# *Transactional Events: Technical Appendix*∗

KEVIN DONNELLY
*Boston University, Boston, MA 02215, USA* (*e-mail:* `kevind@cs.bu.edu`)

MATTHEW FLUET
*Toyota Technological Institute at Chicago, Chicago, IL 60637, USA* (*e-mail:* `fluet@tti-c.org`)

## A Equivalence of the Original and Refined Semantics

In this appendix, we wish show that the refined semantics given in Section 6.2 is equivalent to the semantics given in Section 4, in the sense that programs evaluated under the two semantics have the same observable behavior.

### *A.1 Refined semantics simulates original semantics*

In one direction, we require that every observable behavior admitted by the semantics of Section 4 is also admitted by the semantics of Section 6.2 (Theorem 9).

First, we show that the synchronous evaluation relation of Section 4.2.2 may be simulated by the evaluation of (committable) search threads.

*Lemma 11*
If $\{\langle\theta_1, e_1\rangle, \ldots, \langle\theta_k, e_k\rangle\} \rightsquigarrow \{\langle\theta_1, e'_1\rangle, \ldots, \langle\theta_k, e'_k\rangle\}$ according to the semantics of Figure 3, the set of trails $\{\langle\theta_1, \rho_1\rangle, \ldots, \langle\theta_k, \rho_k\rangle\}$ is committable, and $\mathcal{P}$ is a set of search threads, each with a thread identifier in $\{\theta_1, \ldots, \theta_k\}$,
then $\{\langle\theta_1, e_1, \rho_1\rangle, \ldots, \langle\theta_k, e_k, \rho_k\rangle\} \uplus \mathcal{P} \xrightarrow{\epsilon} \{\langle\theta_1, e'_1, \rho'_1\rangle, \ldots, \langle\theta_k, e'_k, \rho'_k\rangle\} \uplus \mathcal{P}'$ according to the semantics of Figure 9, the set of trails $\{\langle\theta_1, \rho'_1\rangle, \ldots, \langle\theta_k, \rho'_k\rangle\}$ is committable, and $\mathcal{P}'$ is a set of search threads, each with a thread identifier in $\{\theta_1, \ldots, \theta_k\}$.

*Proof*
The lemma follows by case analysis on the synchronous evaluation relation.

For an EvtEval, EvtThenAlways, EvtThenThrow, EvtCatchAlways, EvtCatchThrow, EvtNewSChan, or EvtMyThreadId transition, we take the corresponding transition in the refined semantics (with $\mathcal{P}' = \mathcal{P}$). Note that each of these transitions leaves the trails unchanged; hence, the set of trails remains committable.

For an EvtChooseLeft transition of the form

$$\{\ldots, \langle\theta_i, M_i^{Evt}[\texttt{choosEvt}\ e_l\ e_r]\rangle, \ldots\} \rightsquigarrow \{\ldots, \langle\theta_i, M_i^{Evt}[e_l]\rangle, \ldots\}$$

∗ This technical appendix supplements the paper (Donnelly & Fluet, 2008).

we take an EvtChoose transition in the refined semantics of the form

$$\begin{aligned}&\{\ldots,\langle\theta_i, M_i^{Evt}[\texttt{choosEvt}\ e_l\ e_r],\rho_i\rangle,\ldots\}\uplus\mathcal{P}\\ &\quad\xrightarrow{\epsilon}\{\ldots,\langle\theta_i, M_i^{Evt}[e_l],\mathsf{Left}{:}\rho_i\rangle,\ldots\}\uplus\underbrace{(\mathcal{P}\cup\{\langle\theta_i, M_i^{Evt}[e_r],\mathsf{Right}{:}\rho_i\rangle\})}_{\mathcal{P}'}\end{aligned}$$

Note that $\mathsf{Left}{:}\rho_i \succeq \rho_i$ and $\mathsf{Dep}(\langle\theta_i,\mathsf{Left}{:}\rho_i\rangle) = \{\langle\theta_i,\mathsf{Left}{:}\rho_i\rangle\}\cup\mathsf{Dep}(\langle\theta_i,\rho_i\rangle)$ and that this transition leaves trails other than $\langle\theta_i,\rho_i\rangle$ unchanged; hence, the set of trails is committable. An EvtChooseRight transition is handled in a symmetric manner.

For an EvtSendRecv transition of the form

$$\begin{aligned}&\{\ldots,\langle\theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e]\rangle,\langle\theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa]\rangle,\ldots\}\\ &\quad\rightsquigarrow\{\ldots,\langle\theta_s, M_s^{Evt}[\texttt{alwaysEvt}\ ()]\rangle,\langle\theta_r, M_r^{Evt}[\texttt{alwaysEvt}\ e]\rangle,\ldots\}]\end{aligned}$$

we take an EvtSendRecv transition in the refined semantics of the form

$$\frac{\mathsf{Coherent}(\langle\theta_s,\rho_s\rangle,\langle\theta_r,\rho_r\rangle)}{\begin{aligned}&\{\ldots,\langle\theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e],\rho_s\rangle,\langle\theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa],\rho_r\rangle,\ldots\}\uplus\mathcal{P}\\ &\quad\xrightarrow{\epsilon}\{\ldots,\langle\theta_s, M_s^{Evt}[\texttt{alwaysEvt}\ ()],\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s\rangle,\\ &\qquad\qquad\langle\theta_r, M_r^{Evt}[\texttt{alwaysEvt}\ e],\mathsf{Recv}(\langle\theta_s,\rho_s)\rangle{:}\rho_r\rangle,\ldots\}\\ &\qquad\uplus\underbrace{(\mathcal{P}\cup\{\langle\theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e],\rho_s\rangle,\langle\theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa],\rho_r\rangle\})}_{\mathcal{P}'}\end{aligned}}$$

By Lemma 6, the committability of the trails $\{\langle\theta_1,\rho_1\rangle,\ldots,\langle\theta_k,\rho_k\rangle\}$ implies the coherency of the trails $\langle\theta_s,\rho_s\rangle$ and $\langle\theta_r,\rho_r\rangle$. Note that $\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s \succeq \rho_s$, $\mathsf{Recv}(\langle\theta_s,\rho_s\rangle){:}\rho_r \succeq \rho_r$, $\mathsf{Dep}(\langle\theta_s,\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s\rangle) = \{\langle\theta_s,\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s\rangle\}\cup\{\langle\theta_r,\mathsf{Recv}(\langle\theta_s,\rho_s\rangle){:}\rho_r\rangle\}\cup\mathsf{Dep}(\langle\theta_s,\rho_s\rangle)\cup\mathsf{Dep}(\langle\theta_r,\rho_r\rangle)$, and $\mathsf{Dep}(\langle\theta_r,\mathsf{Recv}(\langle\theta_s,\rho_s\rangle){:}\rho_r\rangle) = \{\langle\theta_r,\mathsf{Recv}(\langle\theta_s,\rho_s\rangle){:}\rho_r\rangle\}\cup\{\langle\theta_s,\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s\rangle\}\cup\mathsf{Dep}(\langle\theta_r,\rho_r\rangle)\cup\mathsf{Dep}(\langle\theta_s,\rho_s\rangle)$ and that this transition leaves trails other than $\langle\theta_s,\rho_s\rangle$ and $\langle\theta_r,\rho_r\rangle$ unchanged; hence, the set of trails is committable. □

This lemma immediately generalizes to the synchronous evaluation of a set of events to a terminal configuration.

*Corollary 12*
If $\{\langle\theta_1,e_1\rangle,\ldots,\langle\theta_k,e_k\rangle\} \rightsquigarrow^n \{\langle\theta_1,\texttt{alwaysEvt}\ e'_1\rangle,\ldots,\langle\theta_k,\texttt{alwaysEvt}\ e'_k\rangle\}$ according to the semantics of Figure 3,
then $\{\langle\theta_1,e_1,\bullet\rangle,\ \ldots,\ \langle\theta_k,e_k,\bullet\rangle\} \xrightarrow{\epsilon;\cdots;\epsilon}{}^n \{\langle\theta_1,\texttt{alwaysEvt}\ e'_1,\rho'_1\rangle,\ \ldots,\ \langle\theta_k,\texttt{alwaysEvt}\ e'_k,\rho'_k\rangle\}\uplus\mathcal{P}'$ according to the semantics of Figure 9, the set of trails $\{\langle\theta_1,\rho'_1\rangle,\ldots,\langle\theta_k,\rho'_k\rangle\}$ is committable, and $\mathcal{P}'$ is a set of search threads, each with a thread identifier in $\{\theta_1,\ldots,\theta_k\}$.

*Proof*
The corollary follows by induction on $n$. For the base case, note that the set of trails $\{\langle\theta_1,\bullet\rangle,\ldots,\langle\theta_k,\bullet\rangle\}$ is committable (with $\mathcal{P}' = \{\}$). For the inductive case, apply Lemma 11. □

Finally, we show that the concurrent evaluation relation of Section 4.2.3 may be simulated by refined semantics.

*Theorem 13*

If $\mathcal{T} \xrightarrow{a} \mathcal{T}'$ according to the semantics of Figures 3 and 4,
then either $\mathcal{T} \xrightarrow{a} \mathcal{T}'$ or $\mathcal{T} \xrightarrow{\epsilon;\cdots;\epsilon}{}^{*} \mathcal{T}'$ (and $a = \epsilon$) according to the semantics of Figure 9.

*Proof*

The theorem follows by case analysis on the concurrent evaluation relation.

For an IOEval, IOBindUnit, IOBindThrow, IOCatchUnit, IOCatchThrow, IOGetChar, IOPutChar, IOFork, or IOMyThreadId transition, we take the corresponding transition in the refined semantics. Note that on these transitions, the semantics of Figure 4 and Figure 9 coincide.

For an IOSync transition of the form

$$\frac{\{\langle\theta_1, e_1\rangle, \ldots, \langle\theta_k, e_k\rangle\} \rightsquigarrow^{*} \{\langle\theta_1, \texttt{alwaysEvt}\ e'_1\rangle, \ldots, \langle\theta_k, \texttt{alwaysEvt}\ e'_k\rangle\}}{\begin{array}{l}\mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}[\texttt{sync}\ e_1]\rangle, \ldots, \langle\theta_k, M_k^{IO}[\texttt{sync}\ e_k]\rangle\} \\ \quad \xrightarrow{\epsilon} \mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}[\texttt{unitIO}\ e'_1]\rangle, \ldots, \langle\theta_k, M_k^{IO}[\texttt{unitIO}\ e'_k]\rangle\}\end{array}}$$

we construct a sequence of $\epsilon$-transitions as follows. First, we have a sequence of $k$ SyncInit transitions for each of the synchronizing threads:

$$\begin{array}{l}\mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}[\texttt{sync}\ e_1]\rangle, \ldots, \langle\theta_k, M_k^{IO}[\texttt{sync}\ e_k]\rangle\} \\ \quad \xrightarrow{\epsilon} \mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \langle\theta_1, e_1, \bullet\rangle, \ldots, \langle\theta_k, M_k^{IO}[\texttt{sync}\ e_k]\rangle\} \\ \quad \ldots \\ \quad \xrightarrow{\epsilon} \mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \langle\theta_1, e_1, \bullet\rangle, \ldots, \langle\theta_k, M_k^{IO}, e_k\rangle, \langle\theta_k, e_k, \bullet\rangle\} \\ \quad \equiv \mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \ldots, \langle\theta_k, M_k^{IO}, e_k\rangle\} \uplus \{\langle\theta_1, e_1, \bullet\rangle, \ldots, \langle\theta_k, e_k, \bullet\rangle\}\end{array}$$

Applying Corollary 12, we have a sequence of transitions implied by the synchronous evaluation relation:

$$\begin{array}{l}\mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \ldots, \langle\theta_k, M_k^{IO}, e_k\rangle\} \uplus \{\langle\theta_1, e_1, \bullet\rangle, \ldots, \langle\theta_k, e_k, \bullet\rangle\} \\ \quad \xrightarrow{\epsilon;\cdots;\epsilon}{}^{*} \mathcal{T} \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \ldots, \langle\theta_k, M_k^{IO}, e_k\rangle\} \\ \qquad\qquad \uplus \{\langle\theta_1, \texttt{alwaysEvt}\ e'_1, \rho'_1\rangle, \ldots, \langle\theta_k, \texttt{alwaysEvt}\ e'_k, \rho'_k\rangle\} \uplus \mathcal{P}' \\ \quad \equiv \mathcal{T} \uplus \mathcal{P}' \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \langle\theta_1, \texttt{alwaysEvt}\ e'_1, \rho'_1\rangle, \ldots, \\ \qquad\qquad \langle\theta_k, M_k^{IO}, e_k\rangle, \langle\theta_k, \texttt{alwaysEvt}\ e'_k, \rho'_k\rangle\}\end{array}$$

where the set of trails $\{\langle\theta_1, \rho'_1\rangle, \ldots, \langle\theta_k, \rho'_k\rangle\}$ is committable and $\mathcal{P}'$ is a set of search threads, each with a thread identifier in $\{\theta_1, \ldots, \theta_k\}$. Finally, we have a SyncCommit transition of the form:

$$\frac{\mathsf{Committable}(\{\langle\theta_1, \rho'_1\rangle, \ldots, \langle\theta_k, \rho'_k\rangle\})}{\begin{array}{l}\mathcal{T} \uplus \mathcal{P}' \uplus \{\langle\theta_1, M_1^{IO}, e_1\rangle, \langle\theta_1, \texttt{alwaysEvt}\ e'_1, \rho'_1\rangle, \ldots, \\ \qquad \langle\theta_k, M_k^{IO}, e_k\rangle, \langle\theta_k, \texttt{alwaysEvt}\ e'_k, \rho'_k\rangle\} \\ \quad \xrightarrow{\epsilon} (\mathcal{T} \uplus \mathcal{P}')\backslash_{\{\theta_1,\ldots,\theta_k\}} \uplus \{\langle\theta_1, M_1^{IO}[\texttt{unitIO}\ e'_1]\rangle, \ldots, \langle\theta_k, M_k^{IO}[\texttt{unitIO}\ e'_k]\rangle\}\end{array}}$$

Note that $(\mathcal{T} \uplus \mathcal{P}')\backslash_{\{\theta_1,\ldots,\theta_k\}} \equiv \mathcal{T}$, since $\mathcal{T}$ is a set of concurrent threads, each with a thread identifier not in $\{\theta_1, \ldots, \theta_k\}$, and $\mathcal{P}'$ is a set of search threads, each with

a thread identifier in $\{\theta_1, \ldots, \theta_k\}$. Hence, we take the sequence of $\epsilon$-transitions:

$$\begin{aligned}
&\mathcal{T} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{sync}\ e_1]\rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{sync}\ e_k]\rangle\} \\
&\quad \xrightarrow{\epsilon;\cdots;\epsilon}{}^{*}\ (\mathcal{T} \uplus \mathcal{P}')\backslash_{\{\theta_1,\ldots,\theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1']\rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k']\rangle\} \\
&\quad \equiv \mathcal{T} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1']\rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k']\rangle\} \quad \square
\end{aligned}$$

An immediate consequence is the simulation theorem stated in Section 6.2.1.

*Theorem 9*
If $\mathcal{T} \xrightarrow{a_1;\cdots;a_n}{}^{*} \mathcal{T}'$ according to the semantics of Figures 3 and 4,
then $\mathcal{T} \xrightarrow{a_1';\cdots;a_m'}{}^{*} \mathcal{T}'$ according to the semantics of Figure 9,
where $a_1;\cdots;a_n$ equals $a_1';\cdots;a_m'$ modulo the insertion of $\epsilon$ actions.

*Proof*
The theorem follows by induction on the concurrent evaluation relation, applying Theorem 13. $\square$

### *A.2 Original semantics simulates refined semantics*

In the other direction, we require that every behavior admitted by the semantics of Section 6.2 is also admitted by the semantics of Section 4 (Theorem 10). For completeness, we recall the simple translation from the thread soups of Section 6.2.1 to the thread soups of Section 4:

$$\begin{aligned}
|\{\}| &= \{\} \\
|\mathcal{P} \uplus \{\langle \theta, e\rangle\}| &= |\mathcal{P}| \uplus \{\langle \theta, e\rangle\} \\
|\mathcal{P} \uplus \{\langle \theta, M^{IO}, e\rangle\}| &= |\mathcal{P}| \uplus \{\langle \theta, M^{IO}[\texttt{sync}\ e]\rangle\} \\
|\mathcal{P} \uplus \{\langle \theta, e, \rho\rangle\}| &= |\mathcal{P}|
\end{aligned}$$

Note that this translation simply erases any search thread and translates a suspended thread to a synchronizing concurrent thread.

As was required in the previous section, we need to relate the evaluation of committable search threads to the synchronous evaluation relation of Section 4.2.2. In order to state lemmas and theorems concisely, we introduce a predicate asserting the fact that, in a thread soup $\mathcal{P}$, every committable subset of search threads may be simulated by a sequence of synchronous evaluation relation steps:

*Definition 6* (*Synchronously Simulable*)
The thread soup $\mathcal{P}$ is *synchronously simulable* if for every committable subset of search threads and corresponding subset of suspended threads, there is a sequence of synchronous evaluation relation steps that take the suspended thread expressions to the search thread expressions. Formally,

$$\begin{aligned}
&\mathsf{SyncSim}(\mathcal{P}) \equiv \\
&\quad \forall \{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\} \subseteq \mathcal{P}. \\
&\quad \mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_k, \rho_k\rangle\}) \\
&\qquad \Rightarrow \{\langle \theta_1, e_1\rangle, \ldots, \langle \theta_k, e_k\rangle\} \rightsquigarrow^{*} \{\langle \theta_1, e_1'\rangle, \ldots, \langle \theta_k, e_k'\rangle\}
\end{aligned}$$

Lemma 15 will show that the refined semantics of Section 6.2 preserves the SyncSim predicate. In that proof, we will need to relate the committability of a set of trails after an "interesting" transition (i.e., SyncInit, EvtChoose, and EvtSendRecv transitions) to the committability of a corresponding set of trails before the transition.

*Lemma 14*

1. If $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_{i-1},\rho_{i-1}\rangle, \langle\theta_i,\bullet\rangle, \langle\theta_{i+1},\rho_{i+1}\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$, then $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_{i-1},\rho_{i-1}\rangle, \langle\theta_{i+1},\rho_{i+1}\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$.
2. If $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_i,\mathsf{Left}{:}\rho_i\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$, then $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_i,\rho_i\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$.
3. If $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_i,\mathsf{Right}{:}\rho_i\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$, then $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_i,\rho_i\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$.
4. If $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_s,\mathsf{Send}(\langle\theta_r,\rho_r\rangle){:}\rho_s\rangle, \langle\theta_r,\mathsf{Recv}(\langle\theta_s,\rho_s\rangle){:}\rho_r\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$, then $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_s,\rho_s\rangle, \langle\theta_r,\rho_r\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$.

*Proof*

The lemma follows from various properties of committablity, consistency, dependencies, and extension. □

We are now prepared to show that the refined semantics of Section 6.2 preserves the SyncSim predicate.

*Lemma 15*

If $\mathsf{SyncSim}(\mathcal{P})$ and $\mathcal{P} \xrightarrow{a} \mathcal{P}'$ according to the semantics of Figure 9,
then $\mathsf{SyncSim}(\mathcal{P}')$.

*Proof*

The lemma follows by case analysis on the concurrent evaluation relation.

For an IOEval, IOBindUnit, IOBindThrow, IOCatchUnit, IOCatchThrow, IOGetChar, IOPutChar, IOFork, or IOMyThreadId transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}[e_i^\dagger]\rangle\} \xrightarrow{a} \mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}[e_i^\ddagger]\rangle\}$$

(for transition appropriate $e_i^\dagger$ and $e_i^\ddagger$). Note that the set of suspended and search threads in the thread soup $\mathcal{P} \uplus \langle\theta_i, M_i^{IO}[e_i^\ddagger]\rangle$ is equal to that in $\mathcal{P} \uplus \langle\theta_i, M_i^{IO}[e_i^\dagger]\rangle$. Hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \langle\theta_i, M_i^{IO}[e_i^\dagger]\rangle)$ implies $\mathsf{SyncSim}(\mathcal{P} \uplus \langle\theta_i, M_i^{IO}[e_i^\ddagger]\rangle)$.

For a SyncInit transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}[\texttt{sync}\ e_i]\rangle\} \xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}, e_i\rangle, \langle\theta_i, e_i, \bullet\rangle\}$$

We must show that $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}, e_i\rangle, \langle\theta_i, e_i, \bullet\rangle\})$. Consider arbitrary $\{\langle\theta_1, M_1^{IO}, e_1\rangle, \langle\theta_1, e_1', \rho_1\rangle, \ldots, \langle\theta_k, M_k^{IO}, e_k\rangle, \langle\theta_k, e_k', \rho_k\rangle\} \subseteq \mathcal{P} \uplus \{\langle\theta_i, M_i^{IO}, e_i\rangle, \langle\theta_i, e_i, \bullet\rangle\}$ such that $\mathsf{Committable}(\{\langle\theta_1,\rho_1\rangle, \ldots, \langle\theta_k,\rho_k\rangle\})$. There are two cases: (1) $\langle\theta_i, e_i, \bullet\rangle \notin \{\langle\theta_1, e_1', \rho_1\rangle, \ldots, \langle\theta_k, e_k', \rho_k\rangle\}$, and (2) $\langle\theta_i, e_i, \bullet\rangle \in \{\langle\theta_1, e_1', \rho_1\rangle, \ldots, \langle\theta_k, e_k', \rho_k\rangle\}$.

1. If $\langle \theta_i, e_i, \bullet \rangle \notin \{\langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, e'_k, \rho_k \rangle\}$, then $\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e'_k, \rho_k \rangle\} \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\}$; hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\})$ and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$ implies that there is a sequence of transitions

$$\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_k, e_k \rangle\} \rightsquigarrow^* \{\langle \theta_1, e'_1 \rangle, \ldots, \langle \theta_k, e'_k \rangle\}$$

2. If $\langle \theta_i, e_i, \bullet \rangle \in \{\langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, e'_k, \rho_k \rangle\}$, then

$$\begin{aligned}
&\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e'_k, \rho_k \rangle\} \\
&\quad \equiv \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_{i-1}, M_{i-1}^{IO}, e_{i-1} \rangle, \langle \theta_{i-1}, e'_{i-1}, \rho_{i-1} \rangle, \\
&\qquad \langle \theta_i, M_i^{IO}, e_i \rangle, \langle \theta_i, e_i, \bullet \rangle, \\
&\qquad \langle \theta_{i+1}, M_{i+1}^{IO}, e_{i+1} \rangle, \langle \theta_{i+1}, e'_{i+1}, \rho_{i+1} \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e'_k, \rho_k \rangle\}
\end{aligned}$$

Note that

$$\begin{aligned}
&\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e'_1, \rho_1 \rangle, \ldots, \langle \theta_{i-1}, M_{i-1}^{IO}, e_{i-1} \rangle, \langle \theta_{i-1}, e'_{i-1}, \rho_{i-1} \rangle, \\
&\quad \langle \theta_{i+1}, M_{i+1}^{IO}, e_{i+1} \rangle, \langle \theta_{i+1}, e'_{i+1}, \rho_{i+1} \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e'_k, \rho_k \rangle\} \\
&\qquad \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\}
\end{aligned}$$

and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_{i-1}, \rho_{i-1} \rangle, \langle \theta_{i+1}, \rho_{i+1} \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$ (which follows from Lemma 14.1 and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_{i-1}, \rho_{i-1} \rangle, \langle \theta_i, \bullet \rangle, \langle \theta_{i+1}, \rho_{i+1} \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$); hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ \mathtt{e_i}] \rangle\})$ implies that there is a sequence of transitions

$$\begin{aligned}
&\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_{i-1}, e_{i-1} \rangle, \langle \theta_{i+1}, e_{i+1} \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\
&\quad \rightsquigarrow^* \{\langle \theta_1, e'_1 \rangle, \ldots, \langle \theta_{i-1}, e'_{i-1} \rangle, \langle \theta_{i+1}, e'_{i+1} \rangle, \ldots, \langle \theta_k, e'_k \rangle\}
\end{aligned}$$

We may uniformly add $\{\langle \theta_i, e_i \rangle\}$ to this sequence of transitions, yielding

$$\begin{aligned}
&\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_{i-1}, e_{i-1} \rangle, \langle \theta_i, e_i \rangle, \langle \theta_{i+1}, e_{i+1} \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\
&\quad \rightsquigarrow^* \{\langle \theta_1, e'_1 \rangle, \ldots, \langle \theta_{i-1}, e'_{i-1} \rangle, \langle \theta_i, e_i \rangle, \langle \theta_{i+1}, e'_{i+1} \rangle, \ldots, \langle \theta_k, e'_k \rangle\}
\end{aligned}$$

For a SyncCommit transition, the transition is of the form

$$\frac{\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})}{\begin{aligned}
&\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e'_1, \rho_1 \rangle, \ldots, \\
&\qquad \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e'_k, \rho_k \rangle\} \\
&\quad \xrightarrow{\epsilon} \mathcal{P}\backslash_{\{\theta_1, \ldots, \theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e'_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e'_k] \rangle\}
\end{aligned}}$$

Note that the set of suspended and search threads in the thread soup $\mathcal{P}\backslash_{\{\theta_1, \ldots, \theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e'_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e'_k] \rangle\}$ is a strict subset of that in $\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e'_k, \rho_k \rangle\}$. Hence $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e'_1, \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e'_k, \rho_k \rangle\})$ implies $\mathsf{SyncSim}(\mathcal{P}\backslash_{\{\theta_1, \ldots, \theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e'_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e'_k] \rangle\})$.

For an EvtEval, EvtThenAlways, EvtThenThrow, EvtCatchAlways, EvtCatchThrow, EvtNewSChan, or EvtMyThreadId transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[e_i^{\dagger}], \rho_i \rangle\} \xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[e_i^{\ddagger}], \rho_i \rangle\}$$

(for transition appropriate $e_i^\dagger$ and $e_i^\ddagger$). We must show that $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\ddagger], \rho_i \rangle\})$. Consider arbitrary $\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\ddagger], \rho_i \rangle\}$ such that $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$. There are two cases: (1) $\langle \theta_i, M^{Evt}[e_i^\ddagger], \rho_i \rangle \notin \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$, and (2) $\langle \theta_i, M^{Evt}[e_i^\ddagger], \rho_i \rangle \in \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$.

1. If $\langle \theta_i, M^{Evt}[e_i^\ddagger], \rho_i \rangle \notin \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$, then $\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\dagger], \rho_i \rangle\}$; hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\dagger], \rho_i \rangle\})$ and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$ implies that there is a sequence of transitions

$$\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_k, e_k \rangle\} \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_k, e_k' \rangle\}$$

2. If $\langle \theta_i, M^{Evt}[e_i^\ddagger], \rho_i \rangle \in \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$, then

$$\begin{aligned} &\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \\ &\quad \equiv \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \\ &\qquad \langle \theta_i, M_i^{IO}, e_i \rangle, \langle \theta_i, M_i^{Evt}[e_i^\ddagger], \rho_i \rangle, \ldots, \\ &\qquad \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \end{aligned}$$

Note that

$$\begin{aligned} &\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \\ &\ \langle \theta_i, M_i^{IO}, e_i \rangle, \langle \theta_i, M_i^{Evt}[e_i^\dagger], \rho_i \rangle, \ldots, \\ &\ \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \\ &\qquad \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\dagger], \rho_i \rangle\} \end{aligned}$$

and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_i, \rho_i \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$; hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^\dagger], \rho_i \rangle\})$ implies that there is a sequence of transitions

$$\begin{aligned} &\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_i, e_i \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\ &\quad \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_i, M^{Evt}[e_i^\dagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \end{aligned}$$

We take the corresponding transition in the original semantics (of the synchronous evaluation relation), yielding

$$\begin{aligned} &\{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_i, M^{Evt}[e_i^\dagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \\ &\quad \rightsquigarrow \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_i, M^{Evt}[e_i^\ddagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \end{aligned}$$

Hence,

$$\begin{aligned} &\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_i, e_i \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\ &\quad \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_i, M^{Evt}[e_i^\ddagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \end{aligned}$$

For an EvtNever transition, the transition is of the form

$$\frac{}{\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[\texttt{neverEvt}], \rho_i \rangle\} \xrightarrow{\epsilon} \mathcal{P}}$$

Note that the set of suspended and search threads in the thread soup $\mathcal{P}$ is a strict subset of that in $\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{neverEvt}], \rho_i \rangle\}$. Hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{neverEvt}], \rho_i \rangle\})$ implies $\mathsf{SyncSim}(\mathcal{P})$.

For an EvtChoose transition, the transition is of the form

$$\begin{aligned}&\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\}\\ &\quad \xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle\}\end{aligned}$$

We must show that $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle\})$. Consider arbitrary $\{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\} \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle\}$ such that $\mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_k, \rho_k\rangle\})$. There are three cases: (1) $\langle \theta_i, M^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle \notin \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\} \wedge \langle \theta_i, M^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle \notin \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$, (2) $\langle \theta_i, M^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle \in \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$, and (3) $\langle \theta_i, M^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle \in \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$. (The case $\langle \theta_i, M^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle \in \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\} \wedge \langle \theta_i, M^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle \in \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$ is impossible, because $\mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_k, \rho_k\rangle\})$ and a committable set requires distinct thread identifiers.)

1. If $\langle \theta_i, M^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle \notin \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$ and $\langle \theta_i, M^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle \notin \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$, then $\{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\} \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\}$; hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\})$ and $\mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_k, \rho_k\rangle\})$ implies that there is a sequence of transitions

$$\{\langle \theta_1, e_1\rangle, \ldots, \langle \theta_k, e_k\rangle\} \rightsquigarrow^* \{\langle \theta_1, e_1'\rangle, \ldots, \langle \theta_k, e_k'\rangle\}$$

2. If $\langle \theta_i, M^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle \in \{\langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, e_k', \rho_k\rangle\}$, then

$$\begin{aligned}&\{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\}\\ &\quad \equiv \{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots,\\ &\qquad \langle \theta_i, M_i^{IO}, e_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \ldots,\\ &\qquad \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\}\end{aligned}$$

Note that

$$\begin{aligned}&\{\langle \theta_1, M_1^{IO}, e_1\rangle, \langle \theta_1, e_1', \rho_1\rangle, \ldots,\\ &\ \langle \theta_i, M_i^{IO}, e_i\rangle, \langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle, \ldots,\\ &\ \langle \theta_k, M_k^{IO}, e_k\rangle, \langle \theta_k, e_k', \rho_k\rangle\}\\ &\qquad \subseteq \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\}\end{aligned}$$

and $\mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_i, \rho_i\rangle, \ldots, \langle \theta_k, \rho_k\rangle\})$ (which follows from Lemma 14.2 and $\mathsf{Committable}(\{\langle \theta_1, \rho_1\rangle, \ldots, \langle \theta_i, \mathsf{Left}{:}\rho_i\rangle, \ldots, \langle \theta_k, \rho_k\rangle\})$); hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\})$ implies that there is a sequence of transitions

$$\begin{aligned}&\{\langle \theta_1, e_1\rangle, \ldots, \langle \theta_i, e_i\rangle, \ldots, \langle \theta_k, e_k\rangle\}\\ &\quad \rightsquigarrow^* \{\langle \theta_1, e_1'\rangle, \ldots, \langle \theta_i, M^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r]\rangle, \ldots, \langle \theta_k, e_k'\rangle\}\end{aligned}$$

We take an EvtChooseLeft transition in the original semantics (of the synchronous evaluation relation), yielding

$$\begin{aligned}&\{\langle \theta_1, e_1'\rangle, \ldots, \langle \theta_i, M^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r]\rangle, \ldots, \langle \theta_k, e_k'\rangle\}\\ &\quad \rightsquigarrow \{\langle \theta_1, e_1'\rangle, \ldots, \langle \theta_i, M^{Evt}[e_i^l]\rangle, \ldots, \langle \theta_k, e_k'\rangle\}\end{aligned}$$

Hence,

$$\{\langle\theta_1, e_1\rangle, \ldots, \langle\theta_i, e_i\rangle, \ldots, \langle\theta_k, e_k\rangle\} \\ \rightsquigarrow^* \{\langle\theta_1, e'_1\rangle, \ldots, \langle\theta_i, M^{Evt}[e^l_i]\rangle, \ldots, \langle\theta_k, e'_k\rangle\}$$

3. The case $\langle\theta_i, M^{Evt}[e^r_i], \mathsf{Right}{:}\rho_i\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$ is similar to the case $\langle\theta_i, M^{Evt}[e^l_i], \mathsf{Left}{:}\rho_i\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$.

For a EvtThrow transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle\theta_i, \texttt{throwEvt}\ e^\dagger_i, \rho_i\rangle\} \xrightarrow{\epsilon} \mathcal{P}$$

Note that the set of suspended and search threads in the thread soup $\mathcal{P}$ is a strict subset of that in $\mathcal{P} \uplus \{\langle\theta_i, \texttt{throwEvt}\ e^\dagger_i, \rho_i\rangle\}$. Hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle\theta_i, \texttt{throwEvt}\ e^\dagger_i, \rho_i\rangle\})$ implies $\mathsf{SyncSim}(\mathcal{P})$.

For an EvtSendRecv transition, the transition is of the form

$$\frac{\mathsf{Coherent}(\langle\theta_s, \rho_s\rangle, \langle\theta_r, \rho_r\rangle)}{\begin{array}{l}\mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle\} \\ \quad \xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle, \\ \qquad\qquad \langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle, \\ \qquad\qquad \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle\}\end{array}}$$

We must show that $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle, \langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle\})$. Consider arbitrary $\{\langle\theta_1, M^{IO}_1, e_1\rangle, \langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, M^{IO}_k, e_k\rangle, \langle\theta_k, e'_k, \rho_k\rangle\} \subseteq \mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle, \langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle\}$ such that $\mathsf{Committable}(\{\langle\theta_1, \rho_1\rangle, \ldots, \langle\theta_k, \rho_k\rangle\})$. There are two cases: (1) $\langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\} \wedge \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$, (2) $\langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\} \wedge \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$. (The cases $\langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\} \wedge \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$ and $\langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle \in \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\} \wedge \langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$ are impossible, because $\mathsf{Committable}(\{\langle\theta_1, \rho_1\rangle, \ldots, \langle\theta_k, \rho_k\rangle\})$ and a committable set requires communication dependencies to be satisfied.)

1. If $\langle\theta_s, M^{Evt}_s[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle\theta_r, \rho_r\rangle){:}\rho_s\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$ and $\langle\theta_r, M^{Evt}_r[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle\theta_s, \rho_s\rangle){:}\rho_r\rangle \notin \{\langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, e'_k, \rho_k\rangle\}$, then $\{\langle\theta_1, M^{IO}_1, e_1\rangle, \langle\theta_1, e'_1, \rho_1\rangle, \ldots, \langle\theta_k, M^{IO}_k, e_k\rangle, \langle\theta_k, e'_k, \rho_k\rangle\} \subseteq \mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle\}$; hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle\theta_s, M^{Evt}_s[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle\theta_r, M^{Evt}_r[\texttt{recvEvt}\ \kappa], \rho_r\rangle\})$ and $\mathsf{Committable}(\{\langle\theta_1, \rho_1\rangle, \ldots, \langle\theta_k, \rho_k\rangle\})$ implies that there is a sequence of transitions

$$\{\langle\theta_1, e_1\rangle, \ldots, \langle\theta_k, e_k\rangle\} \rightsquigarrow^* \{\langle\theta_1, e'_1\rangle, \ldots, \langle\theta_k, e'_k\rangle\}$$

2. If $\langle \theta_s, M_s^{Evt}[\texttt{alwaysEvt ()}], \mathsf{Send}(\langle \theta_r, \rho_r \rangle){:}\rho_s \rangle \in \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$ and $\langle \theta_r, M_r^{Evt}[\texttt{alwaysEvt } e^\dagger], \mathsf{Recv}(\langle \theta_s, \rho_s \rangle){:}\rho_r \rangle \in \{\langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, e_k', \rho_k \rangle\}$, then

$$\begin{aligned}
&\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \\
&\quad \equiv \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \\
&\qquad \langle \theta_s, M_s^{IO}, e_s \rangle, \langle \theta_s, M_s^{Evt}[\texttt{alwaysEvt ()}], \mathsf{Send}(\langle \theta_r, \rho_r \rangle){:}\rho_s \rangle, \\
&\qquad \langle \theta_r, M_r^{IO}, e_r \rangle, \langle \theta_r, M_r^{Evt}[\texttt{alwaysEvt } e^\dagger], \mathsf{Recv}(\langle \theta_s, \rho_s \rangle){:}\rho_r \rangle, \ldots, \\
&\qquad \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\}
\end{aligned}$$

Note that

$$\begin{aligned}
&\{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, e_1', \rho_1 \rangle, \ldots, \\
&\ \langle \theta_s, M_s^{IO}, e_s \rangle, \langle \theta_s, M_s^{Evt}[\texttt{sendEvt } \kappa\ e^\dagger], \rho_s \rangle, \\
&\ \langle \theta_r, M_r^{IO}, e_r \rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt } \kappa], \rho_r \rangle, \ldots, \\
&\ \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, e_k', \rho_k \rangle\} \\
&\qquad \subseteq \mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt } \kappa\ e^\dagger], \rho_s \rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt } \kappa], \rho_r \rangle\}
\end{aligned}$$

and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_s, \rho_s \rangle, \langle \theta_r, \rho_r \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$ (which follows from Lemma 14.4 and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_s, \mathsf{Send}(\langle \theta_r, \rho_r \rangle){:}\rho_s \rangle, \langle \theta_r, \mathsf{Recv}(\langle \theta_s, \rho_s \rangle){:}\rho_r \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$); hence, $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt } \kappa\ e^\dagger], \rho_s \rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt } \kappa], \rho_r \rangle\})$ implies that there is a sequence of transitions

$$\begin{aligned}
&\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_s, e_s \rangle, \langle \theta_r, e_r \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\
&\quad \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_s, M^{Evt}[\texttt{sendEvt } \kappa\ e^\dagger] \rangle, \\
&\qquad \langle \theta_r, M^{Evt}[\texttt{recvEvt } \kappa] \rangle, \ldots, \langle \theta_k, e_k' \rangle\}
\end{aligned}$$

We take an EvtSendRecv transition in the original semantics (of the synchronous evaluation relation), yielding

$$\begin{aligned}
&\{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_s, M^{Evt}[\texttt{sendEvt } \kappa\ e^\dagger] \rangle, \\
&\ \langle \theta_r, M^{Evt}[\texttt{recvEvt } \kappa] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \\
&\quad \rightsquigarrow \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_s, M^{Evt}[\texttt{alwaysEvt ()}] \rangle, \\
&\qquad \langle \theta_r, M^{Evt}[\texttt{alwaysEvt } e^\dagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\}
\end{aligned}$$

Hence,

$$\begin{aligned}
&\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_s, e_s \rangle, \langle \theta_r, e_r \rangle, \ldots, \langle \theta_k, e_k \rangle\} \\
&\quad \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_s, M^{Evt}[\texttt{alwaysEvt ()}] \rangle, \\
&\qquad \langle \theta_r, M^{Evt}[\texttt{alwaysEvt } e^\dagger] \rangle, \ldots, \langle \theta_k, e_k' \rangle\} \quad \square
\end{aligned}$$

Finally, we show that the refined semantics of Section 6.2 may be simulated by the the concurrent evaluation relation of Section 4.2.3.

*Theorem 16*
If $\mathsf{SyncSim}(\mathcal{P})$ and $\mathcal{P} \xrightarrow{a} \mathcal{P}'$ according to the semantics of Figure 9,
then $\mathsf{SyncSim}(\mathcal{P})$ and either $|\mathcal{P}| \xrightarrow{a} |\mathcal{P}'|$ or $|\mathcal{P}| \xrightarrow{\tau}{}^0 |\mathcal{P}'|$ (and $a = \epsilon$) according to the semantics of Figures 3 and 4.

*Proof*

Note that $\mathsf{SyncSim}(\mathcal{P}')$ follows by applying Lemma 15.

The remainder of the theorem follows by case analysis on the concurrent evaluation relation.

For an IOEval, IOBindUnit, IOBindThrow, IOCatchUnit, IOCatchThrow, IOGetChar, IOPutChar, IOFork, or IOMyThreadId transition, the transition is of the form

$$\mathcal{P} \uplus \langle \theta_i, M_i^{IO}[e_i^\dagger] \rangle \xrightarrow{a} \mathcal{P} \uplus \langle \theta_i, M_i^{IO}[e_i^\ddagger] \rangle$$

(for transition appropriate $e_i^\dagger$ and $e_i^\ddagger$). Note that $|\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[e_i^\dagger] \rangle\}| = |\mathcal{P}| \uplus \{\langle \theta_i, M_i^{IO}[e_i^\dagger] \rangle\}$ and $|\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[e_i^\ddagger] \rangle\}| = |\mathcal{P}| \uplus \{\langle \theta_i, M_i^{IO}[e_i^\ddagger] \rangle\}$. We take the corresponding transition in the original semantics (of concurrent evaluation), yielding

$$|\mathcal{P}| \uplus \langle \theta_i, M_i^{IO}[e_i^\dagger] \rangle \xrightarrow{a} |\mathcal{P}| \uplus \langle \theta_i, M_i^{IO}[e_i^\ddagger] \rangle$$

For a SyncInit transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\} \xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}, e_i \rangle, \langle \theta_i, e_i, \bullet \rangle\}$$

We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\}||\mathcal{P}| \uplus \{\langle \theta_i, M_i^{IO}[\texttt{sync}\ e_i] \rangle\} = |\mathcal{P} \uplus \{\langle \theta_i, M_i^{IO}, e_i \rangle, \langle \theta_i, e_i, \bullet \rangle\}|$.

For a SyncCommit transition, the transition is of the form

$$\frac{\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})}{\begin{array}{l}\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e_1', \rho_1 \rangle, \ldots, \\ \quad \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e_k', \rho_k \rangle\} \\ \quad \xrightarrow{\epsilon} \mathcal{P}\backslash_{\{\theta_1,\ldots,\theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1'] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k'] \rangle\}\end{array}}$$

We construct an IOSync transition in the original semantics as follows. First, note that $|\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e_k', \rho_k \rangle\}| = |\mathcal{P}| \uplus \{\langle \theta_1, M_1^{IO}[\texttt{sync}\ e_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{sync}\ e_k] \rangle$. Second, note that $\mathsf{SyncSim}(\mathcal{P} \uplus \{\langle \theta_1, M_1^{IO}, e_1 \rangle, \langle \theta_1, \texttt{alwaysEvt}\ e_1', \rho_1 \rangle, \ldots, \langle \theta_k, M_k^{IO}, e_k \rangle, \langle \theta_k, \texttt{alwaysEvt}\ e_k', \rho_k \rangle\})$ and $\mathsf{Committable}(\{\langle \theta_1, \rho_1 \rangle, \ldots, \langle \theta_k, \rho_k \rangle\})$ implies that there is a sequence of transitions

$$\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_k, e_k \rangle\} \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_k, e_k' \rangle\}$$

Finally, note that $|\mathcal{P}\backslash_{\{\theta_1,\ldots,\theta_k\}} \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k] \rangle\}| = |\mathcal{P}\backslash_{\{\theta_1,\ldots,\theta_k\}}| \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k] \rangle\}$. Hence, we take an IOSync transition of the form:

$$\frac{\{\langle \theta_1, e_1 \rangle, \ldots, \langle \theta_k, e_k \rangle\} \rightsquigarrow^* \{\langle \theta_1, e_1' \rangle, \ldots, \langle \theta_k, e_k' \rangle\}}{\begin{array}{l}|\mathcal{P}| \uplus \{\langle \theta_1, M_1^{IO}[\texttt{sync}\ e_1] \rangle, \langle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{sync}\ e_k] \rangle\} \\ \quad \xrightarrow{\epsilon} |\mathcal{P}\backslash_{\{\theta_1,\ldots,\theta_k\}}| \uplus \{\langle \theta_1, M_1^{IO}[\texttt{unitIO}\ e_1'] \rangle, \ldots, \langle \theta_k, M_k^{IO}[\texttt{unitIO}\ e_k'] \rangle\}\end{array}}$$

For an EvtEval, EvtThenAlways, EvtThenThrow, EvtCatchAlways, EvtCatchThrow, EvtNewSChan, or EvtMyThreadId transition, the transition is of the form

$$\mathcal{P} \uplus \langle \theta_i, M_i^{IO}[e_i^\dagger] \rangle \xrightarrow{a} \mathcal{P} \uplus \langle \theta_i, M_i^{IO}[e_i^\ddagger] \rangle$$

(for transition appropriate $e_i^\dagger$ and $e_i^\ddagger$). We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[e_i^\dagger], \rho_i\rangle\}| = |\mathcal{P}| = |\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[e_i^\ddagger], \rho_i\rangle\}|$.

For an EvtNever transition, the transition is of the form

$$\frac{}{\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[\texttt{neverEvt}], \rho_i\rangle\} \xrightarrow{\epsilon} \mathcal{P}}$$

We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_i, M^{Evt}[\texttt{neverEvt}], \rho_i\rangle\}| = |\mathcal{P}|$.

For an EvtChoose transition, the transition is of the form

$$\begin{aligned}&\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\}\\ &\quad\xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle\}\end{aligned}$$

We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[\texttt{chooseEvt}\ e_i^l\ e_i^r], \rho_i\rangle\}| = |\mathcal{P}| = |\mathcal{P} \uplus \{\langle \theta_i, M_i^{Evt}[e_i^l], \mathsf{Left}{:}\rho_i\rangle, \langle \theta_i, M_i^{Evt}[e_i^r], \mathsf{Right}{:}\rho_i\rangle\}|$.

For an EvtThrow transition, the transition is of the form

$$\mathcal{P} \uplus \{\langle \theta_i, \texttt{throwEvt}\ e_i^\dagger, \rho_i\rangle\} \xrightarrow{\epsilon} \mathcal{P}$$

We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_i, \texttt{throwEvt}\ e_i^\dagger, \rho_i\rangle\}| = |\mathcal{P}|$.

For an EvtSendRecv transition, the transition is of the form

$$\frac{\mathsf{Coherent}(\langle \theta_s, \rho_s\rangle, \langle \theta_r, \rho_r\rangle)}{\begin{aligned}&\mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa], \rho_r\rangle\}\\ &\quad\xrightarrow{\epsilon} \mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa], \rho_r\rangle,\\ &\qquad\qquad \langle \theta_s, M_s^{Evt}[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle \theta_r, \rho_r\rangle){:}\rho_s\rangle,\\ &\qquad\qquad \langle \theta_r, M_r^{Evt}[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle \theta_s, \rho_s\rangle){:}\rho_r\rangle\}\end{aligned}}$$

We take no transition, noting that $|\mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa], \rho_r\rangle\} = \mathcal{P} \uplus \{\langle \theta_s, M_s^{Evt}[\texttt{sendEvt}\ \kappa\ e^\dagger], \rho_s\rangle, \langle \theta_r, M_r^{Evt}[\texttt{recvEvt}\ \kappa], \rho_r\rangle, \langle \theta_s, M_s^{Evt}[\texttt{alwaysEvt}\ ()], \mathsf{Send}(\langle \theta_r, \rho_r\rangle){:}\rho_s\rangle, \langle \theta_r, M_r^{Evt}[\texttt{alwaysEvt}\ e^\dagger], \mathsf{Recv}(\langle \theta_s, \rho_s\rangle){:}\rho_r\rangle\}|$. □

An immediate consequence is a generalized the simulation theorem.

*Theorem 17*
If $\mathsf{SyncSim}(\mathcal{P})$ and $\mathcal{P} \xrightarrow{a_1;\cdots;a_n}{}^* \mathcal{P}'$ according to the semantics of Figure 9,
then $\mathsf{SyncSim}(\mathcal{P})$ and $|\mathcal{P}| \xrightarrow{a'_1;\cdots;a'_m}{}^* |\mathcal{P}'|$ according to the semantics of Figures 3 and 4,
where $a_1;\cdots;a_n$ equals $a'_1;\cdots;a'_m$ modulo the deletion of $\epsilon$ actions.

*Proof*
The theorem follows by induction on the concurrent evaluation relation, applying Theorem 16. □

The simpler simulation theorem stated in Section 6.2.1 (which avoids mentioning the SyncSim predicate) follows directly.

*Theorem 10*

If $\mathcal{T} \xrightarrow{a_1;\cdots;a_n}{}^{*} \mathcal{P}'$ according to the semantics of Figure 9,
then $\mathcal{T} \xrightarrow{a'_1;\cdots;a'_m}{}^{*} |\mathcal{P}'|$ according to the semantics of Figures 3 and 4,
where $a_1;\cdots;a_n$ equals $a'_1;\cdots;a'_m$ modulo the deletion of $\epsilon$ actions.

*Proof*
The theorem follows by applying Theorem 17, noting that $\mathsf{SyncSim}(\mathcal{T})$ vacuously holds. $\square$

## References

Donnelly, Kevin, & Fluet, Matthew. (2008). Transactional Events. *Journal of Functional Programming*, **18**(5&6), 649–706.