H', Q'[i \mapsto Q'.cmd(i); c_{i2}])$ is the commit closure of (G', P').

EC-SEQSKIP Then c_i has the form **skip**; c_{i0} and $G' = G$ and $P_0 = \langle L_i, c_{i0}\rangle$. Therefore by EC-SEQSKIP we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i, c_{i0}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$. Then by Lemma 52 we have that (H', Q') is the commit closure of (G', P').

EC-IFTRUE Then c_i has the form **if** b **do** c_{i1} **else** c_{i2} and $L_i[b] \Downarrow$ **true** and $G' = G$ and $P_0 = \langle L_i, c_{i1}\rangle$. By definition of $\hat{L}_i$ also $\hat{L}_i[b] \Downarrow$ **true**, so by EC-IFTRUE we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \langle \hat{L}_i, c_{i1}\rangle$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto Q_0]$. Then by Lemma 52 we have that (H', Q') is the commit closure of (G', P').

EC-IFFALSE, EC-WHILETRUE, and EC-WHILEFALSE Follows from the same argument as for rule EC-IFTRUE above.

EC-REAP Then c_i has the form **skip** and $L_i.wb = nil$ and $L_i.locks = \emptyset$ and $G' = G$ and $P_0 = \circ$. Therefore by EC-REAP we have $(H, \langle \hat{L}_i, c_i\rangle) \longrightarrow^{op} (H, Q_0)$, where $Q_0 = \circ$. Therefore $(H, Q) \Longrightarrow^{\mathsf{sc}}_{op(i)} (H', Q')$, where $H' = H$ and $Q' = Q[i \mapsto \circ]$.

Now we argue that (H', Q') is the commit closure of (G', P'). Condition 1 continues to hold for all threads other than i, which are unchanged, and thread i is removed from both P and Q. Condition 2 continues to hold since locks are not changed in G or H. For condition 3, since $L_i.wb = nil$ we know that the set of variables that do not appear in any write buffer in P is equivalent to the set that do not appear in any write buffer in P'. Then since $G' = G$ and $H' = H$ condition 3 continues to hold. For condition 4, again since $L_i.wb = nil$ we know that a variable X appears in the write buffer for some thread j in P if and only if it appears in that thread's write buffer in P', and furthermore $j \neq i$. Then again since $G' = G$ and $H' = H$ and no write buffers are modified, condition 4 continues to hold.

□

Lemma 56. *Suppose* $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{tso}} (G_1, P_1) \Longrightarrow^{\mathsf{tso}} \ldots \Longrightarrow^{\mathsf{tso}} (G_n, P_n) \Longrightarrow^{\mathsf{tso}} (G', P')$ *for some* $n \geq 0$ *and* (G_k, P_k) *is well behaved for each* $1 \leq k \leq n$ *and* (G', P') *is not well behaved. Then* $(G, \langle L_\oslash, c\rangle)$ *is not race-free.*

Proof. By Definition 19 we have that $(G, \langle L_\oslash, c\rangle)$ is well behaved, and by the definition of $L_\oslash$ as well as Definition 20 also $(G, \langle L_\oslash, c\rangle)$ is its own commit closure. Therefore by Lemma 55 we have $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{sc}*} (H_1, Q_1) \Longrightarrow^{\mathsf{sc}*} \ldots \Longrightarrow^{\mathsf{sc}*} (H_n, Q_n) \Longrightarrow^{\mathsf{sc}*} (H', Q')$ where for each $1 \leq k \leq n$ we have that (H_k, Q_k) is the commit closure of (G_k, P_k) and (H', Q') is the commit closure of (G', P').

Since (G', P') is not well behaved, by Definition 19 there are four possibilities. First suppose P' exhibits a race. Then by Lemma 53 so does Q' and we have shown that $(G, \langle L_\oslash, c\rangle)$ is not race-free. Second suppose (G', P') is well locked. But since (G_n, P_n) is well behaved it is also well locked, so by Lemma 42 so is (G', P') and we have a contradiction. Third suppose there are distinct threads in P' that both contain a write to some variable X in their write buffers. Since (G_n, P_n) is well behaved we know that this is not also true in P_n. Then there must exist distinct threads ι and ι' in P_n such that there is a write to X in $P_n.ls(\iota).wb$ and $P_n.cmd(\iota')$ has the form $\mathcal{C}[X := x]$. Then by Lemma 54 thread ι' in P_n writes X next, so it also accesses X next. But then (G_n, P_n) is not well behaved, and we have a contradiction.

Finally suppose there are distinct threads ι and ι' in P' such that there is a write to X in $P'.ls(\iota).wb$ and ι' accesses X next in P'. I claim that the step from (G_n, P_n) to (G', P') must execute an *eval* step on thread ι', and hence also the step(s) from (H_n, Q_n) to (H', Q') must execute thread ι' by the construction in Lemma 55. If the step from (G_n, P_n) to (G', P') executes a thread other than ι or ι' or executes a *commit* step on ι', then there is a write to X in $P_n.ls(\iota).wb$ and ι' accesses X next in P_n, contradicting the well-behavedness of (G_n, P_n). If the step executes thread ι, then again ι' accesses X next in P_n, and either there is a write to X in $P_n.ls(\iota).wb$ or ι writes X next in P_n. Either way, the well-behavedness of (G_n, P_n) is contradicted.

Let k be the greatest index such that $P_k.cmd(\iota)$ has the form $\mathcal{C}[X := x]$, the step from (G_k, P_k) in our execution sequence is on thread ι, and for all indices m such that $k < m \leq n$ there is a write to X in $P_m.ls(\iota).wb$. It's clear one such process soup exists, in order to take the step that puts X in the write buffer of thread ι in P'. (Further, the latest such process soup cannot be the very first one, $\langle L_\oslash, c\rangle$, since there is only one thread initially, and forking another thread requires that the write buffer be empty.) Since (H_k, Q_k) is the commit closure of (G_k, P_k) also $Q_k.cmd(\iota)$ has the form $\mathcal{C}[X := x]$.

Consider the sequence of $\Longrightarrow^{\mathsf{sc}}$ steps from (H_k, Q_k) to (H', Q'), and let's rename them as follows: $(H'_1, Q'_1) \Longrightarrow^{\mathsf{sc}}_{op_1(i_1)} (H'_2, Q'_2) \Longrightarrow^{\mathsf{sc}} \ldots \Longrightarrow^{\mathsf{sc}} (H'_m, Q'_m) \Longrightarrow^{\mathsf{sc}}_{op_m(i_m)} (H', Q')$. Call the associated sequence of actions $\overline{\alpha}$, and let $\overline{\alpha_0}$ be identical to $\overline{\alpha}$ but with each action on thread ι removed. I claim that $\overline{\alpha_0}$ is also a valid SC execution sequence from (H'_1, Q'_1) and furthermore that this sequence ends in a state (H'', Q'') such that $Q'/\iota = Q''/\iota$. If we can argue this, then we have shown that $(G, \langle L_\oslash, c\rangle)$ is not race-free, since in the final process soup Q'' of our execution sequence $\overline{\alpha_0}$ we have that ι writes X next (because that thread is unchanged from Q_k) and ι' accesses X next (since ι' accesses X next in Q' and $Q'/\iota = Q''/\iota$).

Consider a step on thread ι from some state (H, Q) in our execution sequence $\overline{\alpha}$. In order for this step to affect the behavior of another thread in a later step of the sequence, this step must modify the global store. There are only three rules that directly modify the store:

EC-SYNCACQUIRE So $Q.cmd(\iota)$ has the form $\mathcal{C}[\textbf{sync}\ \ell\ \textbf{do}\ c_0]$ and $\ell \in H$ and $Q.ls(\iota).wb = nil$. But then by construction of our SC execution in Lemma 55, we know that there is some $j > k$ such that $P_j.cmd(\iota)$ has the form $\mathcal{C}[\textbf{sync}\ \ell\ \textbf{do}\ c_0]$ and $\ell \in G_j$. Further, by well-behavedness we know that $\ell \notin P_j.ls(\iota)$. Therefore, the step from (G_j, P_j) must also use EC-SYNCACQUIRE. But then $P_j.ls(\iota).wb = nil$, which contradicts the fact that a write to X is in $P_j.ls(\iota).wb$.

EC-HOLDRELEASE Similar to the above case.

commit By construction of our SC execution in Lemma 55, a *commit* step only happens on thread ι directly after the associated write is put into ι's write buffer. Further, there is a corresponding write to the write buffer of ι in the original TSO execution. So there is some $j \geq k$ such that $P_j.cmd(\iota)$ has the form $\mathcal{C}[Y := y]$ and (G_j, P_j) executes thread ι, thereby adding Y to ι's write buffer. By our choice of (G_k, P_k) we know that the step from (G_k, P_k) adds a write to X to the write buffer of ι and that this write is never committed in the rest of our execution sequence. Since the write buffer is emptied in FIFO order, this means that the write to Y at (G_j, P_j) is also never committed (note that it is possible that $j = k$ and hence $Y = X$). Therefore by well-behavedness, in each (G_m, P_m) for $j \leq m \leq n$ it is the case that no thread other than ι accesses Y next. Therefore, the same is true for each configuration following (H_j, Q_j) in our SC execution sequence $\overline{\alpha}$. Therefore, we can omit this write of Y without affecting any other thread's behavior in the rest of the execution sequence.

□

Theorem 2. *If* $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{tso}*} (G', \mathfrak{o})$ *and* $(G, \langle L_\oslash, c\rangle)$ *is race-free, then* $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{sc}*} (G', \mathfrak{o})$.

Proof. Since $(G, \langle L_\oslash, c\rangle)$ is race-free, by Lemma 56 each intermediate configuration in execution $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{tso}*} (G', \mathfrak{o})$ is well behaved. By Definition 20, $(G, \langle L_\oslash, c\rangle)$ is its own commit closure. Therefore by Lemma 55 we have $(G, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{sc}*} (H', Q')$ where (H', Q') is the commit closure of $(G', \mathfrak{o})$. But then by Definition 20, $Q' = \mathfrak{o}$ so also $H' = G'$. □

5 Typing for Write Buffers

Write buffers are not as cumbersome as in the proofs about the previous type system, and we won't need to call them out specially in contexts. Notation $\mathcal{E}[t]$ denotes $\mathcal{E}[nil \mid t]$.

$\boxed{pc; wt; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt}$

$$\frac{pc \sqcup \Gamma(Y) \sqsubseteq \Gamma(x)}{pc; wt; \Gamma \vdash^{\mathrm{wb}} x := Y \Rightarrow wt}\ \textsc{wb-Load} \qquad \frac{pc \sqcup \Gamma(y) \sqsubseteq \Gamma(X)}{pc; wt; \Gamma \vdash^{\mathrm{wb}} X := y \Rightarrow wt \sqcap \Gamma(X)}\ \textsc{wb-Store}$$

$$\frac{\Gamma \vdash a : \tau \qquad pc \sqcup \tau \sqsubseteq \Gamma(x)}{pc; wt; \Gamma \vdash^{\mathrm{wb}} x := a \Rightarrow wt}\ \textsc{wb-Eval} \qquad \frac{pc \sqsubseteq \Gamma(\ell) \qquad pc \sqsubseteq wt \qquad \Gamma(\ell); high; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt'}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{sync}\ \ell\ \textbf{do}\ c \Rightarrow high}\ \textsc{wb-Sync}$$

$$\frac{pc \sqsubseteq \Gamma(\ell) \qquad \Gamma(\ell) \sqsubseteq wt \qquad \Gamma(\ell); wt; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt'}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{holding}\ \ell\ \textbf{do}\ c \Rightarrow high}\ \textsc{wb-Hold} \qquad \frac{pc \sqsubseteq wt}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{fence} \Rightarrow high}\ \textsc{wb-Fence}$$

$$\frac{pc; high; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt' \qquad pc \sqsubseteq wt}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{fork}\ c \Rightarrow high}\ \textsc{wb-Fork}$$

$$\frac{pc; wt; \Gamma \vdash^{\mathrm{wb}} c_1 \Rightarrow wt_1 \qquad pc; wt_1; \Gamma \vdash^{\mathrm{wb}} c_2 \Rightarrow wt_2}{pc; wt; \Gamma \vdash^{\mathrm{wb}} c_1;\ c_2 \Rightarrow wt_2}\ \textsc{wb-Seq}$$

$$\frac{\Gamma \vdash b : \tau \qquad pc \sqcup \tau; wt; \Gamma \vdash^{\mathrm{wb}} c_1 \Rightarrow wt_1 \qquad pc \sqcup \tau; wt; \Gamma \vdash^{\mathrm{wb}} c_2 \Rightarrow wt_2}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{if}\ b\ \textbf{do}\ c_1\ \textbf{else}\ c_2 \Rightarrow wt_1 \sqcap wt_2}\ \textsc{wb-If}$$

$$\frac{\Gamma \vdash b : low \qquad pc; wt \sqcap wt'; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt'}{low; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{while}\ b\ \textbf{do}\ c \Rightarrow wt \sqcap wt'}\ \textsc{wb-While} \qquad \frac{}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \textbf{skip} \Rightarrow wt}\ \textsc{wb-Skip}$$

$\boxed{pc; \Gamma \vdash^{\mathrm{wb}} \lambda}$

$$\frac{}{pc; \Gamma \vdash^{\mathrm{wb}} \emptyset} \qquad \frac{pc \sqsubseteq \Gamma(\ell) \qquad pc; \Gamma \vdash^{\mathrm{wb}} \lambda}{pc; \Gamma \vdash^{\mathrm{wb}} \lambda \cup \{\ell\}}$$

$\boxed{wt; \Gamma \vdash^{\mathrm{wb}} W}$

$$\frac{}{wt; \Gamma \vdash^{\mathrm{wb}} nil} \qquad \frac{wt \sqsubseteq \Gamma(X) \qquad wt; \Gamma \vdash^{\mathrm{wb}} W}{wt; \Gamma \vdash^{\mathrm{wb}} (X := i){::}W}$$

$\boxed{pc; wt; \Gamma \vdash^{\mathrm{wb}} t}$

$$\frac{pc; \Gamma \vdash^{\mathrm{wb}} \lambda \qquad wt; \Gamma \vdash^{\mathrm{wb}} W \qquad pc; wt; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow wt'}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \langle (M, \lambda, W), c\rangle_\iota}$$

$\boxed{\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P}$

$$\frac{}{\cdot; \cdot; \Gamma \vdash^{\mathrm{wb}} \mathfrak{o}} \qquad \frac{pc; wt; \Gamma \vdash^{\mathrm{wb}} t \qquad \overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P}{pc, \overline{pc}; wt, \overline{wt}; \Gamma \vdash^{\mathrm{wb}} t \parallel P}$$

Lemma 57 (WB Admissibility of subtyping)**.** *Suppose $pc_1 \sqsubseteq pc_2$ and $wt_1 \sqsubseteq wt_2$. Then the following hold.*

(i) $pc_2; \Gamma \vdash^{\text{wb}} \lambda$ *implies* $pc_1; \Gamma \vdash^{\text{wb}} \lambda$.

(ii) $wt_2; \Gamma \vdash^{\text{wb}} W$ *implies* $wt_1; \Gamma \vdash^{\text{wb}} W$.

(iii) $pc_2; wt_1; \Gamma \vdash^{\text{wb}} c \Rightarrow ut$ *implies* $pc_1; wt_2; \Gamma \vdash^{\text{wb}} c \Rightarrow vt$ *for some* vt *where* $ut \sqsubseteq vt$.

(iv) $pc_2; wt; \Gamma \vdash^{\text{wb}} t$ *implies* $pc_1; wt; \Gamma \vdash^{\text{wb}} t$.

Proof. Statements (i)–(iii) follow from easy inductions on the typing derivations. The only mildly interesting cases are for WB-SEQ, WB-IF, and WB-WHILE, in which we need to reason with the induction hypothesis's $ut \sqsubseteq vt$ constraint. Statement (iv) follows from (i)–(iii). □

Lemma 58. *Suppose* $pc; wt; \Gamma \vdash^{\text{wb}} c \Rightarrow vt$. *Then* $pc \sqcap wt \sqsubseteq vt$.

Proof. by an easy induction. □

Lemma 59. *From* $pc \sqsubseteq \Gamma(\ell)$ *and* $pc; \Gamma \vdash^{\text{wb}} \lambda$, *it follows that* $pc; \Gamma \vdash^{\text{wb}} \lambda \cup \{\ell\}$.

Proof. Immediate. □

Lemma 60. *Suppose* $pc; \Gamma \vdash^{\text{wb}} \lambda$. *It follows that* $pc; \Gamma \vdash^{\text{wb}} \lambda \setminus \{\ell\}$.

Proof. by trivial induction. □

Lemma 61 (WB buffer append typing)**.** *If* $wt; \Gamma \vdash^{\text{wb}} W$ *then* $wt \sqcap \Gamma(X); \Gamma \vdash^{\text{wb}} W +\!\!+ (X := i)$.

Proof. by induction. □

Lemma 62 (WB Commit Step Preservation)**.** *Suppose* $pc; wt; \Gamma \vdash^{\text{wb}} t$ *and* $(G, t) \longrightarrow^{commit} (G', P')$. *Then* $\overline{pc}; \overline{wt}; \Gamma \vdash^{\text{wb}} P'$ *where* $pc \sqsubseteq \overline{pc}$ *and* $wt \sqsubseteq \overline{wt}$.

Proof. Let $\langle L, c\rangle = t$. Then $P' = \langle L_0, c\rangle \parallel \mathsf{o}$ where $L = (X := i) +\!\!+ L_0$ for some X, i, and L_0. We must show that $wt; \Gamma \vdash^{\text{wb}} L_0.wb$, which follows from inverting the typing derivation. □

Lemma 63 (WB Eval Step Preservation)**.** *Suppose* $pc; wt; \Gamma \vdash^{\text{wb}} t$ *and* $(G, t) \longrightarrow^{eval} (G', P')$. *Then* $\overline{pc}; \overline{wt}; \Gamma \vdash^{\text{wb}} P'$ *where* $pc \sqsubseteq \overline{pc}$ *and* $wt \sqcap pc \sqsubseteq \overline{wt}$.

Proof. Let $\langle L, c\rangle = t$ and observe that $pc; wt; \Gamma \vdash^{\text{wb}} c \Rightarrow ut$. Strengthen the induction hypothesis as follows: If $P' = t' \parallel P_0'$, it is the case that $vt; \Gamma \vdash^{\text{wb}} t'.wb$ and $pc; vt; \Gamma \vdash^{\text{wb}} t'.cmd \Rightarrow ut'$ where $wt \sqcap pc \sqsubseteq vt$ and $ut \sqsubseteq ut'$. Proceed by induction on the $\longrightarrow^{eval}$ step derivation.

EC-STORE: Here $c = (X := y)$ and $P' = \langle L +\!\!+ (X := i), \textbf{skip}\rangle \parallel \mathsf{o}$ for some i. It suffices to find vt such that (i) $wt \sqcap pc \sqsubseteq vt$, (ii) $vt; \Gamma \vdash^{\text{wb}} L.wb +\!\!+ (X := i)$, and (iii) $pc; vt; \Gamma \vdash^{\text{wb}} \textbf{skip} \Rightarrow ut$. Taking $vt = ut$ latter is immediate. Inverting the typing relation shows $pc \sqsubseteq pc \sqcup \Gamma(y) \sqsubseteq \Gamma(X)$ and $vt = ut = wt \sqcap \Gamma(X) \sqsupseteq wt \sqcap pc$, satisfying (i). Finally (ii) follows from Lemma 61.

EC-LOAD: Here $c = (x := Y)$ and $P' = \langle L[X \mapsto i], \textbf{skip}\rangle \parallel \mathsf{o}$. Inverting the typing relation shows $ut = wt$. Taking $vt = wt \sqsupseteq wt \sqcup pc$, it suffices to show $vt; \Gamma \vdash^{\text{wb}} L.wb$, which follows from inverting the typing relation and $pc; vt; \Gamma \vdash^{\text{wb}} L.wb \Rightarrow ut$, which is immediate.

EC-EVALEXP: Similar to the EC-LOAD case.

EC-SYNCACQUIRE: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$ and $P' = \langle L \cup \{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ c_0)\rangle \parallel \mathsf{o}$ and $L.wb = nil$. Inverting the typing relation gives $ut = high$ and $pc \sqsubseteq \Gamma(\ell)$ and $\Gamma(\ell); high; \Gamma \vdash^{\text{wb}} c_0 \Rightarrow ut_0$. Because L's lock set changed, we must show that $pc; \Gamma \vdash^{\text{wb}} L.locks \cup \{\ell\}$, which is immediate from Lemma 59. Let $vt = high$. Conclude by constructing derivations of $high; \Gamma \vdash^{\text{wb}} L.wb$ and $pc; high; \Gamma \vdash^{\text{wb}} \textbf{holding}\ \ell\ \textbf{do}\ c_0 \Rightarrow high$.

EC-SYNCREENTER: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$ and $P' = \langle L, \textbf{fence}; (c_0; \textbf{fence})\rangle \parallel \mathsf{o}$. Inverting the typing relation shows $ut = high$ and $pc \sqsubseteq wt$. Inversion also gives $\Gamma(\ell); high; \Gamma \vdash^{\text{wb}} c_0 \Rightarrow wt_0$ and $pc \sqsubseteq \Gamma(\ell)$ so that Lemma 57 yields a derivation showing $pc; high; \Gamma \vdash^{\mathrm{wb}} c_0 \Rightarrow wt_0'$ with $wt_0 \sqsubseteq wt_0'$. By Lemma 58, $pc \sqsubseteq wt_0'$. Taking $vt = wt$, it remains to construct the following derivation:

$$\dfrac{\dfrac{pc \sqsubseteq wt}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \mathbf{fence} \Rightarrow high} \qquad \dfrac{\dfrac{\vdots}{pc; high; \Gamma \vdash^{\mathrm{wb}} c_0 \Rightarrow wt_0'} \qquad \dfrac{pc \sqsubseteq wt_0'}{pc; wt_0'; \Gamma \vdash^{\mathrm{wb}} \mathbf{fence} \Rightarrow high}}{pc; high; \Gamma \vdash^{\mathrm{wb}} c_0;\, \mathbf{fence} \Rightarrow high}}{pc; wt; \Gamma \vdash^{\mathrm{wb}} \mathbf{fence};\, (c_0;\, \mathbf{fence}) \Rightarrow high}$$

EC-HOLDSTEP: Here $c = \mathbf{holding}\ \ell\ \mathbf{do}\ c_0$ and $P' = \langle L', \mathbf{holding}\ \ell\ \mathbf{do}\ c_0' \rangle \parallel P_0'$ where $(G, \langle L, c_0\rangle) \longrightarrow^{eval} (G', \langle L', c_0'\rangle \parallel P_0')$. Inverting the typing derivation gives $\Gamma(\ell); wt; \Gamma \vdash^{\mathrm{wb}} c_0 \Rightarrow wt_0$ and $pc \sqsubseteq \Gamma(\ell)$. Applying the induction hypothesis yields $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P'$ where $pc \sqsubseteq \Gamma(\ell) \sqsubseteq \overline{pc}$ and $pc \sqcap wt \sqsubseteq \Gamma(\ell) \sqcap wt \sqsubseteq \overline{wt}$. Thus P_0' is typed appropriately, and it remains to shows that $\langle L', \mathbf{holding}\ \ell\ \mathbf{do}\ c_0'\rangle$ satisfies all typing requirements. Also by the induction hypothesis $\Gamma(\ell); vt; \Gamma \vdash^{\mathrm{wb}} c_0' \Rightarrow wt_0'$ and $wt \sqcap \Gamma(\ell) \sqsubseteq vt$. It remains to show that $pc; vt; \Gamma \vdash^{\mathrm{wb}} \mathbf{holding}\ \ell\ \mathbf{do}\ c_0' \Rightarrow high$, which follows from rule WB-HOLD and using fact $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P'$ to establish $pc; \Gamma \vdash^{\mathrm{wb}} L'.locks$.

EC-HOLDRELEASE: Here $c = \mathbf{holding}\ \ell\ \mathbf{do\ skip}$ and $P' = \langle L \setminus \{\ell\}, \mathbf{skip}\rangle \parallel \mathfrak{o}$ with $L.wb = nil$. Inverting the typing relation shows $ut = high$. Let $vt = high$. Proceed as in the EC-LOAD step, noting that Lemma 60 gives $pc; \Gamma \vdash^{\mathrm{wb}} L.locks \setminus \{\ell\}$ and $high; \Gamma \vdash^{\mathrm{wb}} L.wb$ is immediate.

EC-FENCE: Similar to, but simpler than EC-HOLDRELEASE.

EC-FORK: Here $c = \mathbf{fork}\ c_0$ and $P' = \langle L, \mathbf{skip}\rangle \parallel \langle L_\oslash, c_0\rangle \parallel \mathfrak{o}$ and $L.wb = nil$.

First we show that $pc; high; \Gamma \vdash^{\mathrm{wb}} \langle L_\oslash, c_0\rangle \parallel \mathfrak{o}$. Inverting the typing relation gives $pc; high; \Gamma \vdash^{\mathrm{wb}} c_0 \Rightarrow wt_0$. Finish building easy derivations of $pc; \Gamma \vdash^{\mathrm{wb}} L_\oslash.locks$ and $high; \Gamma \vdash^{\mathrm{wb}} L_\oslash.wb$.

Second we let $vt = ut = high$ and show $pc; high; \Gamma \vdash^{\mathrm{wb}} \mathbf{skip} \Rightarrow high$, and $high; \Gamma \vdash^{\mathrm{wb}} L.wb$ both of which are immediate.

EC-SEQSTRUCT: Here $c = c_1;\, c_2$ and $P' = \langle L', c_1'\rangle \parallel P_0'$ where $(G, \langle L, c_1\rangle) \longrightarrow^{eval} (G', \langle L', c_1'\rangle \parallel P_0')$. Inverting the typing derivation gives $pc; wt; \Gamma \vdash^{\mathrm{wb}} c_1 \Rightarrow wt_1$ and $pc; wt_1; \Gamma \vdash^{\mathrm{wb}} c_2 \Rightarrow ut$.

The induction hypothesis shows $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P_0'$ where $pc \sqsubseteq \overline{pc}$ and $wt \sqcap pc \sqsubseteq \overline{wt}$.

Conclude by building a derivation of $pc; vt; \Gamma \vdash^{\mathrm{wb}} c_1';\, c_2 \Rightarrow ut'$ using the induction hypothesis to find and appropriate vt where $pc; vt; \Gamma \vdash^{\mathrm{wb}} c_1' \Rightarrow wt_1'$ and $wt_1 \sqsubseteq wt_1'$ and computing ut' as follows. Use Lemma 57 to find $pc; wt_1'; \Gamma \vdash^{\mathrm{wb}} c_2 \Rightarrow ut'$ where $ut \sqsubseteq ut'$, then build a derivation $pc; vt; \Gamma \vdash^{\mathrm{wb}} c_1';\, c_2 \Rightarrow ut$ with rule WB-SEQ.

EC-SEQSKIP: Here $c = \mathbf{skip};\, c_2$ and $P' = \langle L, c_2\rangle \parallel \mathfrak{o}$. Trivial by inverting the typing derivation.

EC-IFTRUE, EC-IFFALSE, EC-WHILETRUE, EC-WHILEFALSE: Immediate, using Lemma 57 as needed.

EC-REAP: Immediate. □

Lemma 64 (WB Preservation)**.** *Suppose $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P$ and $(G, P) \Longrightarrow^{\mathsf{tso}*} (G', P')$. Then there exist $\overline{pc}'$ and $\overline{wt}'$ such that $\overline{pc}'; \overline{wt}'; \Gamma \vdash^{\mathrm{wb}} P'$.*

Proof. By induction on the length of the $\Longrightarrow^{\mathsf{tso}*}$ derivation, using Lemmas 62 or 63. □

Definition 21 ($\sim^{\mathrm{wb}}_{\Gamma}$)**.**

1. *$\lambda_1 \sim^{\mathrm{wb}}_{\Gamma} \lambda_2$ when it is the case that $\Gamma(\ell) = low$ implies that $\ell \in \lambda_1$ iff $\ell \in \lambda_2$.*

2. *$L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$ iff each of the following holds:*

   (a) $L_1.wb \sim_{\Gamma} L_2.wb$

   (b) $L_1.mem \sim_{\Gamma} L_2.mem$

   (c) $L_1.locks \sim^{\mathrm{wb}}_{\Gamma} L_2.locks$

3. *$t_1 \sim^{\mathrm{wb}}_{\Gamma} t_2$ is defined by the following introduction rules.*

(a) $\langle L_1, c\rangle_{\iota_1} \sim^{\text{wb}}_{\Gamma} \langle L_2, c\rangle_{\iota_2}$ *when* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$

(b) $\mathcal{E}[\langle L_1, c_1\rangle_{\iota_1}] \sim^{\text{wb}}_{\Gamma} \mathcal{E}[\langle L_2, c_2\rangle_{\iota_2}]$ *when* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$ *and both* $high; wt; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle_{\iota_1}$ *and* $high; wt; \Gamma \vdash^{\text{wb}} \langle L_2, c_2\rangle_{\iota_2}$ *for some* wt.

4. $G_1 \sim^{\text{wb}}_{\Gamma} G_2$ *iff* $G_1.mem \sim_{\Gamma} G_2.mem$ *and* $G_1.locks \sim^{\text{wb}}_{\Gamma} G_2.locks$.

5. $P_1 \sim^{\text{wb}}_{\Gamma} P_2$ *is defined by the least fixed point of the following implications.*

(a) $\mathfrak{o} \sim^{\text{wb}}_{\Gamma} \mathfrak{o}$, *always*

(b) $t \parallel P_1 \sim^{\text{wb}}_{\Gamma} P_2$ *when* $high; high; \Gamma \vdash^{\text{wb}} t$ *and* $P_1 \sim^{\text{wb}}_{\Gamma} P_2$

(c) $P_1 \sim^{\text{wb}}_{\Gamma} t \parallel P_2$ *when* $high; high; \Gamma \vdash^{\text{wb}} t$ *and* $P_1 \sim^{\text{wb}}_{\Gamma} P_2$

(d) $t_1 \parallel P_1 \sim^{\text{wb}}_{\Gamma} t_2 \parallel P_2$ *when* $t_1 \sim^{\text{wb}}_{\Gamma} t_2$ *and* $P_1 \sim^{\text{wb}}_{\Gamma} P_2$

6. $(G_1, P_1) \sim^{\text{wb}}_{\Gamma} (G_2, P_2)$ *when* $G_1 \sim^{\text{wb}}_{\Gamma} G_2$ *and* $P_1 \sim^{\text{wb}}_{\Gamma} P_2$.

Lemma 65. *Each* $\sim^{\text{wb}}_{\Gamma}$ *relation is an equivalence relation.*

Proof. by inspection. □

Lemma 66. *If* $(G_1, P_1) \sim^{\text{wb}}_{\Gamma} (G_2, P_{22})$ *and* $P_{21} \sim^{\text{wb}}_{\Gamma} \mathfrak{o}$ *then* $(G_1, P_1) \sim^{\text{wb}}_{\Gamma} (G_2, P_{22} \parallel P_{21})$

Lemma 67. *Consider some* G_1, G_2, *and* λ. $G_1 \sim^{\text{wb}}_{\Gamma} G_2$ *iff* $G_1 \cup \lambda \sim^{\text{wb}}_{\Gamma} G_2 \cup \lambda$.

Lemma 68. *Consider some* G_1, G_2, *and* λ. $G_1 \sim^{\text{wb}}_{\Gamma} G_2$ *iff* $G_1 \setminus \lambda \sim^{\text{wb}}_{\Gamma} G_2 \setminus \lambda$.

Lemma 69. *Consider some* L_1, L_2, *and* λ. $L_1 \sim^{\text{wb}}_{\Gamma} L_2$ *iff* $L_1 \cup \lambda \sim^{\text{wb}}_{\Gamma} L_2 \cup \lambda$.

Lemma 70. *Suppose* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$. *Then* $L_1[x \mapsto i] \sim^{\text{wb}}_{\Gamma} L_2[x \mapsto i]$.

Lemma 71. *Suppose* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$ *and* $\Gamma(x) = high$. *Then* $L_1[x \mapsto i_1] \sim^{\text{wb}}_{\Gamma} L_2[x \mapsto i_2]$.

Lemma 72. *Suppose* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$. *Then* $L_1 +\!\!+ (X := i) \sim^{\text{wb}}_{\Gamma} L_2 +\!\!+ (X := i)$.

Lemma 73. *Suppose* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$ *and* $\Gamma(X) = high$. *Then* $L_1 +\!\!+ (X := i) \sim^{\text{wb}}_{\Gamma} L_2 +\!\!+ (X := j)$.

Lemma 74. *Suppose* $L_1 \sim^{\text{wb}}_{\Gamma} L_2$. *Then* $wt; \Gamma \vdash^{\text{wb}} L_1.wb$ *implies* $wt; \Gamma \vdash^{\text{wb}} L_2.wb$, *and* $pc; \Gamma \vdash^{\text{wb}} L_1.locks$ *implies* $pc; \Gamma \vdash^{\text{wb}} L_2.locks$.

Proof. by trivial inductions. □

Lemma 75. *Suppose* $t_1 \sim^{\text{wb}}_{\Gamma} t_2$. *Then* $\mathcal{E}[t_1] \sim^{\text{wb}}_{\Gamma} \mathcal{E}[t_2]$.

Lemma 76. *If* $P_{11} \parallel t_1 \parallel P_{12} \sim^{\text{wb}}_{\Gamma} P_2$ *then* $P_2 = P_{21} \parallel P_2^* \parallel P_{22}$ *where the following hold:*

$$\begin{array}{rcl} P_{21} & \sim^{\text{wb}}_{\Gamma} & P_{11} \\ P_2^* & \sim^{\text{wb}}_{\Gamma} & t_1 \\ P_{22} & \sim^{\text{wb}}_{\Gamma} & P_{12} \\ P_2^* & \in & \{\mathfrak{o}, t_2 \parallel \mathfrak{o}\} \textit{ for some } t_2 \end{array}$$

Proof. By any easy induction on the sum of the lengths of P_1 and P_2. □

Lemma 77. *If* $pc; wt; \Gamma \vdash^{\text{wb}} \mathcal{E}[t]$ *then* $pc; wt; \Gamma \vdash^{\text{wb}} t$.

Proof. Let $\mathcal{E} = (\lambda, \mathcal{C})$. First use induction on the size of λ to show $pc; wt; \Gamma \vdash^{\text{wb}} \mathcal{E}_0[t]$ where $\mathcal{E}_0 = (\emptyset, \mathcal{C})$. Conclude by an easy structural induction on $\mathcal{C}$, using Lemma 57. □

Definition 22 $(\cdot|_{\Gamma,\tau})$**.**

$$\begin{array}{rcl} \lambda|_{\Gamma,\tau} & = & \{\ \ell \mid \ell \in \lambda \textit{ and } \Gamma(\ell) = \tau\ \} \\ L|_{\Gamma,\tau} & = & L.locks|_{\Gamma,\tau} \\ \langle L, c\rangle|_{\Gamma,\tau} & = & L|_{\Gamma,\tau} \\ \\ \mathfrak{o}|_{\Gamma,\tau} & = & \emptyset \\ (t \parallel P)|_{\Gamma,\tau} & = & t|_{\Gamma,\tau} \cup P|_{\Gamma,\tau} \\ \\ G|_{\Gamma,\tau} & = & G.locks|_{\Gamma,\tau} \end{array}$$

Lemma 78. *If $\lambda \subseteq \mathbf{Lock}|_{\Gamma,high}$ and $high; wt; \Gamma \vdash^{\mathrm{wb}} t$ then $high; wt; \Gamma \vdash^{\mathrm{wb}} t \cup \lambda$.*

Proof. Immediate $\square$

Lemma 79. *Suppose that $high; wt; \Gamma \vdash^{\mathrm{wb}} t$. Then $t.cmd.locks \subseteq \mathbf{Lock}|_{\Gamma,high}$.*

Proof. by trivial induction on the typing derivation. $\square$

Lemma 80. *Suppose $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$ and $high; wt; \Gamma \vdash^{\mathrm{wb}} c_1 \Rightarrow ut_1$ and $high; wt; \Gamma \vdash^{\mathrm{wb}} c_2 \Rightarrow ut_2$. Suppose also that $wt; \Gamma \vdash^{\mathrm{wb}} L_1.wb$. Then $\langle L_1, c_1\rangle \sim^{\mathrm{wb}}_{\Gamma} \langle L_2, c_2\rangle$.*

Proof. By the definition of $\sim^{\mathrm{wb}}$ we want to find $\mathcal{E} = (\lambda, \mathcal{C})$, L_{10}, and L_{20}, such that the following hold.

$$\langle L_1, c_1\rangle = \mathcal{E}[\langle L_{10}, c_1\rangle] \qquad \langle L_2, c_2\rangle = \mathcal{E}[\langle L_{20}, c_2\rangle] \qquad high; wt; \Gamma \vdash^{\mathrm{wb}} \langle L_{10}, c_1\rangle \qquad high; wt; \Gamma \vdash^{\mathrm{wb}} \langle L_{20}, c_2\rangle$$

Let $\mathcal{C} = [\cdot]$ and $\lambda = L_1|_{\Gamma,low}$. Let also $L_{10} = L_1 \setminus \lambda$ and $L_{20} = L_2 \setminus \lambda$. Clearly $\langle L_1, c_1\rangle = \mathcal{E}[\langle L_{10}, c_1\rangle]$. Because $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$ it's also true that $\lambda = L_2|_{\Gamma,low}$, so $\langle L_2, c_2\rangle = \mathcal{E}[\langle L_{20}, c_2\rangle]$. We demonstrate $high; wt; \Gamma \vdash^{\mathrm{wb}} \langle L_{10}, c_1\rangle$ by using the definition of L_{10} and the premises to show that $high; \Gamma \vdash^{\mathrm{wb}} L_{10}.locks$ and $wt; \Gamma \vdash^{\mathrm{wb}} L_{10}.wb$. It remains to show $high; wt; \Gamma \vdash^{\mathrm{wb}} \langle L_{20}, c_2\rangle$, which works as above, using Lemma 74 to help establish $wt; \Gamma \vdash^{\mathrm{wb}} L_{20}.wb$. $\square$

Lemma 81 (WB Eval step local confinement)**.** *Suppose $t.cmd \neq \mathbf{skip}$ and both $high; wt; \Gamma \vdash^{\mathrm{wb}} t$ and $(G, \mathcal{E}[t]) \longrightarrow^{eval} (G', P')$. Then $(G, \mathcal{E}[t]) \sim^{\mathrm{wb}}_{\Gamma} (G', P')$, and $P' = t' \parallel P'_0$ where $\mathcal{E}[t] \sim^{\mathrm{wb}}_{\Gamma} t'$.*

Proof. Let $(\lambda, \mathcal{C}) = \mathcal{E}$, define $t_0 = t \cup \lambda|_{\Gamma,high}$, and observe that by Lemma 79, $t_0.ls.locks \supseteq \lambda \cap t_0.cmd.locks$. Apply Lemma 33 to reduction $(G, \mathcal{E}[t]) = (G, \mathcal{E}[nil \mid t_0]) \longrightarrow^{eval} (G', P')$ to find $P' = \mathcal{E}[t'] \parallel P'_0$ and $(G, t_0) \longrightarrow^{eval} (G', t' \parallel P'_0)$.

A low write, acquiring a low lock, or releasing a low lock would contradict t's *high* type. Therefore $G \sim^{\mathrm{wb}}_{\Gamma} G'$.

We show that $P'_0 \sim^{\mathrm{wb}}_{\Gamma} \mathfrak{o}$. If the step was not by EC-FORK this is trivial. Otherwise $P'_0 = \langle L_{\oslash}, c_0\rangle \parallel \mathfrak{o}$ and it remains to show that $high; high; \Gamma \vdash^{\mathrm{wb}} c_0 \Rightarrow ut$ for some ut. This follows from inverting t's typing derivation.

Finally we show that $\mathcal{E}[t] = \mathcal{E}[t_0] \sim^{\mathrm{wb}}_{\Gamma} \mathcal{E}[t']$. Again, acquiring or releasing a low lock, or putting a low write in the write buffer, would contradict t_0's *high* type. Therefore $t_0.ls \sim^{\mathrm{wb}}_{\Gamma} t'.ls$. From the type of t is follows that $high; wt; \Gamma \vdash^{\mathrm{wb}} t_0$, and preservation (Lemma 63) shows $high; vt; \Gamma \vdash^{\mathrm{wb}} t'$ for some vt where $wt \sqsubseteq vt$. To conclude we must show that t_0 and t' may be typed with the same write typing. If $t_0.wb$ contains a low write then so does $t'.wb$ and because $wt = vt = low$ we're done. If $t_0.wb$ does not contain a low write then inverting the typing derivation and using Lemma 57 shows $high; vt; \Gamma \vdash^{\mathrm{wb}} t_0$, allowing us to conclude. $\square$

Lemma 82 (WB Commit step confinement)**.** *Suppose $pc; high; \Gamma \vdash^{\mathrm{wb}} t$ and $(G, t) \longrightarrow^{commit} (G', P')$. Then $(G, t) \sim^{\mathrm{wb}}_{\Gamma} (G', P')$.*

Proof. Suppose a high value is committed; then the conclusion is immediate. Suppose instead a low value is committed, this contradicts they typing of t. $\square$

Lemma 83 (WB Global confinement)**.** *Suppose $high; high; \Gamma \vdash^{\mathrm{wb}} t$ and $(G, t) \longrightarrow^{op} (G', P')$. Then $(G, t \parallel \mathfrak{o}) \sim^{\mathrm{wb}}_{\Gamma} (G', P')$.*

Proof. Suppose $op = commit$. Conclude via Lemma 82. Instead suppose $op = eval$ and the step is not by EC-REAP. Inverting the step relation shows t can be rewritten in form $\mathcal{E}[\langle L, c\rangle]$ where $c \neq \mathbf{skip}$ and we conclude via Lemma 81. Finally suppose the step is by EC-REAP and conclude by observing $P' = \mathfrak{o} \sim^{\mathrm{wb}}_{\Gamma} t \parallel \mathfrak{o}$. $\square$

Lemma 84 (WB Expression Confinement)**.** *Suppose $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$. Then $low; \Gamma \vdash^{\mathrm{tso}} a$ implies $i_1 = i_2$ when $L_1[a] \Downarrow i_1$ and $L_2[a] \Downarrow i_2$. Likewise, $low; \Gamma \vdash^{\mathrm{tso}} b$ implies $\beta_1 = \beta_2$ when $L_1[b] \Downarrow \beta_1$ and $L_2[b] \Downarrow \beta_2$.*

Proof. by induction $\square$

Lemma 85 (Strong inversion for $\sim^{\mathrm{wb}}$)**.** *Suppose $t_1 \sim^{\mathrm{wb}}_{\Gamma} t_2$. Then at least one the following conditions holds. Either,*

(i) $t_1 = \langle L_1, c\rangle$ *and* $t_2 = \langle L_2, c\rangle$ *where* $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$*, or*

(ii) $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ *where* $c_1 \neq \textbf{skip}$ *and* $high; wt; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle$ *for some* wt*, or*

(iii) $t_1 = \mathcal{E}[\langle L_1, \textbf{skip}\rangle]$ *and* $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$*, where the following hold for some* wt*:*

$$c_2 \neq \textbf{skip}, \qquad L_1 \sim^{\text{wb}}_{\Gamma} L_2, \qquad \textit{canEval } t_1 \textit{ implies active } \mathcal{E}, \qquad high; wt; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle, \textit{ and}$$

$$high; wt; \Gamma \vdash^{\text{wb}} \langle L_2, c_2\rangle.$$

Proof. Suppose that t_1 and t_2 are related by Definition 21, case 3a. Conclude as (i) is satisfied.

Suppose instead that t_1 and t_2 are related by case 3b, so Then $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ and $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$ and both $high; wt; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle$ and $high; wt; \Gamma \vdash^{\text{wb}} \langle L_2, c_2\rangle$. Additionally $L_1 \sim^{\text{wb}}_{\Gamma} L_2$. If $c_1 \neq \textbf{skip}$ then condition (ii) is satisfied. Suppose instead that $c_1 = \textbf{skip}$. If $c_2 = \textbf{skip}$ then using Lemma 69 we see that condition (i) is satisfied. Now suppose $c_2 \neq \textbf{skip}$. If it's not the case that *canEval* t_1 then (iii) is satisfied and we conclude. Otherwise use Lemma 2 to find $\mathcal{E}_0$ where *active* $\mathcal{E}_0$ and $\mathcal{E}_0[t_i] = \mathcal{E}[t_i]$. Conclude as condition (iii) is satisfied. ☐

Quiet traces relate consecutive trace components with by $\sim^{\text{wb}}$.

Definition 23 (Quiet traces)**.** *Call trace* $\mathcal{T} = (G_1, P_1), (G_2, P_2), \ldots, (G_n, P_n)$ *quiet in context* Γ*, written* $\textit{quiet}_{\Gamma}\ \mathcal{T}$*, when for each pair of consecutive configurations,* $(G_i, P_i \parallel t_i \parallel R_i)$ *and* $(G_{i+1}, P_i \parallel Q_{i+1} \parallel R_i)$ *where* $(G_i, t_i) \longrightarrow^{op} (G_{i+1}, Q_{i+1})$*, one or more equivalences hold. First,* $(G_i, t_i \parallel \mathsf{o}) \sim^{\text{wb}}_{\Gamma} (G_{i+1}, Q_{i+1})$*. Second, if* $Q_{i+1} = t_{i+1} \parallel Q^0_{i+1}$ *then* $t_{i+1} \sim^{\text{wb}}_{\Gamma} t_i$ *and* $Q^0_{i+1} \sim^{\text{wb}}_{\Gamma} \mathsf{o}$*.*

Lemma 86. *Suppose* $\mathcal{T} :: (G, t \parallel P) \Longrightarrow^{m*} (G', t' \parallel P')$ *where* $\textit{quiet}_{\Gamma}\ \mathcal{T}$ *and* *FrontReapFree* $\mathcal{T}$*. Then* $t \sim^{\text{wb}}_{\Gamma} t'$ *and* $P \sim^{\text{wb}}_{\Gamma} P'$ *and* $G \sim^{\text{wb}}_{\Gamma} G'$*.*

Proof. Proof by an easy induction on the length of $\mathcal{T}$. ☐

Definition 24 (Syntactically held locks)**.**

$$\begin{aligned}
\textit{synlocks } \textbf{skip} &= \emptyset \\
\textit{synlocks } (Y := x) &= \emptyset \\
\textit{synlocks } (x := Y) &= \emptyset \\
\textit{synlocks } (x := a) &= \emptyset \\
\textit{synlocks } \textbf{fence} &= \emptyset \\
\textit{synlocks } (\textbf{fork } c) &= \textit{synlocks } c \\
\textit{synlocks } (\textbf{sync } \ell \textbf{ do } c) &= \textit{synlocks } c \\
\textit{synlocks } (\textbf{holding } \ell \textbf{ do } c) &= \{\ell\} \cup \textit{synlocks } c \\
\textit{synlocks } (c_1;\, c_2) &= \textit{synlocks } c_1 \cup \textit{synlocks } c_2 \\
\textit{synlocks } (\textbf{if } b \textbf{ do } c_1 \textbf{ else } c_2) &= \textit{synlocks } c_1 \cup \textit{synlocks } c_2 \\
\textit{synlocks } (\textbf{while } b \textbf{ do } c) &= \textit{synlocks } c \\
\\
\textit{synlocks } \langle L, c\rangle &= \textit{synlocks } c
\end{aligned}$$

Lemma 87. *Suppose active* $\mathcal{E}$ *and wellStruct* t *where* $\textit{synlocks } t \subseteq t|_{\Gamma, high}$ *and* $high; wt; \Gamma \vdash^{\text{wb}} t$*. Also suppose that* $t|_{\Gamma, high}$ *and* $G|_{\Gamma, high}$ *partition* $\textbf{Lock}|_{\Gamma, high}$*. Then* $\mathcal{T} :: (G, \mathcal{E}[t] \parallel \mathsf{o}) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[\langle L', \textbf{skip}\rangle] \parallel P')$ *where* $\textit{quiet}_{\Gamma}\ \mathcal{T}$ *and FrontReapFree* $\mathcal{T}$*, and where* $L'|_{\Gamma, high}$ *and* $G'|_{\Gamma, high}$ *partition* $\textbf{Lock}|_{\Gamma, high}$ *and where* $L'|_{\Gamma, high} = t|_{\Gamma, high} \setminus (\textit{synlocks } t)|_{\Gamma, high}$ *and* $P'|_{\Gamma, high} = \emptyset$*. Finally, if* $wt = high$ *then* $L'.wb = nil$*.*

Proof. Let $\langle L, c\rangle = t$. Proof is by strong induction on *size* c + *size* $L.wb$. Proceed by inverting the typing derivation. The interesting cases are as follows:

wb-Load: Here $c = x := Y$ and $high \sqsubseteq \Gamma(x)$. Construct a trace showing $(G, \mathcal{E}[t] \parallel \mathsf{o}) \Longrightarrow^{\mathsf{sc}} (G, \langle L[x \mapsto i], \textbf{skip}\rangle \parallel \mathsf{o})$. This is *FrontReapFree*. The trace is *quiet* because $\Gamma(x) = high$ typing ensures $L \sim^{\text{wb}}_{\Gamma} L[x \mapsto i]$ and because $high; wt; \Gamma \vdash^{\text{wb}} \textbf{skip} \Rightarrow wt$. Conclude using the induction hypothesis, which is necessary to ensure that $wt = high$ implies an empty output write buffer.

wb-Store, wb-Eval: Similar to wb-Load. .

WB-SYNC: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$ with $high \sqsubseteq \Gamma(\ell) = high$ and $high \sqsubseteq wt = high$ and $high; high; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow ut$.

Suppose $L = (X := i)::L_0$. Because $wt = high$ we have that $\Gamma(X) = high$. Thus $(G, \mathcal{E}[t]) \Longrightarrow^{\mathsf{tso}} (G[X \mapsto i], \mathcal{E}[\langle L_0, c\rangle])$. This step is *FrontReapFree* and *quiet* because only the "high components" of L and G are modified. Conclude using the induction hypothesis to find a *FrontReapFree* and *quiet* trace witnessing $(G[X \mapsto i], \mathcal{E}[\langle L_0, c\rangle]) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[\langle L', \textbf{skip}\rangle])$ for an appropriate L' and G'.

Suppose instead that $L.wb = nil$ and $\ell \in L$. Reduce $\mathcal{E}[\langle L, c\rangle]$ to $\mathcal{E}[\langle L, \textbf{fence}; (c_0; \textbf{fence})\rangle]$ and conclude by invoking the induction hypothesis.

Finally, if $L.wb = nil$ and $\ell \notin L$ then reduce $\mathcal{E}[\langle L, c\rangle]$ to $\mathcal{E}[\langle L \cup \{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ c_0\rangle]$ and use the induction hypothesis to build a *quiet* and *FrontReapFree* derivation of

$$(G, \mathcal{E}[t]) \Longrightarrow^{\mathsf{tso}} (G \setminus \{\ell\}, \mathcal{E}[\langle L \cup \{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ c_0\rangle]) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[\langle L', \textbf{skip}\rangle] \parallel P').$$

By the induction hypothesis we have that $L'|_{\Gamma,high} = (L \cup \{\ell\})|_{\Gamma,high} \setminus (synlocks\ \textbf{holding}\ \ell\ \textbf{do}\ c_0)|_{\Gamma,high}$. From *wellStruct* t it follows that $synlocks\ c_0|_{\Gamma,high} = \emptyset$ and so $L'|_{\Gamma,high} = L|_{\Gamma,high}$ as required.

WB-HOLD: Here $c = \textbf{holding}\ \ell\ \textbf{do}\ c_0$ with $high \sqsubseteq \Gamma(\ell) = high$ and $\Gamma(\ell) \sqsubseteq wt = high$. As $\ell \in synlocks\ c \subseteq L|_{\Gamma,high}$, we can define $\mathcal{E}_0 = (\lambda \cup \{\ell\}, \mathcal{C}[\textbf{holding}\ \ell\ \textbf{do}\ [\cdot]])$ where $(\lambda, \mathcal{C}) = \mathcal{E}$ and *active* $\mathcal{E}'$. Inverting the typing relation shows $wt = \Gamma(\ell) = high$ and $high; high; \Gamma \vdash^{\mathrm{wb}} \langle L, c_0\rangle$. By the induction hypothesis we have a *quiet* and *FrontReapFree* trace showing $(G, \mathcal{E}_0[\langle L, c_0\rangle]) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}_0[\langle L', \textbf{skip}\rangle])$ where $L'.wb = nil$. Because *wellStruct* t, it is not the case that $\ell \notin synlocks\ c_0$, so $\ell \in L'$. Finish by extending this trace using EC-HOLDRELEASE.

WB-FENCE, WB-FORK: Similar to, but simpler than WB-SYNC.

WB-SEQ, WB-IF, WB-WHILE: Immediate by the induction hypothesis . ☐

Lemma 88. *Suppose active $\mathcal{E}$ and wellStruct t and $high; wt; \Gamma \vdash^{\mathrm{wb}} t$. Also suppose $t|_{\Gamma,high} = \emptyset$ and $G|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$. Then $\mathcal{T} :: (G, \mathcal{E}[t] \parallel \mathsf{o}) \Longrightarrow^{\mathsf{tso}*} (G', \mathcal{E}[\langle L', \textbf{skip}\rangle] \parallel P')$ where $quiet_\Gamma\ \mathcal{T}$ and FrontReapFree $\mathcal{T}$, and where $L'|_{\Gamma,high} = P'|_{\Gamma,high} = \emptyset$ and $G'|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$.*

Proof. Immediate using Lemma 87. ☐

Lemma 89 (WB commit step security)**.** *Suppose the following hold.*

$$(G_1, t_1) \longrightarrow^{commit} (G_1', t_1' \parallel \mathsf{o}) \qquad G_1 \sim_\Gamma^{\mathrm{wb}} G_2 \qquad t_1 \sim_\Gamma^{\mathrm{wb}} t_2$$

Then $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', t_2')$ where $(G_1', t_1') \sim_\Gamma^{\mathrm{wb}} (G_2', t_2')$ and both $t_2'.locks = t_2.locks$ and $G_2'.locks = G_2.locks$.

Proof. Suppose that the $\longrightarrow^{commit}$ operation commits write $X := i$. Consider the case where $\Gamma(X) = high$. We can conclude immediately, taking $t_2' = t_2$ and $G_2' = G_2$.

Suppose instead that $\Gamma(X) = low$. By the definition of $\sim^{\mathrm{wb}}$ we have that $L_2.wb = W_{21} +\!\!+ (X := i) +\!\!+ W_{22}$ where for each $(Y := j) \in W_{21}$, it is the case $\Gamma(Y) = high$. Let n denote the number of writes in W_{21} and define t_2' and G_2' by taking $n + 1$ *commit* steps. ☐

Lemma 90 (WB Eval step security)**.** *Suppose the following hold.*

$$(G_1, t_1) \longrightarrow^{eval} (G_1', P_1') \qquad pc; wt; \Gamma \vdash^{\mathrm{wb}} t_1 \qquad wellStruct\ t_1$$

$$G_1 \sim_\Gamma^{\mathrm{wb}} G_2 \qquad t_1 \sim_\Gamma^{\mathrm{wb}} t_2 \qquad t_2|_{\Gamma,high} = \emptyset \qquad G_2|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$$

Then there exist G_2' and P_2' such that $(G_1', P_1') \sim_\Gamma^{\mathrm{wb}} (G_2', P_2')$ and $(G, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$. Furthermore $P_2'|_{\Gamma,high} = \emptyset$ and $G_2'|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$.

Proof. We strengthen the induction hypothesis as follows. Whenever $P_1' = t_1' \parallel P_{10}'$ there exist $\mathcal{T}, t_2'$ and P_{20}' where $\mathcal{T} :: (G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$ and *FrontReapFree* $\mathcal{T}$ and $P_2' = t_2' \parallel P_{20}'$ and where both $t_1' \sim_\Gamma^{\mathrm{wb}} t_2'$ and $P_{10}' \sim_\Gamma^{\mathrm{wb}} P_{20}'$. Proceed with strong induction on quantity $(size\ (t_1.cmd) + size\ (t_2.cmd))$. Invert $t_1 \sim_\Gamma^{\mathrm{wb}} t_1$ using Lemma 85 to get three cases. Subcases of (i) will occasionally be completed by "falling through" to (ii).

(i) $t_1 = \langle L_1, c\rangle$ and $t_2 = \langle L_2, c\rangle$ where $L_1 \sim^{\text{wb}}_{\Gamma} L_2$. Continue by inverting the $\longrightarrow^{commit}$ derivation.

EC-STORE: Here $c = X := y$. If $\Gamma(y) = \mathit{low}$, the definition of $\sim^{\text{wb}}_{\Gamma}$ shows $L_1(y) = L_2(y)$ and we conclude using Lemma 72. Otherwise $\Gamma(y) = \mathit{high}$, inverting typing rule WB-STORE gives $\Gamma(X) = \mathit{high}$, and the result follows from Lemma 73.

EC-LOAD: Here $c = x := Y$. If $\Gamma(x) = \mathit{high}$ we conclude as updating L_1 and L_2 is not observable. If instead $\Gamma(x) = \mathit{low}$ then typing ensures $\Gamma(Y) = \mathit{low}$ and equivalences $L_1 \sim^{\text{wb}}_{\Gamma} L_2$ and $G_1 \sim^{\text{wb}}_{\Gamma} G_2$, ensures we're writing identical values to x.

EC-EVALEXP: Follows from Lemmas 70 and 71.

EC-SYNCACQUIRE: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$ and both $L_1.wb = \mathit{nil}$ and $\ell \in G_1$. First suppose that $\Gamma(\ell) = \mathit{high}$. Let $\mathcal{E} = (L_1|_{\Gamma,low}, [\cdot])$ and note that $t_1 = \mathcal{E}[\langle L_1 \setminus L_1|_{\Gamma,low}, c\rangle]$. Inverting typing rule WB-SYNC shows that $\mathit{high}; \mathit{wt}; \Gamma \vdash^{\text{wb}} c_0 \Rightarrow \mathit{ut}$ for some wt and ut; using WB-SYNC, and noting that $L_1 \setminus L_1|_{\Gamma,low}$ has both an empty write buffer and no *low* locks, lets us find $\mathit{high}; \mathit{wt}; \Gamma \vdash^{\text{wb}} \langle L_1 \setminus L_1|_{\Gamma,low}, c\rangle$. Continue by falling through to case (ii).

Now suppose $\Gamma(\ell) = \mathit{low}$. Here we will use that, from inversion, $G'_1 = G_1 \setminus \{\ell\}$ and $L'_1 = L_1 \cup \{\ell\}$. By the definition of $\sim^{\text{wb}}$ and fact $L_1.wb = \mathit{nil}$, we see that each write $(X := i)$ in $L_2.wb$ has $\Gamma(X) = \mathit{high}$. We can construct a derivation showing $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G'_{20}, \langle L'_{20}, c\rangle)$ where $(G_2, t_2) \sim^{\text{wb}}_{\Gamma} (G'_{20}, \langle L'_{20}, c\rangle)$, using finitely many $\longrightarrow^{commit}$ steps, each committing *high* variables. Applying the definition of $\sim^{\text{wb}}$ gives $\ell \in G'_{20}$ and $L'_{20}.wb = \mathit{nil}$. Taking an EC-SYNCACQUIRE step gives derivation $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G'_{20} \setminus \{\ell\}, \langle L'_{20} \cup \{\ell\}, c_0\rangle)$. We take this result state to be (G'_2, P'_2) and observe that $P'_2|_{\Gamma,high} = (L'_{20} \cup \{\ell\})|_{\Gamma,high} = (L_2 \cup \{\ell\})|_{\Gamma,high} = \emptyset$. It suffices to show $G'_{20} \setminus \{\ell\} \sim^{\text{wb}}_{\Gamma} G'_1 \setminus \{\ell\}$, which follows from Lemmas 65 and 68, and $L'_{20} \cup \{\ell\} \sim^{\text{wb}}_{\Gamma} L'_1 \cup \{\ell\}$, which follows from Lemmas 65 and 69 .

EC-FENCE, EC-FORK, EC-HOLDRELEASE: Similar to the EC-SYNCACQUIRE case.

EC-SYNCREENTER: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$. If $\Gamma(\ell) = \mathit{high}$ then using an argument to similar to the EC-SYNACQUIRE case fall through to (ii). If $\Gamma(\ell) = \mathit{low}$ then by $\sim^{\text{wb}}$ t_1 and t_2 transition in lockstep and the case is trivial.

EC-HOLDSTEP: Here $c = \textbf{holding}\ \ell\ \textbf{do}\ c_0$.

Suppose that $\Gamma(\ell) = \mathit{high}$. As in case EC-SYNCACQUIRE we fall through to case (ii).

Suppose instead that $\Gamma(\ell) = \mathit{low}$. Let $t_{10} = \langle L_1, c_0\rangle$ and $t_{20} = \langle L_2, c_0\rangle$ as well as $\mathcal{E} = (\{\ell\}, \textbf{holding}\ \ell\ \textbf{do}\ [\cdot])$. Inverting the evaluation relation gives $\ell \in L_1$ and $P'_1 = \mathcal{E}[t'_{10}] \parallel P'_{10}$ and $(G_1, t_{10}) \longrightarrow^{eval} (G'_1, t'_{10})$. Applying the induction hypothesis to this *eval* step, using Lemma 77 to establish $pc; \mathit{wt}; \Gamma \vdash^{\text{wb}} t_{10}$. This yields, among other properties,

$$\mathcal{T} :: (G_2, t_{20}) \Longrightarrow^{\mathsf{tso}*} (G'_2, t'_{20})$$

where *FrontReapFree* $\mathcal{T}$. Take P'_2 to be $\mathcal{E}[t'_{20}] \parallel P'_{20}$ and finish by applying Lemma 75. .

EC-SEQSTRUCT: Similar to EC-HOLDSTEP.

EC-SEQSKIP: Immediate.

EC-IFTRUE: Here $c = \textbf{if}\ b\ \textbf{do}\ c_t\ \textbf{else}\ c_f$ where $L_1[b] \Downarrow \textbf{true}$ and both $P'_1 = \langle L_1, c_t\rangle$ and $G'_1 = G_1$.

Suppose it's not the case that $\Gamma \vdash b : \mathit{low}$. Then inverting the typing relation shows both $\mathit{high}; \mathit{wt}; \Gamma \vdash^{\text{wb}} c_t \Rightarrow \mathit{ut}_t$ and $\mathit{high}; \mathit{wt}; \Gamma \vdash^{\text{wb}} c_f \Rightarrow \mathit{ut}_f$, as well as $\mathit{wt}; \Gamma \vdash^{\text{wb}} L_1.wb$. Without loss of generality assume $L_2[b] \Downarrow \textbf{false}$ and let $(G'_2, P'_2) = (G'_2, \langle L_2, c_f\rangle)$. It suffices to show that $\langle L_1, c_t\rangle \sim^{\text{wb}}_{\Gamma} \langle L_2, c_f\rangle$, which is a consequence of Lemma 80.

Suppose instead that that $\Gamma \vdash b : \mathit{low}$. Lemma 84 shows $L_2[b] \Downarrow \textbf{true}$ so $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2, \langle L_2, c_t\rangle) \sim^{\text{wb}}_{\Gamma} (G_1, \langle L_1, c_t\rangle) = (G'_1, P'_1)$.

EC-IFFALSE, EC-WHILETRUE, EC-WHILEFALSE: Similar to, or simpler than, EC-IFTRUE.

EC-REAP Trivial. .

(ii) $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ where $c_1 \neq \textbf{skip}$ and $\mathit{high}; \mathit{wt}_1; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle$ for some wt_1. By transitivity (Lemma 65) it suffices to show $(G'_1, P'_1) \sim^{\text{wb}}_{\Gamma} (G_1, t_1)$. Conclude via Lemma 81.

(iii) $t_1 = \mathcal{E}[\langle L_1, \textbf{skip}\rangle]$ and $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$. We know the following for some wt_0.

$$c_2 \neq \textbf{skip} \qquad L_1 \sim^{\text{wb}}_{\Gamma} L_2 \qquad \mathit{active}\ \mathcal{E} \qquad \mathit{high}; \mathit{wt}_0; \Gamma \vdash^{\text{wb}} \langle L_1, c_1\rangle \qquad \mathit{high}; \mathit{wt}_0; \Gamma \vdash^{\text{wb}} \langle L_2, c_2\rangle$$

By Lemma 88, we have $\mathcal{T} :: (G, \mathcal{E}[\langle L_2, c_2\rangle] \parallel \circ) \Longrightarrow^{\mathsf{tso}*} (G_2', \mathcal{E}[\langle L_2', \mathbf{skip}\rangle] \parallel P_2')$ where $\mathit{quiet}_\Gamma\ \mathcal{T}$, $\mathit{FrontReapFree}\ \mathcal{T}$, and $L_2'|_{\Gamma,\mathit{high}} = P_2'|_{\Gamma,\mathit{high}} = \emptyset$. By Lemmas 65 and 86 we find:

$$\mathcal{E}[\langle L_2', \mathbf{skip}\rangle] \sim_\Gamma^{\mathrm{wb}} \mathcal{E}[\langle L_2, c_2\rangle] \sim_\Gamma^{\mathrm{wb}} t_1$$

$$G_2' \sim_\Gamma^{\mathrm{wb}} G_2 \sim_\Gamma^{\mathrm{wb}} G_1$$

$$P_2' \sim_\Gamma^{\mathrm{wb}} \circ$$

Because $c_2 \neq \mathbf{skip}$ it is the case that $\mathit{size}\,(\mathcal{E}[\langle L_2', \mathbf{skip}\rangle].\mathit{cmd}) < \mathit{size}\,(\mathcal{E}[\langle L_2, c_2\rangle].\mathit{cmd})$, so we can use the induction hypothesis to find G_2'' and P_2'' such that $(G_1', P_1') \sim_\Gamma^{\mathrm{wb}} (G_2'', P_2'')$ and $(G_2', \mathcal{E}[\langle L_2', \mathbf{skip}\rangle]) \Longrightarrow^{\mathsf{tso}*} (G_2'', P_2'')$. By Lemma 66, $(G_1', P_1') \sim_\Gamma^{\mathrm{wb}} (G_2'', P_2'' \parallel P_2')$. Thus it suffices to show $(P_2'' \parallel P_2')|_{\Gamma,\mathit{high}} = \emptyset$, which is immediate, and $(G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G_2'', P_2'' \parallel P_2')$, which is a consequence of Lemma 3. ☐

Theorem 3 (WB Security)**.** *Suppose* $(G_1, P_1) \sim_\Gamma^{\mathrm{wb}} (G_2, P_2)$ *and* $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P_1$ *and wellStruct* P_1*. Suppose also that* $(G_1, P_1) \Longrightarrow^{\mathsf{tso}} (G_1', P_1')$*. Furthermore* $P_2|_{\Gamma,\mathit{high}} = \emptyset$ *and* $G_2|_{\Gamma,\mathit{high}} = \mathbf{Lock}|_{\Gamma,\mathit{high}}$*. Then there exists* G_2', P_2' *such that* $(G_1', P_1') \sim_\Gamma^{\mathrm{wb}} (G_2', P_2')$ *and* $(G_2, P_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$*, and both* $P_2'|_{\Gamma,\mathit{high}} = \emptyset$ *and* $G_2'|_{\Gamma,\mathit{high}} = \mathbf{Lock}|_{\Gamma,\mathit{high}}$*.*

Proof. Inverting the tso-evaluation relation and appealing to Lemma 76 gives

$$\begin{aligned} P_1 &= P_{11} \parallel t_1 \parallel P_{12} \\ P_2 &= P_{21} \parallel P_2^* \parallel P_{22} \\ P_1' &= P_{11} \parallel Q_1' \parallel P_{12} \end{aligned}$$

where P_2^* contains at most one thread (i.e., $P_2^* \in \{\circ, t_2 \parallel \circ\}$ for some t_2) and the following hold:

$$(G, t_1) \longrightarrow^{op} (G_1', Q_1')$$

$$P_{11} \sim_\Gamma^{\mathrm{wb}} P_{21}$$

$$t \parallel \circ \sim_\Gamma^{\mathrm{wb}} P_2^*$$

$$P_{12} \sim_\Gamma^{\mathrm{wb}} P_{22}$$

It suffices to show that there exists G_2' and Q_2' such that $(G_1', Q_1') \sim_\Gamma^{\mathrm{wb}} (G_2', Q_2')$ and $(G_2, P_2^*) \Longrightarrow^{\mathsf{tso}*} (G_2', Q_2')$. (Observe that while we could rename threads in Q_2', we do not need to; thread names are only really relevant for the data-race freedom argument.) Inspecting the definition of $\sim_\Gamma^{\mathrm{wb}}$ shows there are only three ways in which to find $t \parallel \circ \sim_\Gamma^{\mathrm{wb}} P_2^*$. Proceed by case analysis.

First suppose that that the equivalence arises from Definition 21, clause 5b. Here $\mathit{high}; \mathit{high}; \Gamma \vdash^{\mathrm{wb}} t_1$ and via Lemma 83, $(G_1, t \parallel \circ) \sim_\Gamma^{\mathrm{wb}} (G_1', Q_1')$. Conclude using Lemma 65, which states $\sim^{\mathrm{wb}}$ is an equivalence relation, and taking G_2 and P_2^* as existential witnesses G_2' and Q_2'.

Second suppose that that the equivalence arises from definition 21, clause 5c. Here $P_2^* = t_2 \parallel \circ$ where $\mathit{high}; \mathit{high}; \Gamma \vdash^{\mathrm{wb}} t_2$ and $t_1 \sim_\Gamma^{\mathrm{wb}} \circ$. From $t_1 \sim_\Gamma^{\mathrm{wb}} \circ$ it follows that $\mathit{high}; \mathit{high}; \Gamma \vdash^{\mathrm{wb}} t_1$. Again taking G_2 and P_2^* to be witnesses G_2' and Q_2' conclude with the following equational reasoning:

$$\begin{array}{lll} (G', Q_1') & \sim_\Gamma^{\mathrm{wb}} (G_1, t_1 \parallel \circ) & \text{by Lemma 83} \\ & \sim_\Gamma^{\mathrm{wb}} (G_1, \circ) & \\ & \sim_\Gamma^{\mathrm{wb}} (G_2, \circ) & \text{by assumption} \\ & \sim_\Gamma^{\mathrm{wb}} (G_2, t_2 \parallel \circ) & \\ & = (G_2, P_2^*) & \end{array}$$

Third suppose that that the equivalence arises from Definition 21, clause 5d. Here $P_2^* = t_2 \parallel \circ$ for some t_2 with $t_1 \sim_\Gamma^{\mathrm{wb}} t_2$. Finitely many inversions of the typing relation show $pc; wt; \Gamma \vdash^{\mathrm{wb}} t_1$ for some pc and wt. Similarly $\mathit{wellStruct}\ t_1$ and $t_2|_{\Gamma,\mathit{high}} = \emptyset$. Conclude via Lemmas 89 and 90. ☐

Corollary 5. *Suppose* $(G_1, P_1) \sim_\Gamma^{\mathrm{wb}} (G_2, P_2)$ *and* $\overline{pc}; \overline{wt}; \Gamma \vdash^{\mathrm{wb}} P_1$ *and wellStruct* P_1*. Suppose also that* $(G_1, P_1) \Longrightarrow^{\mathsf{tso}*} (G_1', P_1')$*. Furthermore* $P_2|_{\Gamma,\mathit{high}} = \emptyset$ *and* $G_2|_{\Gamma,\mathit{high}} = \mathbf{Lock}|_{\Gamma,\mathit{high}}$*. Then there exist* G_2' *and* P_2' *such that* $(G_1', P_1') \sim_\Gamma^{\mathrm{wb}} (G_2', P_2')$ *and* $(G_2, P_2) \Longrightarrow^{\mathsf{tso}*} (G_2', P_2')$ *and* $P_2'|_{\Gamma,\mathit{high}} = \emptyset$*.*

Proof. By finitely many application of Theorem 3 and Lemmas 5 and 64. ☐

Corollary 6. *Suppose* $G_1 \sim^{\mathrm{wb}}_{\Gamma} G_2$ *and* $pc; wt; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow ut$ *and src* c*. Also assume* $G_2|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$*. If* $(G_1, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{tso}*} (G_1', \mathfrak{o})$ *then* $(G_2, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{tso}*} (G_2', \mathfrak{o})$ *for some* G_2' *where* $G_1' \sim^{\mathrm{wb}}_{\Gamma} G_2'$*.*

Proof. An instantiation of the first corollary of Theorem 3 shows (G_2, t) evaluates to a configuration related to $(G_1', \mathfrak{o})$, and preservation (Lemma 64) and Lemmas 5, 9, and 88, show this evaluates to pool $\mathfrak{o}$. ☐

Corollary 7 (WB Simple possibilistic noninterference)**.** *Suppose* $pc; wt; \Gamma \vdash^{\mathrm{wb}} c \Rightarrow ut$ *and src* c*. Then* c *is possibilistically noninterfering under* tso *and* Γ*.*

6 Expressive typing for SC programs

6.1 Typing

As above:

$$\text{Static Security Context} \quad \Gamma \quad ::= \quad \mathbf{HeapVar} \cup \mathbf{LocalVar} \cup \mathbf{Lock} \to \tau$$

The type system will borrow a shared notion of $\vdash^{\mathrm{wb}}$ for locks and write buffers. Note that while we wrote $wt; \Gamma \vdash^{\mathrm{wb}} W$ earlier, we will write $pc; \Gamma \vdash^{\mathrm{wb}} W$ here. This is okay because both wt and pc are metavariables ranging over labels.

$\boxed{pc; \Gamma \vdash^{\mathrm{sc}} c}$

$$\frac{pc \sqcup \Gamma(Y) \sqsubseteq \Gamma(x)}{pc; \Gamma \vdash^{\mathrm{sc}} x := Y}\ \textsc{sc-Load} \qquad \frac{pc \sqcup \Gamma(y) \sqsubseteq \Gamma(X)}{pc; \Gamma \vdash^{\mathrm{sc}} X := y}\ \textsc{sc-Store} \qquad \frac{\Gamma \vdash a : \tau \qquad pc \sqcup \tau \sqsubseteq \Gamma(x)}{pc; \Gamma \vdash^{\mathrm{sc}} x := a}\ \textsc{sc-Eval}$$

$$\frac{pc \sqsubseteq \Gamma(\ell) \qquad \Gamma(\ell); \Gamma \vdash^{\mathrm{sc}} c}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{sync}\ \ell\ \mathbf{do}\ c}\ \textsc{sc-Sync} \qquad \frac{pc \sqsubseteq \Gamma(\ell) \qquad \Gamma(\ell); \Gamma \vdash^{\mathrm{sc}} c}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{holding}\ \ell\ \mathbf{do}\ c}\ \textsc{sc-Hold} \qquad \frac{}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{fence}}\ \textsc{sc-Fence}$$

$$\frac{pc; \Gamma \vdash^{\mathrm{sc}} c}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{fork}\ c}\ \textsc{sc-Fork} \qquad \frac{pc; \Gamma \vdash^{\mathrm{sc}} c_1 \qquad pc; \Gamma \vdash^{\mathrm{sc}} c_2}{pc; \Gamma \vdash^{\mathrm{sc}} c_1;\ c_2}\ \textsc{sc-Seq}$$

$$\frac{\Gamma \vdash b : \tau \qquad pc \sqcup \tau; \Gamma \vdash^{\mathrm{sc}} c_1 \qquad pc \sqcup \tau; \Gamma \vdash^{\mathrm{sc}} c_2}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2}\ \textsc{sc-If} \qquad \frac{\Gamma \vdash b : low \qquad pc; \Gamma \vdash^{\mathrm{sc}} c}{low; \Gamma \vdash^{\mathrm{sc}} \mathbf{while}\ b\ \mathbf{do}\ c}\ \textsc{sc-While}$$

$$\frac{}{pc; \Gamma \vdash^{\mathrm{sc}} \mathbf{skip}}\ \textsc{sc-Skip}$$

$\boxed{pc; \Gamma \vdash^{\mathrm{sc}} t}$

$$\frac{pc; \Gamma \vdash^{\mathrm{wb}} \lambda \qquad pc; \Gamma \vdash^{\mathrm{wb}} W \qquad pc; \Gamma \vdash^{\mathrm{sc}} c}{pc; \Gamma \vdash^{\mathrm{sc}} \langle (M, \lambda, W), c\rangle_\iota}$$

$\boxed{\overline{pc}; \Gamma \vdash^{\mathrm{sc}} P}$

$$\frac{}{\cdot; \Gamma \vdash^{\mathrm{sc}} \mathfrak{o}} \qquad \frac{pc; \Gamma \vdash^{\mathrm{sc}} t \qquad \overline{pc}; \Gamma \vdash^{\mathrm{sc}} P}{pc, \overline{pc}; \Gamma \vdash^{\mathrm{sc}} t \parallel P}$$

Lemma 91 (SC Admissibility of subtyping)**.** *Suppose* $pc_1 \sqsubseteq pc_2$*. Then the following hold.*

(i) $pc_2; \Gamma \vdash^{\mathrm{sc}} pc_1$ *implies* $pc_1; \Gamma \vdash^{\mathrm{sc}} pc_1$

(ii) $pc_2; \Gamma \vdash^{\mathrm{sc}} t$ *implies* $pc_1; \Gamma \vdash^{\mathrm{sc}} t$*.*

Proof. Statement (i) follows from easy inductions on the typing derivations. Statement (ii) follows from (i) and Lemma 57. ☐

Lemma 92. *If $pc;\Gamma \vdash^{\mathrm{sc}} \mathcal{E}[t]$ then $pc;\Gamma \vdash^{\mathrm{sc}} t$.*

Proof. Let $\mathcal{E} = (\lambda, \mathcal{C})$. First use induction on the size of λ to show $pc;\Gamma \vdash^{\mathrm{sc}} \mathcal{E}_0[t]$ where $\mathcal{E}_0 = (\emptyset, \mathcal{C})$. Conclude by an easy structural induction on $\mathcal{C}$, using Lemma 57. ☐

Lemma 93. *Suppose that $high;\Gamma \vdash^{\mathrm{sc}} t$. Then $t.cmd.locks \subseteq \mathbf{Lock}|_{\Gamma,high}$.*

Proof. by trivial induction on the typing derivation. ☐

Lemma 94 (SC Commit Step Preservation)**.** *Suppose $pc;\Gamma \vdash^{\mathrm{sc}} t$ and $(G,t) \longrightarrow^{commit} (G', P')$. Then $\overline{pc};\Gamma;P' \vdash^{\mathrm{wb}} w$ here $pc \sqsubseteq \overline{pc}$.*

Proof. Let $\langle L, c\rangle = t$. Then $P' = \langle L_0, c\rangle \parallel \mathsf{o}$ where $L = (X := i) +\!\!+ L_0$ for some X, i, and L_0. We must show that $pc;\Gamma \vdash^{\mathrm{wb}} L_0.wb$, which follows from inverting the typing derivation. ☐

Lemma 95 (SC Eval Step Preservation)**.** *Suppose $pc;\Gamma \vdash^{\mathrm{sc}} t$ and $(G,t) \longrightarrow^{eval} (G', P')$. Then $\overline{pc};\Gamma \vdash^{\mathrm{sc}} P'$ where $pc \sqsubseteq \overline{pc}$.*

Proof. Let $\langle L, c\rangle = t$ and proceed by induction on the $\longrightarrow^{eval}$ step derivation.

EC-STORE: Here $c = (X := y)$ and $P' = \langle L +\!\!+ (X := i), \mathbf{skip}\rangle \parallel \mathsf{o}$ for some i. Inverting the typing derivation shows $pc \sqsubseteq \Gamma(X)$ and $pc;\Gamma \vdash^{\mathrm{sc}} L.wb$. We must show $pc;\Gamma \vdash^{\mathrm{wb}} L.wb +\!\!+ (X := i)$, which follows from Lemma 61. It remains to show $pc;\Gamma \vdash^{\mathrm{sc}} \mathbf{skip}$; this is immediate.

EC-LOAD, EC-EVALEXP, EC-FENCE: Immediate.

EC-SYNCACQUIRE: Here $c = \mathbf{sync}\ \ell\ \mathbf{do}\ c_0$ and $P' = \langle L \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c_0)\rangle \parallel \mathsf{o}$. Recall that $L.wb = nil$. Inverting the typing relation gives $pc \sqsubseteq \Gamma(\ell)$ and $\Gamma(\ell);\Gamma \vdash^{\mathrm{sc}} c_0$. Because the lock set changed, we must show that $pc;\Gamma \vdash^{\mathrm{wb}} L.locks \cup \{\ell\}$, which is immediate from Lemma 59. Conclude by constructing a derivation of $pc;\Gamma \vdash^{\mathrm{sc}} \mathbf{holding}\ \ell\ \mathbf{do}\ c_0$.

EC-SYNCREENTER: Here $c = \mathbf{sync}\ \ell\ \mathbf{do}\ c_0$ and $P' = \langle L, \mathbf{fence}; (c_0; \mathbf{fence})\rangle \parallel \mathsf{o}$. Inverting the typing relation shows $\Gamma(\ell);\Gamma \vdash^{\mathrm{sc}} c_0$ and $pc \sqsubseteq \Gamma(\ell)$ so that Lemma 91 yields a derivation showing $pc;\Gamma \vdash^{\mathrm{sc}} c_0$. It remains to construct the a derivation showing $pc;\Gamma \vdash^{\mathrm{sc}} \mathbf{fence}; (c_0; \mathbf{fence})$.

EC-HOLDSTEP: Here $c = \mathbf{holding}\ \ell\ \mathbf{do}\ c_0$ and $P' = \langle L', \mathbf{holding}\ \ell\ \mathbf{do}\ c_0'\rangle \parallel P_0'$ where $(G, \langle L, c_0\rangle) \longrightarrow^{eval} (G', \langle L', c_0'\rangle \parallel P_0')$. Inverting the typing derivation gives $\Gamma(\ell);\Gamma \vdash^{\mathrm{sc}} c_0$ and $pc \sqsubseteq \Gamma(\ell)$. Applying the induction hypothesis yields $\overline{pc};\Gamma \vdash^{\mathrm{sc}} \langle L', c_0'\rangle \parallel P_0'$ where $\Gamma(\ell) \sqsubseteq \overline{pc}$. It remains to show that $pc;\Gamma \vdash^{\mathrm{sc}} \mathbf{holding}\ \ell\ \mathbf{do}\ c_0'$, which follows from rule SC-HOLD and Lemma 91.

EC-HOLDRELEASE: Here $c = \mathbf{holding}\ \ell\ \mathbf{do}\ \mathbf{skip}$ and $P' = \langle L \setminus \{\ell\}, \mathbf{skip}\rangle \parallel \mathsf{o}$ with $L.wb = nil$. Noting that $pc;\Gamma \vdash^{\mathrm{wb}} L.wb$ and that Lemma 60 gives $pc;\Gamma \vdash^{\mathrm{wb}} L.locks \setminus \{\ell\}$, the result is immediate.

EC-FORK: Here $c = \mathbf{fork}\ c_0$ and $P' = \langle L, \mathbf{skip}\rangle \parallel \langle L_{\oslash}, c_0\rangle \parallel \mathsf{o}$ and $L.wb = nil$.

First we show that $pc;\Gamma \vdash^{\mathrm{sc}} \langle L_{\oslash}, c_0\rangle \parallel \mathsf{o}$. Inverting the typing relation gives $pc;\Gamma \vdash^{\mathrm{sc}} c_0$. Finish building easy derivations of $pc;\Gamma \vdash^{\mathrm{wb}} L_{\oslash}.locks$ and $pc;\Gamma \vdash^{\mathrm{wb}} L_{\oslash}.wb$.

Second we show $pc;\Gamma \vdash^{\mathrm{sc}} \mathbf{skip}$, and $pc;\Gamma \vdash^{\mathrm{wb}} L.wb$ both of which are immediate.

EC-SEQSTRUCT: Here $c = c_1; c_2$ and $P' = \langle L', c_1'\rangle \parallel P_0'$ where $(G, \langle L, c_1\rangle) \longrightarrow^{eval} (G', \langle L', c_1'\rangle \parallel P_0')$. Inverting the typing derivation gives $pc;\Gamma \vdash^{\mathrm{sc}} c_1$ and $pc;\Gamma \vdash^{\mathrm{sc}} c_2$.

The induction hypothesis shows $\overline{pc};\Gamma \vdash^{\mathrm{sc}} \langle L', c_1'\rangle \parallel P_0'$. where $pc \sqsubseteq \overline{pc}$. By Lemma 91, $pc;\Gamma \vdash^{\mathrm{sc}} \langle L', c_1'\rangle$, and we conclude by inverting this judgment and constructing constructing a derivation of $pc;\Gamma \vdash^{\mathrm{sc}} c_1'; c_2$ with SC-SEQ.

EC-SEQSKIP: Here $c = \mathbf{skip}; c_2$ and $P' = \langle L, c_2\rangle \parallel \mathsf{o}$. Trivial by inverting the typing derivation.

EC-IFTRUE, EC-IFFALSE, EC-WHILETRUE, EC-WHILEFALSE: Immediate, using Lemma 57 for EC-WHILETRUE.

EC-REAP: Immediate. ☐

Lemma 96 (SC Preservation)**.** *Suppose $\overline{pc};\Gamma \vdash^{\mathrm{sc}} P$ and $(G, P) \Longrightarrow^{\mathsf{sc}*} (G', P')$. Then there exist $\overline{pc}'$ such that $\overline{pc}';\Gamma \vdash^{\mathrm{sc}} P'$.*

Proof. By induction on the length of the $\Longrightarrow^{\mathsf{sc}*}$ derivation, using Lemmas 94 or 95. ☐

6.2 Equivalences

Definition 25 ($\sim_\Gamma^{\mathrm{sc}}$)**.**

1. *$t_1 \sim_\Gamma^{\mathrm{sc}} t_2$ is defined by the following introduction rules.*

   (a) *$\langle L_1, c\rangle_{\iota_1} \sim_\Gamma^{\mathrm{sc}} \langle L_2, c\rangle_{\iota_2}$ when $L_1 \sim_\Gamma^{\mathrm{wb}} L_2$*

   (b) *$\mathcal{E}[\langle L_1, c_1\rangle_{\iota_1}] \sim_\Gamma^{\mathrm{sc}} \mathcal{E}[\langle L_2, c_2\rangle_{\iota_2}]$ when $L_1 \sim_\Gamma^{\mathrm{wb}} L_2$ and both $high; \Gamma \vdash^{\mathrm{sc}} \langle L_1, c_1\rangle_{\iota_1}$ and $high; \Gamma \vdash^{\mathrm{sc}} \langle L_2, c_2\rangle_{\iota_2}$ for some wt.*

2. *$P_1 \sim_\Gamma^{\mathrm{sc}} P_2$ is defined by the least fixed point of the following implications.*

   (a) *$\mathsf{o} \sim_\Gamma^{\mathrm{sc}} \mathsf{o}$, always*

   (b) *$t \parallel P_1 \sim_\Gamma^{\mathrm{sc}} P_2$ when $high; \Gamma \vdash^{\mathrm{sc}} t$ and $P_1 \sim_\Gamma^{\mathrm{sc}} P_2$*

   (c) *$P_1 \sim_\Gamma^{\mathrm{sc}} t \parallel P_2$ when $high; \Gamma \vdash^{\mathrm{sc}} t$ and $P_1 \sim_\Gamma^{\mathrm{sc}} P_2$*

   (d) *$t_1 \parallel P_1 \sim_\Gamma^{\mathrm{sc}} t_2 \parallel P_2$ when $t_1 \sim_\Gamma^{\mathrm{sc}} t_2$ and $P_1 \sim_\Gamma^{\mathrm{sc}} P_2$*

3. *$(G_1, P_1) \sim_\Gamma^{\mathrm{sc}} (G_2, P_2)$ when $G_1 \sim_\Gamma^{\mathrm{wb}} G_2$ and $P_1 \sim_\Gamma^{\mathrm{sc}} P_2$.*

Lemma 97. *Each $\sim_\Gamma^{\mathrm{sc}}$ relation is an equivalence relation.*

Proof. by inspection. □

Lemma 98. *If $P_{11} \parallel t_1 \parallel P_{12} \sim_\Gamma^{\mathrm{sc}} P_2$ then $P_2 = P_{21} \parallel P_2^* \parallel P_{22}$ where the following hold:*

$$\begin{array}{rcl} P_{21} & \sim_\Gamma^{\mathrm{sc}} & P_{11} \\ P_2^* & \sim_\Gamma^{\mathrm{sc}} & t_1 \\ P_{22} & \sim_\Gamma^{\mathrm{sc}} & P_{12} \\ P_2^* & \in & \{\mathsf{o}, t_2 \parallel \mathsf{o}\} \text{ for some } t_2 \end{array}$$

Proof. By any easy induction on the sum of the lengths of P_1 and P_2. □

Lemma 99 (Strong inversion for $\sim^{\mathrm{sc}}$)**.** *Suppose $t_1 \sim_\Gamma^{\mathrm{sc}} t_2$. Then at least one the following conditions holds. Either,*

(i) $t_1 = \langle L_1, c\rangle$ and $t_2 = \langle L_2, c\rangle$ where $L_1 \sim_\Gamma^{\mathrm{sc}} L_2$, or

*(ii) $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ where $c_1 \neq$ **skip** and $high; \Gamma \vdash^{\mathrm{sc}} \langle L_1, c_1\rangle$, or*

(iii) $t_1 = \mathcal{E}[\langle L_1, \textbf{skip}\rangle]$ and $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$, where the following hold:

$$c_2 \neq \textbf{skip}, \qquad L_1 \sim_\Gamma^{\mathrm{wb}} L_2, \qquad \mathit{canEval}\ t_1 \text{ implies } \mathit{active}\ \mathcal{E}, \qquad high; \Gamma \vdash^{\mathrm{sc}} \langle L_1, c_1\rangle, \text{and}$$

$$high; \Gamma \vdash^{\mathrm{sc}} \langle L_2, c_2\rangle.$$

Proof. Suppose that t_1 and t_2 are related by Definition 25, case 1a. Conclude as (i) is satisfied.

Suppose instead that t_1 and t_2 are related by case 1b, so Then $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ and $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$ and both $high; \Gamma \vdash^{\mathrm{sc}} \langle L_1, c_1\rangle$ and $high; \Gamma \vdash^{\mathrm{sc}} \langle L_2, c_2\rangle$. Additionally $L_1 \sim_\Gamma^{\mathrm{wb}} L_2$. If $c_1 \neq$ **skip** then condition (ii) is satisfied. Suppose instead that $c_1 =$ **skip**. If $c_2 =$ **skip** then using Lemma 69 we see that condition (i) is satisfied. Now suppose $c_2 \neq$ **skip**. If it's not the case that *canEval* t_1 then (iii) is satisfied and we conclude. Otherwise use Lemma 2 to find $\mathcal{E}_0$ where *active* $\mathcal{E}_0$ and $\mathcal{E}_0[t_i] = \mathcal{E}[t_i]$. Conclude as condition (iii) is satisfied. □

Lemma 100. *Suppose $t_1 \sim_\Gamma^{\mathrm{sc}} t_2$. Then $\mathcal{E}[t_1] \sim_\Gamma^{\mathrm{sc}} \mathcal{E}[t_2]$.*

Lemma 101. *Suppose $L_1 \sim_\Gamma^{\mathrm{wb}} L_2$ and $high; \Gamma \vdash^{\mathrm{sc}} c_1$ and $high; \Gamma \vdash^{\mathrm{sc}} c_2$. Suppose also that $high; \Gamma \vdash^{\mathrm{wb}} L_1.wb$. Then $\langle L_1, c_1\rangle \sim_\Gamma^{\mathrm{sc}} \langle L_2, c_2\rangle$.*

Proof. By the definition of $\sim^{\mathrm{wb}}$ we want to find $\mathcal{E} = (\lambda, \mathcal{C})$, L_{10}, and L_{20}, such that the following hold.

$$\langle L_1, c_1\rangle = \mathcal{E}[\langle L_{10}, c_1\rangle] \qquad \langle L_2, c_2\rangle = \mathcal{E}[\langle L_{20}, c_2\rangle] \qquad high; \Gamma \vdash^{\mathrm{sc}} \langle L_{10}, c_1\rangle \qquad high; \Gamma \vdash^{\mathrm{sc}} \langle L_{20}, c_2\rangle$$

Let $\mathcal{C} = [\cdot]$ and $\lambda = L_1|_{\Gamma,low}$. Let also $L_{10} = L_1 \setminus \lambda$ and $L_{20} = L_2 \setminus \lambda$. Clearly $\langle L_1, c_1\rangle = \mathcal{E}[\langle L_{10}, c_1\rangle]$. Because $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$ it's also true that $\lambda = L_2|_{\Gamma,low}$, so $\langle L_2, c_2\rangle = \mathcal{E}[\langle L_{20}, c_2\rangle]$. We demonstrate $high; \Gamma \vdash^{\mathrm{sc}} \langle L_{10}, c_1\rangle$ by using the definition of L_{10} and the premises to show that $high; \Gamma \vdash^{\mathrm{wb}} L_{10}.locks$ and $high; \Gamma \vdash^{\mathrm{wb}} L_{10}.wb$. It remains to show $high; wt; \Gamma \vdash^{\mathrm{wb}} \langle L_{20}, c_2\rangle$, which works as above, using Lemma 74 to help establish $high; \Gamma \vdash^{\mathrm{wb}} L_{20}.wb$. ☐

Lemma 102. *If* $(G_1, P_1) \sim^{\mathrm{sc}}_{\Gamma} (G_2, P_{21})$ *and* $P_{22} \sim^{\mathrm{sc}}_{\Gamma} \mathfrak{o}$ *then* $(G_1, P_1) \sim^{\mathrm{sc}}_{\Gamma} (G_2, P_{21} \parallel P_{22})$

Proof. By any easy induction on the sum of the lengths of P_1 and P_2. ☐

6.3 Security proof

Lemma 103 (SC Eval step local confinement)**.** *Suppose* $t.cmd \neq \mathbf{skip}$ *and both* $high; \Gamma \vdash^{\mathrm{sc}} t$ *and* $(G, \mathcal{E}[t]) \longrightarrow^{eval} (G', P')$. *Then* $(G, \mathcal{E}[t]) \sim^{\mathrm{sc}}_{\Gamma} (G', P')$, *and* $P' = t' \parallel P'_0$ *where* $\mathcal{E}[t] \sim^{\mathrm{sc}}_{\Gamma} t'$.

Proof. Let $(\lambda, \mathcal{C}) = \mathcal{E}$, define $t_0 = t \cup \lambda|_{\Gamma,high}$, and observe that by Lemma 93, $t_0.ls.locks \supseteq \lambda \cap t_0.cmd.locks$. Apply Lemma 33 to reduction $(G, \mathcal{E}[t]) = (G, \mathcal{E}[nil \,|\, t_0]) \longrightarrow^{eval} (G', P')$ to find $P' = \mathcal{E}[t'] \parallel P'_0$ and $(G, t_0) \longrightarrow^{eval} (G', t' \parallel P'_0)$.

A low write, acquiring a low lock, or releasing a low lock would contradict t's *high* type. Therefore $G \sim^{\mathrm{wb}}_{\Gamma} G'$.

We show that $P'_0 \sim^{\mathrm{sc}}_{\Gamma} \mathfrak{o}$. If the step was not by EC-FORK this is trivial. Otherwise $c = \mathbf{fork}\ c_0$ and $P'_0 = \langle L_{\oslash}, c_0\rangle \parallel \mathfrak{o}$, and it remains to show that $high; \Gamma \vdash^{\mathrm{sc}} c_0$. This follows from inverting t's typing derivation.

Finally we show that $\mathcal{E}[t] = \mathcal{E}[t_0] \sim^{\mathrm{sc}}_{\Gamma} \mathcal{E}[t']$. Again, acquiring or releasing a low lock, or putting a low write in the write buffer, would contradict t_0's *high* type. Therefore $t_0.ls \sim^{\mathrm{sc}}_{\Gamma} t'.ls$. From the type of t is follows that $high; \Gamma \vdash^{\mathrm{sc}} t_0$, and preservation (Lemma 95) shows $high; \Gamma \vdash^{\mathrm{sc}} t'$. Therefore $\mathcal{E}[t_0] \sim^{\mathrm{sc}}_{\Gamma} \mathcal{E}[t']$ holds. ☐

Lemma 104 (SC Commit step confinement)**.** *Suppose* $high; \Gamma \vdash^{\mathrm{sc}} t$ *and* $(G, t) \longrightarrow^{commit} (G', P')$. *Then* $(G, t) \sim^{\mathrm{sc}}_{\Gamma} (G', P')$.

Proof. Suppose a high value is committed; then the conclusion is immediate. Suppose instead a low value is committed, this contradicts they typing of t. ☐

Lemma 105 (SC Global confinement)**.** *Suppose* $high; \Gamma \vdash^{\mathrm{sc}} t$ *and* $(G, t) \longrightarrow^{op} (G', P')$. *Then* $(G, t \parallel \mathfrak{o}) \sim^{\mathrm{sc}}_{\Gamma} (G', P')$.

Proof. Suppose $op = commit$. Conclude via Lemma 104. Instead suppose $op = eval$ and the step is not by EC-REAP. Inverting the step relation shows t can be rewritten in form $\mathcal{E}[\langle L, c\rangle]$ where $c \neq \mathbf{skip}$ and we conclude via Lemma 103. Finally suppose the step is by EC-REAP and conclude by observing $P' = \mathfrak{o} \sim^{\mathrm{wb}}_{\Gamma} t \parallel \mathfrak{o}$. ☐

Definition 26 (SC-Quiet traces)**.** *Call trace* $\mathcal{T} = (G_1, P_1), (G_2, P_2), \ldots, (G_n, P_n)$ *sc-quiet in context* Γ, *written* $quiet^{\mathrm{sc}}_{\Gamma}\ \mathcal{T}$, *when for each pair of consecutive configurations,* $(G_i, P_i \parallel t_i \parallel R_i)$ *and* $(G_{i+1}, P_i \parallel Q_{i+1} \parallel R_i)$ *where* $(G_i, t_i) \longrightarrow^{op} (G_{i+1}, Q_{i+1})$, *one or more equivalences hold. First,* $(G_i, t_i \parallel \mathfrak{o}) \sim^{\mathrm{sc}}_{\Gamma} (G_{i+1}, Q_{i+1})$. *Second, if* $Q_{i+1} = t_{i+1} \parallel Q^0_{i+1}$ *then* $t_{i+1} \sim^{\mathrm{sc}}_{\Gamma} t_i$ *and* $Q^0_{i+1} \sim^{\mathrm{sc}}_{\Gamma} \mathfrak{o}$.

Lemma 106. *Suppose* $\mathcal{T} :: (G, t \parallel P) \Longrightarrow^{m*} (G', t' \parallel P')$ *where* $quiet^{\mathrm{sc}}_{\Gamma}\ \mathcal{T}$ *and* $FrontReapFree\ \mathcal{T}$. *Then* $t \sim^{\mathrm{sc}}_{\Gamma} t'$ *and* $P \sim^{\mathrm{sc}}_{\Gamma} P'$ *and* $G \sim^{\mathrm{wb}}_{\Gamma} G'$.

Proof. Proof by an easy induction on the length of $\mathcal{T}$. ☐

Lemma 107. *Suppose active* $\mathcal{E}$ *and wellStruct* t *where* $synlocks\ t \subseteq t|_{\Gamma,high}$ *and* $high; \Gamma \vdash^{\mathrm{sc}} t$. *Also suppose that* $t|_{\Gamma,high}$ *and* $G|_{\Gamma,high}$ *partition* $\mathbf{Lock}|_{\Gamma,high}$. *Then* $\mathcal{T} :: (G, \mathcal{E}[t] \parallel \mathfrak{o}) \Longrightarrow^{\mathsf{sc}*} (G', \mathcal{E}[\langle L', \mathbf{skip}\rangle] \parallel P')$ *where* $quiet^{\mathrm{sc}}_{\Gamma}\ \mathcal{T}$ *and FrontReapFree* $\mathcal{T}$, *and where* $L'|_{\Gamma,high}$ *and* $G'|_{\Gamma,high}$ *partition* $\mathbf{Lock}|_{\Gamma,high}$ *and where* $L'|_{\Gamma,high} = t|_{\Gamma,high} \setminus (synlocks\ t)|_{\Gamma,high}$ *and* $P'|_{\Gamma,high} = \emptyset$ *and* $L'.wb = nil$. *Also hasEmptyWBs*$(\mathcal{E}[\langle L', \mathbf{skip}\rangle] \parallel P')$.

Proof. Let $\langle L, c\rangle = t$. Proof is by strong induction on *size* c + *size* $L.wb$.

First suppose that $L.wb = (X := i) +\!\!+ L_0$. Inverting the typing derivation shows $\Gamma(X) = high$. Using a commit step and invoking the induction hypothesis yields a appropriate $quiet^{sc}$, *FrontReapFree* trace with form,

$$(G, t) \Longrightarrow^{\mathsf{sc}} (G[X \mapsto i], \langle L_0, c\rangle) \Longrightarrow^{\mathsf{sc}*} (G[X \mapsto i], \langle L', \mathbf{skip}\rangle).$$

Suppose instead that $L.wb = nil$ and proceed by case analysis on the typing derivation.

sc-Load: Here $c = x := Y$ and $high \sqsubseteq \Gamma(x)$. Construct a trace showing $(G, \mathcal{E}[t] \parallel \circ) \Longrightarrow^{\mathsf{tso}} (G, \langle L[x \mapsto i], \mathbf{skip}\rangle \parallel \circ)$. This trace is $quiet^{sc}$ because $\Gamma(x) = high$ typing ensures $L \sim^{wb}_{\Gamma} L[x \mapsto i]$ and because $high; \Gamma \vdash^{sc} \mathbf{skip}$. Conclude using the induction hypothesis, which is necessary to ensure that $wt = high$ implies an empty output write buffer.

sc-Store, sc-Eval: Similar to sc-Load. In the sc-Store case we must also commit a *high* write to ensure that the output state *hasEmptyWBs*.

sc-Sync: Here $c = \mathbf{sync}\ \ell\ \mathbf{do}\ c_0$ with $high \sqsubseteq \Gamma(\ell) = high$ and $high; \Gamma \vdash^{sc} c$. Recall $L.wb = nil$ and

Suppose that $\ell \in L$. Reduce $\mathcal{E}[\langle L, c\rangle]$ to $\mathcal{E}[\langle L, \mathbf{fence}; (c_0; \mathbf{fence})\rangle]$ and conclude by invoking the induction hypothesis.

Suppose instead that $\ell \notin L$ then reduce $\mathcal{E}[\langle L, c\rangle]$ to $\mathcal{E}[\langle L \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c_0\rangle]$ and use the induction hypothesis to build a $quiet^{sc}$ and *FrontReapFree* derivation of

$$(G, \mathcal{E}[t]) \Longrightarrow^{\mathsf{sc}} (G \setminus \{\ell\}, \mathcal{E}[\langle L \cup \{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ c_0\rangle]) \Longrightarrow^{\mathsf{sc}*} (G', \mathcal{E}[\langle L', \mathbf{skip}\rangle] \parallel P').$$

By the induction hypothesis we have that $L'|_{\Gamma,high} = (L \cup \{\ell\})|_{\Gamma,high} \setminus (synlocks\ \mathbf{holding}\ \ell\ \mathbf{do}\ c_0)|_{\Gamma,high}$. From *wellStruct* t it follows that $synlocks\ c_0|_{\Gamma,high} = \emptyset$ and so $L'|_{\Gamma,high} = L|_{\Gamma,high}$ as required.

sc-Hold: Here $c = \mathbf{holding}\ \ell\ \mathbf{do}\ c_0$ with $high \sqsubseteq \Gamma(\ell) = high$. As $\ell \in synlocks\ c \subseteq L|_{\Gamma,high}$, we can define $\mathcal{E}_0 = (\lambda \cup \{\ell\}, \mathcal{C}[\mathbf{holding}\ \ell\ \mathbf{do}\ [\cdot]])$ where $(\lambda, \mathcal{C}) = \mathcal{E}$ and *active* $\mathcal{E}'$. Inverting the typing relation shows $wt = \Gamma(\ell) = high$ and $high; high; \Gamma \vdash^{wb} \langle L, c_0\rangle$. By the induction hypothesis we have a $quiet^{sc}$ and *FrontReapFree* trace showing $(G, \mathcal{E}_0[\langle L, c_0\rangle]) \Longrightarrow^{\mathsf{sc}*} (G', \mathcal{E}_0[\langle L', \mathbf{skip}\rangle])$ where $L'.wb = nil$. Because *wellStruct* t, it is not the case that $\ell \notin synlocks\ c_0$, so $\ell \in L'$. Finish by extending this trace using ec-HoldRelease.

sc-Fence, sc-Fork: Similar to, but simpler than sc-Sync. Note that in the sc-Fork case the spawned thread has an empty write buffer and holds no locks.

sc-Seq, sc-If, sc-While: Immediate by the induction hypothesis. ☐

Lemma 108. *Suppose active $\mathcal{E}$ and wellStruct t and $high; \Gamma \vdash^{sc} t$. Also suppose $t|_{\Gamma,high} = \emptyset$ and $G|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$. Then $\mathcal{T} :: (G, \mathcal{E}[t] \parallel \circ) \Longrightarrow^{\mathsf{sc}*} (G', \mathcal{E}[\langle L', \mathbf{skip}\rangle] \parallel P')$ where $quiet^{sc}_{\Gamma}\ \mathcal{T}$ and FrontReapFree $\mathcal{T}$, and where $L'|_{\Gamma,high} = P'|_{\Gamma,high} = \emptyset$ and $G'|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$. Also hasEmptyWBs$(\mathcal{E}[\langle L', \mathbf{skip}\rangle] \parallel P')$.*

Proof. Immediate using Lemma 107. ☐

Lemma 109 (SC commit step security)**.** *Suppose the following hold.*

$$(G_1, t_1) \longrightarrow^{commit} (G_1', t_1' \parallel \circ) \qquad G_1 \sim^{sc}_{\Gamma} G_2 \qquad t_1 \sim^{sc}_{\Gamma} t_2$$

Then $(G_2, t_2) \Longrightarrow^{\mathsf{sc}*} (G_2', t_2')$ where $(G_1', t_1') \sim^{sc}_{\Gamma} (G_2', t_2')$ and both $t_2'.locks = t_2.locks$ and $G_2'.locks = G_2.locks$.

Proof. Suppose that the $\longrightarrow^{commit}$ operation commits write $X := i$. Consider the case where $\Gamma(X) = high$. We can conclude immediately, taking $t_2' = t_2$ and $G_2' = G_2$.

Suppose instead that $\Gamma(X) = low$. By the definition of $\sim^{sc}$ we have that $L_2.wb = W_{21} +\!\!+ (X := i) +\!\!+ W_{22}$ where for each $(Y := j) \in W_{21}$, it is the case $\Gamma(Y) = high$. (Although dynamic sc execution traces will never have more than one write buffer entry, we do not require a premise of this form.) Let n denote the number of writes in W_{21} and define t_2' and G_2' by taking $n + 1$ *commit* steps. ☐

Lemma 110 (SC Eval step security)**.** *Suppose the following hold.*

$$(G_1, t_1) \longrightarrow^{eval} (G'_1, P'_1) \qquad pc; \Gamma \vdash^{\mathrm{sc}} t_1 \qquad wellStruct\ t_1 \qquad t_1.wb = nil$$

$$G_1 \sim^{\mathrm{sc}}_{\Gamma} G_2 \qquad t_1 \sim^{\mathrm{sc}}_{\Gamma} t_2 \qquad t_2|_{\Gamma,high} = \emptyset \qquad G_2|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$$

Then there exist G'_2 and P'_2 such that $(G'_1, P'_1) \sim^{\mathrm{wb}}_{\Gamma} (G'_2, P'_2)$ and $(G, t_2) \Longrightarrow^{\mathsf{sc}*} (G'_2, P'_2)$. Furthermore $P'_2|_{\Gamma,high} = \emptyset$ and $G'_2|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$.

Proof. We strengthen the induction hypothesis as follows. Whenever $P'_1 = t'_1 \parallel P'_{10}$ there exist $\mathcal{T}, t'_2$ and P'_{20} where $\mathcal{T} :: (G_2, t_2) \Longrightarrow^{\mathsf{tso}*} (G'_2, P'_2)$ and *FrontReapFree* $\mathcal{T}$ and $P'_2 = t'_2 \parallel P'_{20}$ and where both $t'_1 \sim^{\mathrm{wb}}_{\Gamma} t'_2$ and $P'_{10} \sim^{\mathrm{wb}}_{\Gamma} P'_{20}$. Proceed with strong induction on quantity $(size\ (t_1.cmd) + size\ (t_2.cmd))$. Invert $t_1 \sim^{\mathrm{wb}}_{\Gamma} t_1$ using Lemma 85 to get three cases. Subcases of (i) will occasionally be completed by "falling through" to (ii).

(i) $t_1 = \langle L_1, c\rangle$ and $t_2 = \langle L_2, c\rangle$ where $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$. Continue by inverting the $\longrightarrow^{commit}$ derivation.

EC-STORE: Here $c = X := y$. If $\Gamma(y) = low$, the definition of $\sim^{\mathrm{wb}}_{\Gamma}$ shows $L_1(y) = L_2(y)$ and we conclude using Lemma 72. Otherwise $\Gamma(y) = high$, inverting typing rule WB-STORE gives $\Gamma(X) = high$, and the result follows from Lemma 73.

EC-LOAD: Here $c = x := Y$. If $\Gamma(x) = high$ we conclude as updating L_1 and L_2 is not observable. If instead $\Gamma(x) = low$ then typing ensures $\Gamma(Y) = low$ and equivalences $L_1 \sim^{\mathrm{wb}}_{\Gamma} L_2$ and $G_1 \sim^{\mathrm{wb}}_{\Gamma} G_2$, ensures we're writing identical values to x.

EC-EVALEXP: Follows from Lemmas 70 and 71.

EC-SYNCACQUIRE: Here $c = \mathbf{sync}\ \ell\ \mathbf{do}\ c_0$ and $\ell \in G_1$. First suppose that $\Gamma(\ell) = high$. Let $\mathcal{E} = (L_1|_{\Gamma,low}, [\cdot])$ and note that $t_1 = \mathcal{E}[\langle L_1 \setminus L_1|_{\Gamma,low}, c\rangle]$. Inverting typing rule SC-SYNC shows that $high; \Gamma \vdash^{\mathrm{sc}} c_0$; using SC-SYNC, and noting that $L_1 \setminus L_1|_{\Gamma,low}$ has both an empty write buffer and no *low* locks, lets us find $high; \Gamma \vdash^{\mathrm{sc}} \langle L_1 \setminus L_1|_{\Gamma,low}, c\rangle$. Continue by falling through to case (ii).

Now suppose $\Gamma(\ell) = low$. Here we will use that, from inversion, $G'_1 = G_1 \setminus \{\ell\}$ and $L'_1 = L_1 \cup \{\ell\}$. By the definition of $\sim^{\mathrm{sc}}$ and fact $L_1.wb = nil$, we see that each write $(X := i)$ in $L_2.wb$ has $\Gamma(X) = high$. We can construct a derivation showing $(G_2, t_2) \Longrightarrow^{\mathsf{sc}*} (G'_{20}, \langle L'_{20}, c\rangle)$ where $(G_2, t_2) \sim^{\mathrm{sc}}_{\Gamma} (G'_{20}, \langle L'_{20}, c\rangle)$, using finitely many $\longrightarrow^{commit}$ steps, each committing *high* variables . (Again, the premises of this lemma are weak, in that we don't assume t_2's write buffer contains at most one elements.) Applying the definition of $\sim^{\mathrm{sc}}$ gives $\ell \in G'_{20}$ and $L'_{20}.wb = nil$. Taking an EC-SYNCACQUIRE step gives derivation $(G_2, t_2) \Longrightarrow^{\mathsf{sc}*} (G'_{20} \setminus \{\ell\}, \langle L'_{20} \cup \{\ell\}, c_0\rangle)$. We take this result state to be (G'_2, P'_2) and observe that $P'_2|_{\Gamma,high} = (L'_{20} \cup \{\ell\})|_{\Gamma,high} = (L_2 \cup \{\ell\})|_{\Gamma,high} = \emptyset$. It suffices to show $G'_{20} \setminus \{\ell\} \sim^{\mathrm{wb}}_{\Gamma} G'_1 \setminus \{\ell\}$, which follows from Lemmas 65 and 68, and $L'_{20} \cup \{\ell\} \sim^{\mathrm{wb}}_{\Gamma} L'_1 \cup \{\ell\}$, which follows from Lemmas 65 and 69 .

EC-FENCE, EC-FORK, EC-HOLDRELEASE: Similar to the EC-SYNCACQUIRE case.

EC-SYNCREENTER: Here $c = \mathbf{sync}\ \ell\ \mathbf{do}\ c_0$. If $\Gamma(\ell) = high$ then using an argument to similar to the EC-SYNACQUIRE case fall through to (ii). If $\Gamma(\ell) = low$ then by $\sim^{\mathrm{wb}}$ t_1 and t_2 transition in lockstep and the case is trivial.

EC-HOLDSTEP: Here $c = \mathbf{holding}\ \ell\ \mathbf{do}\ c_0$.

Suppose that $\Gamma(\ell) = high$. As in case EC-SYNCACQUIRE we fall through to case (ii).

Let $t_{10} = \langle L_1, c_0\rangle$ and $t_{20} = \langle L_2, c_0\rangle$ as well as $\mathcal{E} = (\{\ell\}, \mathbf{holding}\ \ell\ \mathbf{do}\ [\cdot])$. Inverting the evaluation relation gives $\ell \in L_1$ and $P'_1 = \mathcal{E}[t'_{10}] \parallel P'_{10}$ and $(G_1, t_{10}) \longrightarrow^{eval} (G'_1, t'_{10})$. Applying the induction hypothesis to this *eval* step, using Lemma 92 to establish $pc; \Gamma \vdash^{\mathrm{sc}} t_{10}$. This yields, among other properties,

$$\mathcal{T} :: (G_2, t_{20}) \Longrightarrow^{\mathsf{sc}*} (G'_2, t'_{20})$$

where *FrontReapFree* $\mathcal{T}$. Take P'_2 to be $\mathcal{E}[t'_{20}] \parallel P'_{20}$ and finish by applying Lemma 100. .

EC-SEQSTRUCT: Similar to EC-HOLDSTEP.

EC-SEQSKIP: Immediate.

EC-IFTRUE: Here $c = \textbf{if } b \textbf{ do } c_t \textbf{ else } c_f$ where $L_1[b] \Downarrow \textbf{true}$ and both $P'_1 = \langle L_1, c_t\rangle$ and $G'_1 = G_1$.

Suppose it's not the case that $\Gamma \vdash b : low$. Then inverting the typing relation shows both $high; \Gamma \vdash^{\text{sc}} c_t$ and $high; \Gamma \vdash^{\text{sc}} c_f$. Without loss of generality assume $L_2[b] \Downarrow \textbf{false}$ and let $(G'_2, P'_2) = (G'_2, \langle L_2, c_f\rangle)$. It suffices to show that $\langle L_1, c_t\rangle \sim^{\text{sc}}_{\Gamma} \langle L_2, c_f\rangle$. By Lemma 101 it suffices to show $high; \Gamma \vdash^{\text{wb}} L_1.wb$, which follows from hypothesis $L_1.wb = nil$.

Suppose instead that that $\Gamma \vdash b : low$. Lemma 84 shows $L_2[b] \Downarrow \textbf{true}$ so $(G_2, t_2) \overset{\mathsf{sc}*}{\Longrightarrow} (G_2, \langle L_2, c_t\rangle) \sim^{\text{wb}}_{\Gamma} (G_1, \langle L_1, c_t\rangle) = (G'_1, P'_1)$. Observe that this trace may begin with finitely many *high* commits.

EC-IFFALSE, EC-WHILETRUE, EC-WHILEFALSE: Similar to, or simpler than, EC-IFTRUE.

EC-REAP Trivial. .

(ii) $t_1 = \mathcal{E}[\langle L_1, c_1\rangle]$ where $c_1 \neq \textbf{skip}$ and $high; \Gamma \vdash^{\text{sc}} \langle L_1, c_1\rangle$ for some wt_1. By transitivity (Lemma 97) it suffices to show $(G'_1, P'_1) \sim^{\text{sc}}_{\Gamma} (G_1, t_1)$. Conclude via Lemma 103.

(iii) $t_1 = \mathcal{E}[\langle L_1, \textbf{skip}\rangle]$ and $t_2 = \mathcal{E}[\langle L_2, c_2\rangle]$. We know the following for some wt_0.

$$c_2 \neq \textbf{skip} \qquad L_1 \sim^{\text{wb}}_{\Gamma} L_2 \qquad active\ \mathcal{E} \qquad high; \Gamma \vdash^{\text{sc}} \langle L_1, c_1\rangle \qquad high; \Gamma \vdash^{\text{sc}} \langle L_2, c_2\rangle$$

By Lemma 108, we have $\mathcal{T} :: (G, \mathcal{E}[\langle L_2, c_2\rangle] \parallel \mathsf{o}) \overset{\mathsf{sc}*}{\Longrightarrow} (G'_2, \mathcal{E}[\langle L'_2, \textbf{skip}\rangle] \parallel P'_2)$ where $quiet_{\Gamma}\ \mathcal{T}$, $FrontReapFree\ \mathcal{T}$, and $L'_2|_{\Gamma,high} = P'_2|_{\Gamma,high} = \emptyset$. Furthermore $hasEmptyWBs(\mathcal{E}[\langle L'_2, \textbf{skip}\rangle] \parallel P'_2)$ so $hasEmptyWBs(P'_2)$. By Lemmas 65, 97 and 106 we find:

$$\mathcal{E}[\langle L'_2, \textbf{skip}\rangle] \sim^{\text{sc}}_{\Gamma} \mathcal{E}[\langle L_2, c_2\rangle] \sim^{\text{sc}}_{\Gamma} t_1$$

$$G'_2 \sim^{\text{wb}}_{\Gamma} G_2 \sim^{\text{wb}}_{\Gamma} G_1$$

$$P'_2 \sim^{\text{sc}}_{\Gamma} \mathsf{o}$$

Because $c_2 \neq \textbf{skip}$ it is the case that $size\,(\mathcal{E}[\langle L'_2, \textbf{skip}\rangle].cmd) < size\,(\mathcal{E}[\langle L_2, c_2\rangle].cmd)$, so we can use the induction hypothesis to find G''_2 and P''_2 such that $(G'_1, P'_1) \sim^{\text{wb}}_{\Gamma} (G''_2, P''_2)$ and $(G'_2, \mathcal{E}[\langle L'_2, \textbf{skip}\rangle]) \overset{\mathsf{tso}*}{\Longrightarrow} (G''_2, P''_2)$. By Lemma 102, $(G'_1, P'_1) \sim^{\text{wb}}_{\Gamma} (G''_2, P''_2 \parallel P'_2)$. Thus it suffices to show $(P''_2 \parallel P'_2)|_{\Gamma,high} = \emptyset$, which is immediate, and $(G_2, t_2) \overset{\mathsf{tso}*}{\Longrightarrow} (G''_2, P''_2 \parallel P'_2)$, which is a consequence of Lemma 4. ☐

Theorem 4 (SC Security)**.** *Suppose* $(G_1, P_1) \sim^{\text{sc}}_{\Gamma} (G_2, P_2)$ *and* $\overline{pc}; \Gamma \vdash^{\text{sc}} P_1$ *and wellStruct* P_1. *Suppose also that* $(G_1, P_1) \overset{\mathsf{sc}}{\Longrightarrow} (G'_1, P'_1)$. *Furthermore* $P_2|_{\Gamma,high} = \emptyset$ *and* $G_2|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$. *Then there exists* G'_2, P'_2 *such that* $(G'_1, P'_1) \sim^{\text{sc}}_{\Gamma} (G'_2, P'_2)$ *and* $(G_2, P_2) \overset{\mathsf{sc}*}{\Longrightarrow} (G'_2, P'_2)$, *and both* $P'_2|_{\Gamma,high} = \emptyset$ *and* $G'_2|_{\Gamma,high} = \textbf{Lock}|_{\Gamma,high}$.

Proof. Inverting the tso-evaluation relation and appealing to Lemma 98 gives

$$P_1 = P_{11} \parallel t_1 \parallel P_{12}$$
$$P_2 = P_{21} \parallel P^*_2 \parallel P_{22}$$
$$P'_1 = P_{11} \parallel Q'_1 \parallel P_{12}$$

where P^*_2 contains at most one thread (i.e., $P^*_2 \in \{\mathsf{o}, t_2 \parallel \mathsf{o}\}$ for some t_2) and the following hold:

$$(G, t_1) \longrightarrow^{op} (G'_1, Q'_1)$$

$$P_{11} \sim^{\text{sc}}_{\Gamma} P_{21}$$

$$t \parallel \mathsf{o} \sim^{\text{sc}}_{\Gamma} P^*_2$$

$$P_{12} \sim^{\text{sc}}_{\Gamma} P_{22}$$

It suffices to show that there exists G'_2 and Q'_2 such that $(G'_1, Q'_1) \sim^{\text{sc}}_{\Gamma} (G'_2, Q'_2)$ and $(G_2, P^*_2) \overset{\mathsf{sc}*}{\Longrightarrow} (G'_2, Q'_2)$. (Observe that while we could rename threads in Q'_2, we do not need to; thread names are only really relevant for the data-race freedom argument.) Inspecting the definition of $\sim^{\text{sc}}_{\Gamma}$ shows there are only three ways in which to find $t \parallel \mathsf{o} \sim^{\text{sc}}_{\Gamma} P^*_2$. Proceed by case analysis.

First suppose that that the equivalence arises from Definition 25, clause 2b. Here $high; \Gamma \vdash^{\text{sc}} t_1$ and via Lemma 105, $(G_1, t \parallel \mathsf{o}) \sim^{\text{sc}}_{\Gamma} (G'_1, Q'_1)$. Conclude using Lemma 97, which states $\sim^{\text{sc}}$ is an equivalence relation, and taking G_2 and P^*_2 as existential witnesses G'_2 and Q'_2.

Second suppose that that the equivalence arises from definition 25, clause 2c. Here $P_2^* = t_2 \parallel \mathfrak{o}$ where $high; \Gamma \vdash^{sc} t_2$ and $t_1 \sim^{wb}_{\Gamma} \mathfrak{o}$. From $t_1 \sim^{wb}_{\Gamma} \mathfrak{o}$ it follows that $high; \Gamma \vdash^{sc} t_1$. Again taking G_2 and P_2^* to be witnesses G_2' and Q_2' conclude with the following equational reasoning:

$$
\begin{array}{rcll}
(G', Q_1') & \sim^{sc}_{\Gamma} & (G_1, t_1 \parallel \mathfrak{o}) & \text{by Lemma 105} \\
& \sim^{sc}_{\Gamma} & (G_1, \mathfrak{o}) & \\
& \sim^{sc}_{\Gamma} & (G_2, \mathfrak{o}) & \text{by assumption} \\
& \sim^{sc}_{\Gamma} & (G_2, t_2 \parallel \mathfrak{o}) & \\
& = & (G_2, P_2^*) &
\end{array}
$$

Third suppose that that the equivalence arises from Definition 25, clause 2d. Here $P_2^* = t_2 \parallel \mathfrak{o}$ for some t_2 with $t_1 \sim^{sc}_{\Gamma} t_2$. Finitely many inversions of the typing relation show $pc; \Gamma \vdash^{sc} t_1$ for some pc. Similarly *wellStruct* t_1 and $t_2|_{\Gamma,high} = \emptyset$. Suppose the step is a commit (that is, $op = commit$); then conclude via Lemmas 109. If the step is an *eval* inverting the $\Longrightarrow^{\mathsf{sc}}$ relation shows $t_1.wb = nil$, and we conclude using Lemma 110. ☐

Corollary 8. *Suppose* $(G_1, P_1) \sim^{sc}_{\Gamma} (G_2, P_2)$ *and* $\overline{pc}; \Gamma \vdash^{sc} P_1$ *and wellStruct* P_1. *Suppose also that* $(G_1, P_1) \Longrightarrow^{\mathsf{sc}*} (G_1', P_1')$. *Furthermore* $P_2|_{\Gamma,high} = \emptyset$ *and* $G_2|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$. *Then there exist* G_2' *and* P_2' *such that* $(G_1', P_1') \sim^{sc}_{\Gamma} (G_2', P_2')$ *and* $(G_2, P_2) \Longrightarrow^{\mathsf{sc}*} (G_2', P_2')$ *and* $P_2'|_{\Gamma,high} = \emptyset$.

Proof. By finitely many application of Theorem 4 and Lemmas 5 and 96. ☐

Corollary 9. *Suppose* $G_1 \sim^{sc}_{\Gamma} G_2$ *and* $pc; \Gamma \vdash^{sc} c$ *and src* c. *Also assume* $G_2|_{\Gamma,high} = \mathbf{Lock}|_{\Gamma,high}$. *If* $(G_1, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{sc}*} (G_1', \mathfrak{o})$ *then* $(G_2, \langle L_\oslash, c\rangle) \Longrightarrow^{\mathsf{sc}*} (G_2', \mathfrak{o})$ *for some* G_2' *where* $G_1' \sim^{wb}_{\Gamma} G_2'$.

Proof. The first corollary to Theorem 4 shows (G_2, t) evaluates to a configuration related to $(G_1', \mathfrak{o})$, and preservation (Lemma 96) and Lemmas 5, 9, and 108, show this evaluates to pool $\mathfrak{o}$.

☐

Corollary 10 (SC Simple possibilistic noninterference)**.** *Suppose* $pc; \Gamma \vdash^{sc} c$ *and src* c. *Then* c *is possibilistically noninterfering under* sc *and* Γ.

7 Relating the type systems

Lemma 111. *If* $pc; \Gamma \vdash^{tso} c$ *and src* c *then there exists* wt *such that* $pc; low; \Gamma \vdash^{wb} c \Rightarrow wt$.

Proof. By induction on the derivation of $pc; \Gamma \vdash^{tso} c$.

TSO-LOAD, TSO-STORE, TSO-EVAL Follows immediately since WB-LOAD, WB-STORE, and WB-EVAL respectively have the same premises.

TSO-SYNC Then $pc = low$ and c has the form **sync** ℓ **do** c' and $\Gamma(\ell); \Gamma \vdash^{tso} c'$. Since $pc = low$ we have $pc \sqsubseteq \Gamma(\ell)$ and $pc \sqsubseteq low$. By induction there exists wt' such that $\Gamma(\ell); low; \Gamma \vdash^{wb} c' \Rightarrow wt'$ and by Lemma 57 (iii) there exists wt'' such that $\Gamma(\ell); high; \Gamma \vdash^{wb} c' \Rightarrow wt''$. Then the result follows by WB-SYNC.

TSO-HOLD Then c has the form **holding** ℓ **do** c' contradicting the premise *src* c.

TSO-FENCE Then $pc = low$ so $pc \sqsubseteq low$ and the result follows by WB-FENCE.

TSO-FORK Then $pc = low$ and c has the form **fork** c' and $pc'; \Gamma \vdash^{tso} c'$. Since $pc = low$ we have $pc \sqsubseteq low$. By induction there exists wt' such that $pc'; low; \Gamma \vdash^{wb} c' \Rightarrow wt'$ and by Lemma 57 (iii) there exists wt'' such that $low; high; \Gamma \vdash^{wb} c' \Rightarrow wt''$. Then the result follows by WB-FORK.

TSO-SEQ Then c has the form $c_1; c_2$ and $pc; \Gamma \vdash^{tso} c_1$ and $pc; \Gamma \vdash^{tso} c_2$. By induction there exist wt_1 and wt_2 such that $pc; low; \Gamma \vdash^{wb} c_1 \Rightarrow wt_1$ and $pc; low; \Gamma \vdash^{wb} c_2 \Rightarrow wt_2$. By Lemma 57 (iii) there exists wt_2' such that $pc; wt_1; \Gamma \vdash^{wb} c_2 \Rightarrow wt_2$, and the result follows by WB-SEQ.

TSO-IF Then c has the form **if** b **do** c_1 **else** c_2 and $\Gamma \vdash b : \tau$ and $pc \sqcup \tau; \Gamma \vdash^{tso} c_1$ and $pc \sqcup \tau; \Gamma \vdash^{tso} c_2$. By induction there exist wt_1 and wt_2 such that $pc \sqcup \tau; low; \Gamma \vdash^{wb} c_1 \Rightarrow wt_1$ and $pc \sqcup \tau; low; \Gamma \vdash^{wb} c_2 \Rightarrow wt_2$, and the result follows by WB-IF.

TSO-WHILE Then $pc = low$ and c has the form **while** b **do** c' and $\Gamma \vdash b : low$ and $pc'; \Gamma \vdash^{\text{tso}} c'$. By induction there exists wt' such that $pc'; low; \Gamma \vdash^{\text{wb}} c' \Rightarrow wt'$, and the result follows by WB-WHILE.

TSO-SKIP Then c has the form **skip** and the result follows by WB-SKIP. □

Lemma 112. *If* $pc; wt; \Gamma \vdash^{\text{wb}} c \Rightarrow ut$ *then* $pc; \Gamma \vdash^{\text{sc}} c$.

Proof. by induction on the structure of the $\vdash^{\text{wb}}$ judgment.

WB-LOAD, WB-STORE, WB-EVAL WB-FENCE, WB-SKIP: Immediate as the corresponding SC-* rules have the same or fewer premises.

WB-SYNC: Here $c = \textbf{sync}\ \ell\ \textbf{do}\ c_0$ and $pc \sqsubseteq \Gamma(\ell)$ by the induction hypothesis $\Gamma(\ell); \Gamma \vdash^{\text{sc}} c_0$. The result follows from SC-SYNC.

WB-HOLD: Similar to WB-SYNC.

WB-FORK: Here $c = \textbf{fork}\ c_0$ and induction gives $pc; \Gamma \vdash^{\text{sc}} c_0$. The result follows from SC-FORK.

WB-SEQ: Here $c = c_1; c_2$ and the induction hypothesis gives $pc; \Gamma \vdash^{\text{sc}} c_1$ and $pc; \Gamma \vdash^{\text{sc}} c_2$. The result follows from SC-SEQ.

WB-IF: Here $c = \textbf{if}\ b\ \textbf{do}\ c_1\ \textbf{else}\ c_2$ where $\Gamma \vdash b : \tau$ and the induction hypothesis gives $pc \sqcup \tau; \Gamma \vdash^{\text{sc}} c_1$ and $pc \sqcup \tau; \Gamma \vdash^{\text{sc}} c_2$. Conclude with rule SC-IF.

WB-WHILE: Here $c = \textbf{while}\ b\ \textbf{do}\ c_0$. Inverting the typing relation shows $pc = low$ and $\Gamma \vdash b : low$. Furthermore, for some pc_0 and ut_0, it is the case that $pc_0; ut; \Gamma \vdash^{\text{wb}} c_0 \Rightarrow ut_0$. By induction $pc_0; \Gamma \vdash^{\text{sc}} c_0$. Conclude with rule SC-WHILE. □

References

Geoffrey Smith and Dennis Volpano. Secure information flow in a multi-threaded imperative language. In *Proc. 25th ACM Symp. on Principles of Programming Languages (POPL)*, pages 355–364, San Diego, California, January 1998.

Jefrey A. Vaughan and Todd Millstein. Secure information flow for concurrent programs under total store order. In *CSF '12*, 2012.