Addendum for the Paper "Taming Message-passing Communication in Compositional Reasoning about Confidentiality"

Ximeng Li   Heiko Mantel   Markus Tasch

Department of Computer Science, TU-Darmstadt, Germany
{li,mantel,tasch}@mais.informatik.tu-darmstadt.de

In this document, we provide the supplementary material for the paper "Taming Message-passing Communication for Compositional Reasoning about Confidentiality" [1]. In Sect. 1, we provide the complete structural operational semantics for our programming language, and the complete instrumented semantics supporting our definition of "sound use of assumptions". In Sect. 2, we provide the proofs for Theorem 1 and Theorem 2 from the paper. In Sect. 3, we present an information-flow type system for local security that does not contain semantic side conditions. In Sect. 4, we discuss the typability of the authentication example and the auction example from the paper, outlining the differences between the type system from the paper and the variant from Sect. 3 of this addendum.

1 Operational Semantics of Message-passing Language

1.1 Basic Semantics

Below we present the rules defining the operational semantics of our message-passing language.

$$\frac{}{\langle\langle \mathsf{skip}; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma\rangle} \qquad \frac{\mathit{mem}' = \mathit{mem}[x \mapsto [\![e]\!]_{\mathit{mem}}]}{\langle\langle x := e; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma\rangle}$$

$$\frac{b \in \{\mathit{tt}, \mathit{ff}\} \quad [\![e]\!]_{\mathit{mem}} = b}{\langle\langle \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_{\mathit{tt}}\ \mathsf{else}\ \mathit{prog}_{\mathit{ff}}\ \mathsf{fi}; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathit{prog}_b; \mathit{mem}\rangle; \sigma\rangle}$$

$$\frac{[\![e]\!]_{\mathit{mem}} = \mathit{tt}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathit{prog}; \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma\rangle}$$

$$\frac{[\![e]\!]_{\mathit{mem}} = \mathit{ff}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma\rangle}$$

$$\frac{\langle\langle \mathit{prog}_1; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathit{prog}_1'; \mathit{mem}'\rangle; \sigma'\rangle \quad \mathit{prog}_1' \neq \mathsf{stop}}{\langle\langle \mathit{prog}_1; \mathit{prog}_2; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathit{prog}_1'; \mathit{prog}_2; \mathit{mem}'\rangle; \sigma'\rangle} \qquad \frac{\langle\langle \mathit{prog}_1; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma'\rangle}{\langle\langle \mathit{prog}_1; \mathit{prog}_2; \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathit{prog}_2; \mathit{mem}'\rangle; \sigma'\rangle}$$

$$\frac{\sigma(\mathit{ch}) = v \cdot \gamma \quad \mathit{mem}' = \mathit{mem}[x \mapsto v] \quad \sigma' = \sigma[\mathit{ch} \mapsto \gamma]}{\langle\langle \mathsf{recv}(\mathit{ch}, x); \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma'\rangle}$$

$$\frac{\sigma(\mathit{ch}) = v \cdot \gamma \quad \mathit{mem}' = \mathit{mem}[x \mapsto v][x_{\mathrm{b}} \mapsto \mathit{tt}] \quad \sigma' = \sigma[\mathit{ch} \mapsto \gamma]}{\langle\langle \mathsf{if\text{-}recv}(\mathit{ch}, x, x_{\mathrm{b}}); \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma'\rangle}$$

$$\frac{\sigma(\mathit{ch}) = \epsilon \quad \mathit{mem}' = \mathit{mem}[x_{\mathrm{b}} \mapsto \mathit{ff}]}{\langle\langle \mathsf{if\text{-}recv}(\mathit{ch}, x, x_{\mathrm{b}}); \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma\rangle}$$

$$\frac{[\![e]\!]_{\mathit{mem}} = v \quad \sigma' = \sigma[\mathit{ch} \mapsto \sigma(\mathit{ch}) \cdot v]}{\langle\langle \mathsf{send}(\mathit{ch}, e); \mathit{mem}\rangle; \sigma\rangle \to \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma'\rangle}$$

1.2 Instrumented Semantics

Below we present a calculus defining the judgment $\langle \mathit{pcnf}; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle \mathit{pcnf}'; \sigma'; \mu'\rangle$. This judgment says: A process with the process configuration *pcnf* and the set *chs* of channels used in the continuation executes one step from the channel state σ, resulting in the process configuration pcnf' and channel state σ', updating the instrumentation state from μ to μ'.
In the calculus to be presented, we write $\mu^\bullet$ for $\lambda ch\!:\!\mathit{ICh}.\mu(ch)\!\downarrow_1$, and μ° for $\lambda ch\!:\!\mathit{ICh}.\mu(ch)\!\downarrow_2$. We use the function $\mathit{ichs\text{-}of} : \mathit{Prog} \rightarrow 2^{\mathit{ICh}}$ to obtain the set of internal channels syntactically occurring in a program. This function is inductively defined by $\mathit{ichs\text{-}of}(\mathsf{send}(ch, e)) = \mathit{ichs\text{-}of}({}^{as}\mathsf{recv}(ch, x)) = \mathit{ichs\text{-}of}({}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}})) = \{ch\}$, $\mathit{ichs\text{-}of}(\mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}) = \mathit{chs\text{-}of}(\mathit{prog}_1; \mathit{prog}_2) = \mathit{ichs\text{-}of}(\mathit{prog}_1) \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)$, $\mathit{ichs\text{-}of}(\mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}) = \mathit{ichs\text{-}of}(\mathit{prog})$, and $\mathit{ichs\text{-}of}(\mathit{prog}) = \emptyset$ otherwise.

$$\frac{}{\langle\langle \mathsf{skip}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma; \mu\rangle} \qquad \frac{\mathit{mem}' = \mathit{mem}[x \mapsto [\![e]\!]_{\mathit{mem}}]}{\langle\langle x := e; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma; \mu\rangle}$$

$$\frac{\begin{array}{c} \mathit{lev}\langle e\rangle = \mathbb{H} \quad b \in \{\mathit{tt}, \mathit{ff}\} \quad [\![e]\!]_{\mathit{mem}} = b \\ \mathit{chs}' = \mathit{ichs\text{-}of}(\mathit{prog}_{\mathit{tt}}) \cup \mathit{ichs\text{-}of}(\mathit{prog}_{\mathit{ff}}) \cup \mathit{chs} \quad \mu' = (\mu^\bullet[\mathit{chs}' \mapsto \mathbb{H}^{|\mu^\bullet(\mathit{chs}')|}], \mu^\circ[\mathit{chs}' \mapsto \mathbb{H}]) \end{array}}{\langle\langle \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_{\mathit{tt}}\ \mathsf{else}\ \mathit{prog}_{\mathit{ff}}\ \mathsf{fi}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}_b; \mathit{mem}\rangle; \sigma; \mu'\rangle}$$

$$\frac{\mathit{lev}\langle e\rangle = \mathbb{L} \quad b \in \{\mathit{tt}, \mathit{ff}\} \quad [\![e]\!]_{\mathit{mem}} = b}{\langle\langle \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_{\mathit{tt}}\ \mathsf{else}\ \mathit{prog}_{\mathit{ff}}\ \mathsf{fi}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}_b; \mathit{mem}\rangle; \sigma; \mu\rangle}$$

$$\frac{\begin{array}{c} \mathit{lev}\langle e\rangle = \mathbb{H} \quad [\![e]\!]_{\mathit{mem}} = \mathit{tt} \\ \mathit{chs}' = \mathit{ichs\text{-}of}(\mathit{prog}) \cup \mathit{chs} \quad \mu' = (\mu^\bullet[\mathit{chs}' \mapsto \mathbb{H}^{|\mu^\bullet(\mathit{chs}')|}], \mu^\circ[\mathit{chs}' \mapsto \mathbb{H}]) \end{array}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}; \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu'\rangle}$$

$$\frac{\begin{array}{c} \mathit{lev}\langle e\rangle = \mathbb{H} \quad [\![e]\!]_{\mathit{mem}} = \mathit{ff} \\ \mathit{chs}' = \mathit{ichs\text{-}of}(\mathit{prog}) \cup \mathit{chs} \quad \mu' = (\mu^\bullet[\mathit{chs}' \mapsto \mathbb{H}^{|\mu^\bullet(\mathit{chs}')|}], \mu^\circ[\mathit{chs}' \mapsto \mathbb{H}]) \end{array}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma; \mu'\rangle}$$

$$\frac{\mathit{lev}\langle e\rangle = \mathbb{L} \quad [\![e]\!]_{\mathit{mem}} = \mathit{tt}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}; \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu\rangle}$$

$$\frac{\mathit{lev}\langle e\rangle = \mathbb{L} \quad [\![e]\!]_{\mathit{mem}} = \mathit{ff}}{\langle\langle \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathsf{stop}; \mathit{mem}\rangle; \sigma; \mu\rangle}$$

$$\frac{\begin{array}{c} \langle\langle \mathit{prog}_1; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}'} \langle\langle \mathit{prog}_1'; \mathit{mem}'\rangle; \sigma'; \mu'\rangle \\ \mathit{chs}' = \mathit{chs} \cup \mathit{ichs\text{-}of}(\mathit{prog}_2) \quad \mathit{prog}_1' \neq \mathsf{stop} \end{array}}{\langle\langle \mathit{prog}_1; \mathit{prog}_2; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}_1'; \mathit{prog}_2; \mathit{mem}'\rangle; \sigma'; \mu'\rangle}$$

$$\frac{\langle\langle \mathit{prog}_1; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}'} \langle\langle \mathsf{stop}; \mathit{mem}'\rangle; \sigma'; \mu'\rangle \quad \mathit{chs}' = \mathit{chs} \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)}{\langle\langle \mathit{prog}_1; \mathit{prog}_2; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}_2; \mathit{mem}'\rangle; \sigma'; \mu'\rangle}$$

$$\frac{ch \in ECh \quad \sigma(ch) = v \cdot \gamma \quad mem' = mem[x \mapsto v] \quad \sigma' = \sigma[ch \mapsto \gamma]}{\langle\langle {}^{as}\mathsf{recv}(ch, x); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem'\rangle; \sigma'; \mu\rangle}$$

$$\frac{\begin{array}{c} ch \in ICh \quad \mu' = (\mu^\bullet[ch \mapsto tail(\mu^\bullet(ch))], \mu^\circ) \\ \sigma(ch) = v \cdot \gamma \quad mem' = mem[x \mapsto v] \quad \sigma' = \sigma[ch \mapsto \gamma]\end{array}}{\langle\langle {}^{as}\mathsf{recv}(ch, x); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem'\rangle; \sigma'; \mu'\rangle}$$

$$\frac{ch \in ECh \quad \sigma(ch) = v \cdot \gamma \quad mem' = mem[x \mapsto v][x_{\mathrm{b}} \mapsto tt] \quad \sigma' = \sigma[ch \mapsto \gamma]}{\langle\langle {}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}}); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem'\rangle; \sigma'; \mu\rangle}$$

$$\frac{\begin{array}{c} ch \in ICh \quad \mu' = (\mu^\bullet[ch \mapsto tail(\mu^\bullet(ch))], \mu^\circ) \\ \sigma(ch) = v \cdot \gamma \quad mem' = mem[x \mapsto v][x_{\mathrm{b}} \mapsto tt] \quad \sigma' = \sigma[ch \mapsto \gamma]\end{array}}{\langle\langle {}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}}); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem'\rangle; \sigma'; \mu'\rangle}$$

$$\frac{\sigma(ch) = \epsilon \quad mem' = mem[x_{\mathrm{b}} \mapsto ff]}{\langle\langle {}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}}); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem'\rangle; \sigma; \mu\rangle}$$

$$\frac{ch \in ECh \quad [\![e]\!]_{mem} = v \quad \sigma' = \sigma[ch \mapsto \sigma(ch) \cdot v]}{\langle\langle \mathsf{send}(ch, e); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem\rangle; \sigma'; \mu\rangle}$$

$$\frac{ch \in ICh \quad [\![e]\!]_{mem} = v \quad \sigma' = \sigma[ch \mapsto \sigma(ch) \cdot v] \quad \mu' = (\mu^\bullet[ch \mapsto \mu^\bullet(ch) \cdot (\mu^\circ(ch) \sqcup lev\langle e\rangle)], \mu^\circ)}{\langle\langle \mathsf{send}(ch, e); mem\rangle; \sigma; \mu\rangle \rightarrow_{chs} \langle\langle \mathsf{stop}; mem\rangle; \sigma'; \mu'\rangle}$$

We use the judgment $\langle pcnf; \sigma; \mu\rangle \rightarrow \langle pcnf'; \sigma'; \mu'\rangle$ to say that a process with the process configuration $pcnf$ executes one step from the channel state σ, resulting in the process configuration $pcnf'$ and channel state σ', updating the instrumentation state from μ to μ'. We define this judgment by $\langle pcnf; \sigma; \mu\rangle \rightarrow \langle pcnf'; \sigma'; \mu'\rangle$ if and only if $\langle pcnf; \sigma; \mu\rangle \rightarrow_{\emptyset} \langle pcnf'; \sigma'; \mu'\rangle$. Our semantic instrumentation is *transparent* wrt. the basic semantics—it neither forbids any local computation steps, nor enables any additional ones. This result will be formalized in Sect. 2.
We use the judgment $rtr \rightarrow_{\xi} rtr'$ to state that the rich trace rtr is extended into the rich trace rtr' after one step of a distributed program, or of the environment, under the strategy ξ. We define this judgment on the basis of the judgment $\langle pcnf; \sigma; \mu\rangle \rightarrow \langle pcnf'; \sigma'; \mu'\rangle$ for the process-local instrumented semantics.

$$\frac{\begin{array}{c} last(rtr) = \langle pcnfl; \sigma; \mu\rangle \\ \xi(trace\text{-}of(rtr)) = \sigma' \end{array}}{rtr \rightarrow_{\xi} rtr \cdot \langle pcnfl; \sigma[ECh \mapsto \sigma'(ECh)]; \mu\rangle} \qquad \frac{\begin{array}{c} last(rtr) = \langle [..., pcnf_i, ...]; \sigma; \mu\rangle \\ \langle pcnf_i; \sigma; \mu\rangle \rightarrow \langle pcnf'_i; \sigma'; \mu'\rangle \end{array}}{rtr \rightarrow_{\xi} rtr \cdot \langle [..., pcnf'_i, ...]; \sigma'; \mu'\rangle}$$

In the above, we obtain the trace underlying a rich trace using the function $trace\text{-}of : RTr \rightarrow Tr$, where $trace\text{-}of([\langle pcnfl_1; \sigma_1; \mu_1\rangle, \ldots, \langle pcnfl_n; \sigma_n; \mu_n\rangle]) = [\langle pcnfl_1; \sigma_1\rangle, \ldots, \langle pcnfl_n; \sigma_n\rangle]$.
Given a distributed program $dprog = [prog_1 || \ldots || prog_n]$, we define

$$rtraces_{\xi}(dprog) \triangleq \{rtr \mid [\ \langle[\langle prog_1; mem_{\mathrm{init}}\rangle, \ldots, \langle prog_n; mem_{\mathrm{init}}\rangle]; \sigma_{\mathrm{init}}; \mu_{\mathrm{init}}\rangle\ ](\rightarrow_{\xi})^* rtr\},$$

where μ_{init} is $\lambda ch\!:\!ICh.(\epsilon, \mathbb{L})$.

2 Proofs

We define a subset of high conditional programs by the following syntax. We denote the set of all high conditional programs by HCond.

$$\mathit{hcond} ::= \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}\ (\mathit{lev}\langle e\rangle = \mathbb{H}) \mid \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}\ (\mathit{lev}\langle e\rangle = \mathbb{H}) \mid \mathit{hcond}; \mathit{prog}$$

Lemma 1. *If* $\langle\langle \mathit{prog}; \mathit{mem}_1\rangle; \sigma_1; \mu_1\rangle \rightarrow_{\mathit{chs}_1} \langle\langle \mathit{prog}'_1; \mathit{mem}'_1\rangle; \sigma'_1; \mu'_1\rangle$, $\langle\langle \mathit{prog}; \mathit{mem}_2\rangle; \sigma_2; \mu_2\rangle \rightarrow_{\mathit{chs}_2} \langle\langle \mathit{prog}'_2; \mathit{mem}'_2\rangle; \sigma'_2; \mu'_2\rangle$, $\mathit{mem}_1 =_{\mathbb{L}} \mathit{mem}_2$, *and* $\mathit{prog}'_1 \neq \mathit{prog}'_2$, *then* $\mathit{prog} \in \mathit{HCond}$.

Proof. We proceed with a structural induction on $\mathit{prog} \in \mathit{Prog}$.

Case $\mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}$: From $\mathit{mem}_1 =_{\mathbb{L}} \mathit{mem}_2$ and $\mathit{prog}'_1 \neq \mathit{prog}'_2$, it is not difficult to see that $\mathit{lev}\langle e\rangle = \mathbb{H}$. Thus, $\mathit{prog} \in \mathit{HCond}$.

Case $\mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}_1\ \mathsf{od}$: Analogous to the previous case.

Case $\mathit{prog}_1; \mathit{prog}_2$: Without loss of generality, we first show $\mathit{prog}_1 \in \mathit{HCond}$ by the following case analysis on prog'_1 and prog'_2 in the lemma statement.

Sub-case $\mathit{prog}'_1 \neq \mathit{prog}_2 \wedge \mathit{prog}'_2 \neq \mathit{prog}_2$: There exist prog' and prog'' such that $\mathit{prog}' \neq \mathit{prog}''$, $\mathit{prog}'_1 = \mathit{prog}'; \mathit{prog}_2$, $\mathit{prog}'_2 = \mathit{prog}''; \mathit{prog}_2$, and

$$\langle\langle \mathit{prog}_1; \mathit{mem}_1\rangle; \sigma_1; \mu_1\rangle \rightarrow_{\mathit{chs}_1 \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)} \langle\langle \mathit{prog}'; \mathit{mem}'_1\rangle; \sigma'_1; \mu'_1\rangle$$
$$\langle\langle \mathit{prog}_1; \mathit{mem}_2\rangle; \sigma_2; \mu_2\rangle \rightarrow_{\mathit{chs}_2 \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)} \langle\langle \mathit{prog}''; \mathit{mem}'_2\rangle; \sigma'_2; \mu'_2\rangle$$

Using the induction hypothesis, we can derive $\mathit{prog}_1 \in \mathit{HCond}$.

Sub-case $\mathit{prog}'_1 = \mathit{prog}_2 \wedge \mathit{prog}'_2 \neq \mathit{prog}_2$: There exists some $\mathit{prog}'' \neq \mathsf{stop}$ such that $\mathit{prog}'_2 = \mathit{prog}''; \mathit{prog}_2$, and

$$\langle\langle \mathit{prog}_1; \mathit{mem}_1\rangle; \sigma_1; \mu_1\rangle \rightarrow_{\mathit{chs}_1 \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)} \langle\langle \mathsf{stop}; \mathit{mem}'_1\rangle; \sigma'_1; \mu'_1\rangle$$
$$\langle\langle \mathit{prog}_1; \mathit{mem}_2\rangle; \sigma_2; \mu_2\rangle \rightarrow_{\mathit{chs}_2 \cup \mathit{ichs\text{-}of}(\mathit{prog}_2)} \langle\langle \mathit{prog}''; \mathit{mem}'_2\rangle; \sigma'_2; \mu'_2\rangle$$

Using the induction hypothesis, we can derive $\mathit{prog}_1 \in \mathit{HCond}$.

Sub-case $\mathit{prog}'_1 = \mathit{prog}_2 \wedge \mathit{prog}'_2 = \mathit{prog}_2$: Contradiction with $\mathit{prog}'_1 \neq \mathit{prog}'_2$.

Since $\mathit{prog}_1 \in \mathit{HCond}$, $\mathit{prog}_2 \in \mathit{Prog}$, we have $\mathit{prog}_1; \mathit{prog}_2 \in \mathit{HCond}$.

In the remaining cases, the statement of the lemma vacuously holds. $\square$

Lemma 2. *If* $\mathit{prog} \in \mathit{HCond}$, *and* $\langle\langle \mathit{prog}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}'; \mathit{mem}'\rangle; \sigma'; \mu'\rangle$, *then* $\forall \mathit{ch} \in \mathit{ichs\text{-}of}(\mathit{prog}) \cup \mathit{chs} : \exists \mathit{llst} \in \mathit{Lev}^* : \mu'(\mathit{ch}) = (\mathit{llst}, \mathbb{H})$.

Proof. It is sufficient to prove

If $\mathit{hcond} \rightarrow^* \mathit{prog}$, and $\langle\langle \mathit{prog}; \mathit{mem}\rangle; \sigma; \mu\rangle \rightarrow_{\mathit{chs}} \langle\langle \mathit{prog}'; \mathit{mem}'\rangle; \sigma'; \mu'\rangle$, then $\forall \mathit{ch} \in \mathit{ichs\text{-}of}(\mathit{prog}) \cup \mathit{chs} : \exists \mathit{llst} \in \mathit{Lev}^* : \mu'(\mathit{ch}) = (\mathit{llst}, \mathbb{H})$.

We proceed with an induction on the derivation of prog.

- Suppose $\mathit{hcond} \rightarrow \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}$, $\mathit{lev}\langle e\rangle = \mathbb{H}$, and $\mathit{prog} = \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}$. It is straightforward to derive the conclusion of the lemma by inspection of the rules of the instrumented semantics.
- Suppose $\mathit{hcond} \rightarrow \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}_1\ \mathsf{od}$, $\mathit{lev}\langle e\rangle = \mathbb{H}$, and $\mathit{prog} = \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}_1\ \mathsf{od}$.
  It is straightforward to derive the conclusion of the lemma by inspection of the rules of the instrumented semantics.

– Suppose $hcond \rightarrow hcond_1; prog_2$, $hcond_1 \rightarrow^* prog_1$, and $prog = prog_1; prog_2$.
  We have $\langle\langle prog_1; mem\rangle; \sigma; \mu\rangle \rightarrow_{chs \cup ichs\text{-}of(prog_2)} \langle\langle prog_1'; mem'\rangle; \sigma'; \mu'\rangle$. Thus by the induction hypothesis, we have $\forall ch \in ichs\text{-}of(prog_1) \cup (chs \cup ichs\text{-}of(prog_2)) : \exists llst \in Lev^* : \mu'(ch) = (llst, \mathbb{H})$. Hence $\forall ch \in ichs\text{-}of(prog_1; prog_2) \cup chs : \exists llst \in Lev^* : \mu'(ch) = (llst, \mathbb{H})$.

This completes the proof. □

Lemma 3. *If $prog \in HCond$, and $\langle\langle prog; mem\rangle; \sigma; \mu\rangle \rightarrow \langle\langle prog'; mem'\rangle; \sigma'; \mu'\rangle$, then we have $\forall ch \in ichs\text{-}of(prog) : \exists llst \in Lev^* : \mu'(ch) = (llst, \mathbb{H})$.*

Proof. The result can be directly obtained by instantiating Lemma 2 with $chs = \emptyset$. □

By writing the low projection $\lfloor vlst \rfloor_{(llst,\ell)}$, we assume that the expression is defined. For a list $l = [a_0, \dots, a_{|l|-1}]$, and $i \in \{0, \dots, |l| - 1\}$, we also write $l(i)$ for the element a_i.

Lemma 4. *If $\lfloor vlst_1 \rfloor_{(llst_1,\ell_1)} = \lfloor vlst_2 \rfloor_{(llst_2,\ell_2)}$, then we have*
1. *$\ell_1 = \ell_2$,*
2. *$\ell_1 = \mathbb{L} \Rightarrow (llst_1 = llst_2 \wedge |vlst_1| = |vlst_2|)$, and*
3. *$\ell_1 = \mathbb{L} \wedge j \in \{0, \dots, |vlst| - 1\} \wedge llst_1(j) = \mathbb{L} \Rightarrow vlst_1(j) = vlst_2(j)$.*

Proof. Straightforward using the facts that $\lfloor vlst_1 \rfloor_{(llst_1,\ell_1)}$ and $\lfloor vlst_2 \rfloor_{(llst_2,\ell_2)}$ are defined, $\odot \notin Val$ and $\circledcirc \notin Val$. □

We restate and prove Proposition 1.

Proposition 1. *If $\forall ch \in ICh : \lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)}$, $(\sigma_1, \mu_1) \models acs_1$, and $(\sigma_2, \mu_2) \models acs_2$, then $\sigma_1 \overset{acs_1 \cup acs_2}{=\!=\!=} \sigma_2$.*

Proof. Under the hypotheses of the proposition, we show

$$\forall ch \in ICh : (\mathbb{L}^\bullet, ch) \in acs_1 \cup acs_2 \Rightarrow ((|\sigma_1(ch)| > 0 \wedge |\sigma_2(ch)| > 0) \Rightarrow first(\sigma_1(ch)) = first(\sigma_2(ch)))$$
$$\forall ch \in ICh : (\mathbb{L}^\circ, ch) \in acs_1 \cup acs_2 \Rightarrow (|\sigma_1(ch)| > 0 \Leftrightarrow |\sigma_2(ch)| > 0)$$

Pick an arbitrary $ch \in ICh$.
Suppose $(\mathbb{L}^\bullet, ch) \in acs_1 \cup acs_2$. Without loss of generality, we assume $(\mathbb{L}^\bullet, ch) \in acs_1$. From $(\sigma_1, \mu_1) \models acs_1$, we have $|\mu_1^\bullet(ch)| > 0 \Rightarrow first(\mu_1^\bullet(ch)) = \mathbb{L}$. Hence $|\mu_1^\bullet(ch)| > 0 \Rightarrow \mu_1^\circ(ch) = \mathbb{L}$. Thus we have if $|\sigma_1(ch)| > 0 \wedge |\sigma_2(ch)| > 0$ then $first(\sigma_1(ch)) = first(\sigma_2(ch))$, by Lemma 4 and $\lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)}$.
Suppose $(\mathbb{L}^\circ, ch) \in acs_1 \cup acs_2$. Without loss of generality, we assume $(\mathbb{L}^\circ, ch) \in acs_1$. From $(\sigma_1, \mu_1) \models acs_1$, we have $\mu_1^\circ(ch) = \mathbb{L}$. Thus it is not difficult to derive $|\sigma_1(ch)| > 0 \Leftrightarrow |\sigma_2(ch)| > 0$ using $\lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)}$ and Lemma 4. □

Lemma 5. *If $mem_1 =_{\mathbb{L}} mem_2$, $\forall ch \in ICh : \lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)}$, $\langle\langle prog; mem_1\rangle; \sigma_1; \mu_1\rangle \rightarrow \langle\langle prog_1'; mem_1'\rangle; \sigma_1'; \mu_1'\rangle$, and $\langle\langle prog; mem_2\rangle; \sigma_2; \mu_2\rangle \rightarrow \langle\langle prog_2'; mem_2'\rangle; \sigma_2'; \mu_2'\rangle$, then $\forall ch \in ICh : \lfloor \sigma_1'(ch) \rfloor_{\mu_1'(ch)} = \lfloor \sigma_2'(ch) \rfloor_{\mu_2'(ch)}$.*

Proof. We proceed with an induction on the derivation of

$$\langle\langle prog; mem_1\rangle; \sigma_1; \mu_1\rangle \rightarrow \langle\langle prog_1'; mem_1'\rangle; \sigma_1'; \mu_1'\rangle \tag{1}$$

We only present a few representative cases.

Case Transition (1) is established by a rule for send. Here *prog* is of the form ${}^{as}\mathsf{send}(ch, e)$.

If $ch \in ECh$, then the conclusion of the lemma is trivial.

Suppose $ch \in ICh$. We have

$$\langle\langle {}^{as}\mathsf{send}(ch, e); mem_1\rangle; \sigma_1; \mu_1\rangle \to \langle\langle \mathsf{stop}; mem_1'\rangle; \sigma_1'; \mu_1'\rangle$$
$$\langle\langle {}^{as}\mathsf{send}(ch, e); mem_2\rangle; \sigma_2; \mu_2\rangle \to \langle\langle \mathsf{stop}; mem_2'\rangle; \sigma_2'; \mu_2'\rangle$$

By $\forall ch \in ICh : \lfloor \sigma_1(ch)\rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch)\rfloor_{\mu_2(ch)}$, and Lemma 4, we have $\mu_1^\circ(ch) = \mu_2^\circ(ch)$ for the output channel ch.

We proceed with a case analysis on $\mu_1^\circ(ch)$.

Sub-case $\mu_1^\circ(ch) = \mathbb{H}$. In this case we have $(\mu_1')^\circ(ch) = \mu_1^\circ(ch) = \mathbb{H}$. We obviously have $\lfloor \sigma_1'(ch)\rfloor_{\mu_1'(ch)} = \lfloor \sigma_1(ch)\rfloor_{\mu_1(ch)}$ for the output channel ch. Analogously we have $\lfloor \sigma_2'(ch)\rfloor_{\mu_2'(ch)} = \lfloor \sigma_2(ch)\rfloor_{\mu_2(ch)}$. On this basis, it is not difficult to obtain $\forall ch \in ICh : \lfloor \sigma_1'(ch)\rfloor_{\mu_1'(ch)} = \lfloor \sigma_2'(ch)\rfloor_{\mu_2'(ch)}$.

Sub-case $\mu_1^\circ(ch) = \mathbb{L}$. We have $(\mu_1')^\circ(ch) = \mu_1^\circ(ch) = \mathbb{L}$, and $(\mu_2')^\circ(ch) = \mu_2^\circ(ch) = \mathbb{L}$.

If $[\![e]\!]_{mem_1} = [\![e]\!]_{mem_2}$, then it is trivial to establish $\forall ch \in ICh : \lfloor \sigma_1(ch)\rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch)\rfloor_{\mu_2(ch)}$. Suppose $[\![e]\!]_{mem_1} \neq [\![e]\!]_{mem_2}$. Since $mem_1 =_{\mathbb{L}} mem_2$, we have $lev\langle e\rangle = \mathbb{H}$, and the two messages $[\![e]\!]_{mem_1}$ and $[\![e]\!]_{mem_2}$ are associated with the same security level $\mathbb{H}$ in $(\mu_1')^\bullet(ch)$ and $(\mu_2')^\bullet(ch)$. On this basis, it is not difficult to obtain $\forall ch \in ICh : \lfloor \sigma_1'(ch)\rfloor_{\mu_1'(ch)} = \lfloor \sigma_2'(ch)\rfloor_{\mu_2'(ch)}$.

Case Transition (1) is established by the first rule for sequential composition. We have that *prog* $=$ $prog_{\mathrm{a}}; prog_{\mathrm{b}}$ for some $prog_{\mathrm{a}}$ and $prog_{\mathrm{b}}$, such that $\langle\langle prog_{\mathrm{a}}; mem_1\rangle; \sigma_1; \mu_1\rangle \to \langle\langle prog_{\mathrm{a}}'; mem_1'\rangle; \sigma_1'; \mu_1'\rangle$ for some $prog_{\mathrm{a}}'$ with $prog_1' = prog_{\mathrm{a}}'; prog_{\mathrm{b}}$, and $\langle\langle prog_{\mathrm{a}}; mem_2\rangle; \sigma_2; \mu_2\rangle \to \langle\langle prog_{\mathrm{a}}''; mem_2'\rangle; \sigma_2'; \mu_2'\rangle$ for some $prog_{\mathrm{a}}''$ with $prog_2' = prog_{\mathrm{a}}''; prog_{\mathrm{b}}$, or $prog_{\mathrm{a}}'' = \mathsf{stop}$.

The conclusion of the lemma can now be derived using the induction hypothesis.

Case Transition (1) is established by a rule for if. We have $\sigma_1' = \sigma_1$, $\sigma_2' = \sigma_2$, $\forall ch \in ICh : \mu_1^\circ(ch) \sqsubseteq (\mu_1')^\circ(ch) \wedge \mu_2^\circ(ch) \sqsubseteq (\mu_2')^\circ(ch)$, $(\mu_1')^\circ = (\mu_2')^\circ$, and $(\mu_1')^\bullet(ch') = \mu_1^\bullet(ch') \wedge (\mu_2')^\bullet(ch') = \mu_2^\bullet(ch')$ for all channels ch' such that $(\mu_1')^\circ(ch') = (\mu_2')^\circ(ch') = \mathbb{L}$.

On this basis, it is not difficult to derive the conclusion of the lemma.

The completion of the induction above completes the proof. □

Lemma 6. *If* $\langle\langle prog; mem\rangle; \sigma; \mu\rangle \to \langle\langle prog'; mem'\rangle; \sigma'; \mu'\rangle$*,* $\forall ch \in ichs\text{-}of(prog) : \mu^\circ(ch) = \mathbb{H}$*, and* $(\mu')^\circ = \mu^\circ$*, then* $\forall ch \in ICh : \lfloor \sigma'(ch)\rfloor_{\mu'(ch)} = \lfloor \sigma(ch)\rfloor_{\mu(ch)}$*.*

Proof. We proceed with an induction on the derivation

$$\langle\langle prog; mem\rangle; \sigma; \mu\rangle \to \langle\langle prog'; mem'\rangle; \sigma'; \mu'\rangle \tag{2}$$

We only show a few representative cases in detail.

Case (2) is by the rule for send, over an external channel. Here *prog* is $\mathsf{send}(ch, e)$, with $ch \in ECh$. The result of the lemma is straightforward since $\forall ch \in ICh : \sigma'(ch) = \sigma(ch)$, and $\mu' = \mu$.

Case (2) is by the rule for send, over an internal channel. Here *prog* is $\mathsf{send}(ch, e)$ with $ch \in ICh$. We have $ch \in ichs\text{-}of(prog)$. Hence $\mu^\circ(ch) = \mathbb{H}$. By the instrumented semantics, we have $\forall ch' \in ICh \setminus \{ch\} : (\mu')^\bullet(ch') = \mu^\bullet(ch')$. Thus, we can derive the conclusion of the lemma using the hypothesis $(\mu')^\circ = \mu^\circ$.

Case (2) is by a rule for sequential composition. We have $prog = prog_1; prog_2$ for some $prog_1$ and $prog_2$, and $\langle\langle prog_1; mem\rangle; \sigma; \mu\rangle \rightarrow \langle\langle prog_1'; mem'\rangle; \sigma'; \mu'\rangle$ for some $prog_1'$. The result of the lemma can now be derived using the induction hypothesis,

Case (2) is by a rule for if. We have $\sigma' = \sigma$. Thus it is not difficult to derive the conclusion of the lemma using the hypothesis $(\mu')^\circ = \mu^\circ$.

The remaining cases where a non-composite command is executed are straightforward – the cases for communication are analogous to the cases for send. The cases for while are analogous to the cases for if. The induction completes the proof. □

Lemma 7. *If $prog_{10} = prog_{20}$, for all $k < n$, $\langle prog_{1k}; mem_{1k}\rangle \approx \langle prog_{2k}; mem_{2k}\rangle$, $\langle\langle prog_{1k}; mem_{1k}\rangle; \sigma_{1k}'; \mu_{1k}'\rangle \rightarrow \langle\langle prog_{1(k+1)}; mem_{1(k+1)}\rangle; \sigma_{1(k+1)}; \mu_{1(k+1)}\rangle$, $\langle\langle prog_{2k}; mem_{2k}\rangle; \sigma_{2k}'; \mu_{2k}'\rangle \rightarrow \langle\langle prog_{2(k+1)}; mem_{2(k+1)}\rangle; \sigma_{2(k+1)}; \mu_{2(k+1)}\rangle$, $\forall ch \in ICh : \lfloor\sigma_{1k}'(ch)\rfloor_{\mu_{1k}'(ch)} = \lfloor\sigma_{2k}'(ch)\rfloor_{\mu_{2k}'(ch)}$, and $\forall ch \in ICh : \mu_{1k}^\circ(ch) \sqsubseteq (\mu_{1k}')^\circ(ch) \wedge \mu_{2k}^\circ(ch) \sqsubseteq (\mu_{2k}')^\circ(ch)$, then $\forall ch \in ICh : \lfloor\sigma_{1n}(ch)\rfloor_{\mu_{1n}(ch)} = \lfloor\sigma_{2n}(ch)\rfloor_{\mu_{2n}(ch)}$.*

Proof. We make a case analysis on whether the derivatives of $prog_{10}$ and $prog_{20}$ become different prior to $prog_{1n}$ and $prog_{2n}$.

– Suppose there exists some $k_0 < n$ such that $prog_{1k_0} \neq prog_{2k_0}$. We have some $k' \leq k_0$ such that $prog_{1(k'-1)} = prog_{2(k'-1)}$ but $prog_{1k'} \neq prog_{2k'}$. By $\langle prog_{1(k'-1)}; mem_{1(k'-1)}\rangle \approx \langle prog_{2(k'-1)}; mem_{2(k'-1)}\rangle$, we have $mem_{1(k'-1)} =_{\mathbb{L}} mem_{2(k'-1)}$. Using Lemma 1, we can obtain $prog_{1(k'-1)} \in HCond$ and $prog_{2(k'-1)} \in HCond$. Using Lemma 2, we obtain that $\forall ch \in ichs\text{-}of(prog_{1(k'-1)}) : \mu_{1k'}^\circ(ch) = \mathbb{H}$, and $\forall ch \in ichs\text{-}of(prog_{2(k'-1)}) : \mu_{2k'}^\circ(ch) = \mathbb{H}$. It is not difficult to see that

$$ichs\text{-}of(prog_{1(k'-1)}) \supseteq ichs\text{-}of(prog_{1(n-1)}) \wedge ichs\text{-}of(prog_{2(k'-1)}) \supseteq ichs\text{-}of(prog_{2(n-1)})$$

Using the condition that for all $k < n, \forall ch \in ICh : \mu_{1k}^\circ(ch) \sqsubseteq (\mu_{1k}')^\circ(ch) \wedge \mu_{2k}^\circ(ch) \sqsubseteq (\mu_{2k}')^\circ(ch)$, and the instrumented semantics, we derive $\forall ch \in ichs\text{-}of(prog_{1(n-1)}) : (\mu_{1(n-1)}')^\circ(ch) = \mathbb{H}$, and $\forall ch \in ichs\text{-}of(prog_{2(n-1)}) : (\mu_{2(n-1)}')^\circ(ch) = \mathbb{H}$. It is not difficult to see that $\mu_{1n}^\circ = (\mu_{1(n-1)}')^\circ$ and that $\mu_{2n}^\circ = (\mu_{2(n-1)}')^\circ$.
Using Lemma 6, we can derive

$$\forall ch \in ICh : \lfloor\sigma_{1(n-1)}'(ch)\rfloor_{\mu_{1(n-1)}'(ch)} = \lfloor\sigma_{1n}(ch)\rfloor_{\mu_{1n}(ch)}$$
$$\forall ch \in ICh : \lfloor\sigma_{2(n-1)}'(ch)\rfloor_{\mu_{2(n-1)}'(ch)} = \lfloor\sigma_{2n}(ch)\rfloor_{\mu_{2n}(ch)}$$

Thus it follows from $\forall ch \in ICh : \lfloor\sigma_{1(n-1)}'(ch)\rfloor_{\mu_{1(n-1)}'(ch)} = \lfloor\sigma_{2(n-1)}'(ch)\rfloor_{\mu_{2(n-1)}'(ch)}$ that $\forall ch \in ICh : \lfloor\sigma_{1n}(ch)\rfloor_{\mu_{1n}(ch)} = \lfloor\sigma_{2n}(ch)\rfloor_{\mu_{2n}(ch)}$ holds.

– Suppose for all $k_0 < n$, $prog_{1k_0} = prog_{2k_0}$. We have $prog_{1(n-1)} = prog_{2(n-1)}$. It follows from Lemma 5 that $\forall ch \in ICh : \lfloor\sigma_{1n}(ch)\rfloor_{\mu_{1n}(ch)} = \lfloor\sigma_{2n}(ch)\rfloor_{\mu_{2n}(ch)}$ holds.

The case analysis above completes the proof. □

Proposition 2. *The following statements hold.*

– *If $\langle pcnf; \sigma; \mu\rangle \rightarrow \langle pcnf'; \sigma'; \mu'\rangle$, then $\langle pcnf; \sigma\rangle \rightarrow \langle pcnf'; \sigma'\rangle$.*
– *If $\langle pcnf; \sigma\rangle \rightarrow \langle pcnf'; \sigma'\rangle$, then for all μ, there exists μ' such that $\langle pcnf;\ \sigma; \mu\rangle \rightarrow \langle pcnf'; \sigma'; \mu'\rangle$.*

Proof. We prove augmented versions of the statements with universal quantification over sets chs of internal channels:

1. For all $chs \in 2^{ICh}$, if $\langle pcnf; \sigma; \mu\rangle \rightarrow_{chs} \langle pcnf'; \sigma'; \mu'\rangle$, then $\langle pcnf; \sigma\rangle \rightarrow \langle pcnf'; \sigma'\rangle$.
2. If $\langle pcnf; \sigma\rangle \rightarrow \langle pcnf'; \sigma'\rangle$, then for all μ, and $chs \in 2^{ICh}$, there exists μ' such that $\langle pcnf; \sigma; \mu\rangle \rightarrow_{chs} \langle pcnf'; \sigma'; \mu'\rangle$.

Each of 1 and 2 can be proved straightforwardly by induction on the appropriate semantic derivation. We omit the details. It is not difficult to see that the proposition directly follows from 1 and 2. □

We proceed with the proof of Theorem 1 from the paper.

Theorem 1. *For a distributed program $dprog = ||_i prog_i$, if $LSec(prog_i)$ holds for all i, and $dprog$ ensures a sound use of assumptions, then we have $KBSec(dprog)$.*

We construct the following binary relation on traces.

$$
\begin{array}{l}
R_{dprog} \triangleq \\
\{(trace\text{-}of(rtr_1), trace\text{-}of(rtr_2)) \mid \\
\quad rtr_1 \in rtraces_{\xi_1}(dprog) \wedge rtr_2 \in rtraces_{\xi_2}(dprog) \wedge |rtr_1| = |rtr_2| \wedge \\
\quad \forall k \in \{0, \ldots, |rtr_1| - 1\} : \\
\qquad \exists n, prog_{11}, \ldots, prog_{1n}, prog_{21}, \ldots, prog_{2n}, mem_{11}, \ldots, mem_{1n}, mem_{21}, \ldots, mem_{2n}, \sigma_1, \sigma_2, \mu_1, \mu_2 : \\
\qquad rtr_1(k) = \langle [\langle prog_{11}; mem_{11}\rangle, \ldots, \langle prog_{1n}; mem_{1n}\rangle]; \sigma_1; \mu_1\rangle \wedge \\
\qquad rtr_2(k) = \langle [\langle prog_{21}; mem_{21}\rangle, \ldots, \langle prog_{2n}; mem_{2n}\rangle]; \sigma_2; \mu_2\rangle \wedge \\
\qquad (\forall j \in \{1, \ldots, n\} : \langle prog_{1j}; mem_{1j}\rangle \approx \langle prog_{2j}; mem_{2j}\rangle) \wedge \\
\qquad \sigma_1 \simeq_{\Sigma} \sigma_2 \ \wedge \ \forall ch \in ICh : \lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)} \wedge \\
\qquad \left(k < |rtr_1| - 1 \Rightarrow \begin{pmatrix} (\forall j : \text{step from } rtr_1(k) \text{ is by } prog_{1j} \Leftrightarrow \ \text{step from } rtr_2(k) \text{ is by } prog_{2j}) \\ \wedge \ (\text{step from } rtr_1(k) \text{ is by env.} \ \Leftrightarrow \ \text{step from } rtr_2(k) \text{ is by env.}) \end{pmatrix} \right) \\
\}
\end{array}
$$

Theorem 1 immediately follows from the following lemma, which is stated using R_{dprog}.

Lemma 8. *For $dprog = [prog_1 || \ldots || prog_n]$, if $\forall i \in \{1, \ldots, n\} : LSec(prog_i)$, and dprog ensures a sound use of assumptions, then for all ξ_1, ξ_2 satisfying $\xi_1 \simeq_{\Xi} \xi_2$, all $tr_1 \in traces_{\xi_1}(dprog)$, there exists $tr_2 \in traces_{\xi_2}(dprog)$, such that $(tr_1, tr_2) \in R_{dprog}$.*

Proof. Fix $tr_1 \in traces_{\xi_1}(dprog)$. We prove that there exists $tr_2 \in traces_{\xi_2}(dprog)$ such that $(tr_1, tr_2) \in R_{dprog}$ by an induction on $|tr_1|$.

Base case. We prove the result for the case where $|tr_1| = 1$. By $tr_1 \in traces_{\xi_1}(dprog)$, we have $tr_1 = [\,\langle [\langle prog_1; mem_{\text{init}}\rangle, \ldots, \langle prog_n; mem_{\text{init}}\rangle]; \sigma_{\text{init}}\rangle\,]$. Pick $tr_2 = tr_1$. By $\forall i \in \{1, \ldots, n\} : LSec(prog_i)$, we have $\forall i \in \{1, \ldots, n\} : \langle prog_i; mem_{\text{init}}\rangle \approx \langle prog_i; mem_{\text{init}}\rangle$. Using $rtr_1 = rtr_2 = [\langle [\langle prog_1; mem_{\text{init}}\rangle, \ldots, \langle prog_n; mem_{\text{init}}\rangle]; \sigma_{\text{init}}; \mu_{\text{init}}\rangle]$, it is not difficult to obtain $(tr_1, tr_2) \in R_{dprog}$.

Inductive case. We prove the existence of a proper tr_2 in the case where $|tr_1| = n$ $(n > 1)$.

We know that there exist some tr_{10} and $gcnf_1$ such that $|tr_{10}| = n - 1$ and $tr_1 = tr_{10} \cdot gcnf_1$. By the induction hypothesis, there exists $tr_{20} \in traces_{\xi_2}(dprog)$ such that $(tr_{10}, tr_{20}) \in R_{dprog}$. By the construction of R_{dprog}, there exist rtr_{10} and rtr_{20} such that

$$tr_{10} = \mathit{trace\text{-}of}(rtr_{10}) \wedge tr_{20} = \mathit{trace\text{-}of}(rtr_{20}) \tag{3}$$

$$rtr_{10} \in rtraces_{\xi_1}(dprog) \wedge rtr_{20} \in rtraces_{\xi_2}(dprog) \tag{4}$$

$$|rtr_{10}| = |rtr_{20}| \tag{5}$$

$$\begin{aligned}
&\forall k \in \{0, \ldots, |rtr_{10}| - 1\} : \\
&\quad \exists n, prog_{11}, \ldots, prog_{1n}, prog_{21}, \ldots, prog_{2n}, mem_{11}, \ldots, mem_{1n}, mem_{21}, \ldots, mem_{2n}, \sigma_1, \sigma_2, \mu_1, \mu_2 : \\
&\quad rtr_{10}(k) = \langle [\langle prog_{11}; mem_{11}\rangle, \ldots, \langle prog_{1n}; mem_{1n}\rangle]; \sigma_1; \mu_1\rangle \wedge \\
&\quad rtr_{20}(k) = \langle [\langle prog_{21}; mem_{21}\rangle, \ldots, \langle prog_{2n}; mem_{2n}\rangle]; \sigma_2; \mu_2\rangle \wedge \\
&\quad (\forall j \in \{1, \ldots, n\} : \langle prog_{1j}; mem_{1j}\rangle \approx \langle prog_{2j}; mem_{2j}\rangle) \wedge \\
&\quad \sigma_1 \simeq_\Sigma \sigma_2 \ \wedge \ \forall ch \in ICh : \lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)} \wedge \\
&\quad \left(k < |rtr_{10}| - 1 \Rightarrow \left(\begin{array}{l} (\forall j : \text{step from } rtr_{10}(k) \text{ is by } prog_{1j} \Leftrightarrow \text{ step from } rtr_{20}(k) \text{ is by } prog_{2j}) \\ \wedge \ (\text{step from } rtr_{10}(k) \text{ is by env. } \Leftrightarrow \text{ step from } rtr_{20}(k) \text{ is by env.}) \end{array} \right) \right)
\end{aligned} \tag{6}$$

By (6), there exist $prog_{11}$,...,$prog_{1n}$, $prog_{21}$,...,$prog_{2n}$, mem_{11},...,mem_{1n}, mem_{21},...,mem_{2n}, σ_1, σ_2, μ_1, μ_2, such that

$$rtr_{10}(n-2) = \langle [\langle prog_{11}; mem_{11}\rangle, \ldots, \langle prog_{1n}; mem_{1n}\rangle]; \sigma_1; \mu_1\rangle \tag{7}$$

$$rtr_{20}(n-2) = \langle [\langle prog_{21}; mem_{21}\rangle, \ldots, \langle prog_{2n}; mem_{2n}\rangle]; \sigma_2; \mu_2\rangle \tag{8}$$

$$\forall j \in \{1, \ldots, n\} : \langle prog_{1j}; mem_{1j}\rangle \approx \langle prog_{2j}; mem_{2j}\rangle \tag{9}$$

$$\sigma_1 \simeq_\Sigma \sigma_2 \tag{10}$$

$$\forall ch \in ICh : \lfloor \sigma_1(ch) \rfloor_{\mu_1(ch)} = \lfloor \sigma_2(ch) \rfloor_{\mu_2(ch)} \tag{11}$$

We make a case analysis as to whether the step from the last configuration of tr_{10} in tr_1 is by a process or by the environment.

– Suppose the step from the last configuration of tr_{10} in tr_1 is by the j-the process. That is, we have $\langle\langle prog_{1j}; mem_{1j}\rangle; \sigma_1\rangle \to \langle\langle prog'_{1j}; mem'_{1j}\rangle; \sigma'_1\rangle$ for some $prog'_{1j}$, mem'_{1j}, and σ'_1. Hence, there exists μ'_1 such that

$$\langle\langle prog_{1j}; mem_{1j}\rangle; \sigma_1; \mu_1\rangle \to \langle\langle prog'_{1j}; mem'_{1j}\rangle; \sigma'_1; \mu'_1\rangle \tag{12}$$

by Proposition 2. Let $rtr_1 = rtr_{10} \cdot \langle [\ldots, \langle prog'_{1j}; mem'_{1j}\rangle, \ldots]; \sigma'_1; \mu'_1\rangle$. We then have $rtr_1 \in rtraces_{\xi_1}(dprog)$. We also have $tr_1 = \mathit{trace\text{-}of}(rtr_1)$.
From the hypothesis that *dprog* ensures a sound use of assumptions, (4), and (7), we have

$$(\sigma_1, \mu_1) \models \mathit{asm\text{-}of}(prog_{1j}) \wedge (\sigma_2, \mu_2) \models \mathit{asm\text{-}of}(prog_{2j}) \tag{13}$$

From (11), (13), and Proposition 1, we have

$$\sigma_1 \overset{\mathit{asm\text{-}of}(prog_{1j}) \cup \mathit{asm\text{-}of}(prog_{2j})}{=\!=\!=\!=\!=\!=\!=} \sigma_2 \tag{14}$$

By (13), we also have

$$\sigma_1 \ltimes^{NE} \mathit{asm\text{-}of}(prog_{1j}) \wedge \sigma_2 \ltimes^{NE} \mathit{asm\text{-}of}(prog_{2j}) \tag{15}$$

By (9), (10), (14), (15), and (12), there exist prog'_{2j}, mem'_{2j}, σ'_2 and μ'_2 such that

$$\langle\langle \mathit{prog}_{2j}; \mathit{mem}_{2j}\rangle; \sigma_2; \mu_2\rangle \rightarrow \langle\langle \mathit{prog}'_{2j}; \mathit{mem}'_{2j}\rangle; \sigma'_2; \mu'_2\rangle \tag{16}$$
$$\sigma'_1 \simeq_{\Sigma} \sigma'_2 \tag{17}$$
$$\langle \mathit{prog}'_{1j}; \mathit{mem}'_{1j}\rangle \approx \langle \mathit{prog}'_{2j}; \mathit{mem}'_{2j}\rangle \tag{18}$$

We proceed to show $\forall ch \in \mathit{ICh} : \lfloor \sigma'_1(ch)\rfloor_{\mu'_1(ch)} = \lfloor \sigma'_2(ch)\rfloor_{\mu'_2(ch)}$. Without loss of generality, we assume that the indices k such that the steps from $\mathit{rtr}_{10}(k)$ and $\mathit{rtr}_{20}(k)$ are taken by the j-th process are $k_1, \ldots, k_m$ for some $m > 0$, where $k_m = n - 2$. For each r, supposing the instrumentation states right after the steps from $\mathit{rtr}_{10}(k_r)$ and $\mathit{rtr}_{20}(k_r)$ are μ'_{1k_r} and μ'_{2k_r}, and the instrumentation states right before the steps from $\mathit{rtr}_{10}(k_{r+1})$ and $\mathit{rtr}_{20}(k_{r+1})$ are $\mu_{1k_{r+1}}$ and $\mu_{2k_{r+1}}$, we have $\forall ch \in \mathit{ICh} : (\mu'_{1k_r})^{\circ}(ch) \sqsubseteq \mu^{\circ}_{1k_{r+1}}(ch) \wedge (\mu'_{2k_r})^{\circ}(ch) \sqsubseteq \mu^{\circ}_{2k_{r+1}}(ch)$, by (4). Using the above conditoins together with (6), we now have two related executions of prog_j (that might be interrupted in the middle by steps from other programs or from the overall environment of the distributed program) that satisfy the conditions of Lemma 7. Thus we have

$$\forall ch \in \mathit{ICh} : \lfloor \sigma'_1(ch)\rfloor_{\mu'_1(ch)} = \lfloor \sigma'_2(ch)\rfloor_{\mu'_2(ch)} \tag{19}$$

by Lemma 7.

Let $\mathit{rtr}_2 = \mathit{rtr}_{20} \cdot \langle [\ldots, \langle \mathit{prog}'_{2j}; \mathit{mem}'_{2j}\rangle, \ldots]; \sigma'_2; \mu'_2\rangle$, and $\mathit{tr}_2 = \mathit{trace\text{-}of}(\mathit{rtr}_2)$. It is not difficult to see that $\mathit{tr}_2 \in \mathit{traces}_{\xi_2}(\mathit{dprog})$ with the help of Proposition 2. We can derive $(\mathit{tr}_1, \mathit{tr}_2) \in R_{\mathit{dprog}}$ by using rtr_1 and rtr_2.

– Suppose the step from the last configuration of tr_{10} in tr_1 is by the environment. We have

$$\mathit{tr}_1 = \mathit{trace\text{-}of}(\mathit{rtr}_{10} \cdot \langle [\langle \mathit{prog}_{11}; \mathit{mem}_{11}\rangle, \ldots, \langle \mathit{prog}_{1n}; \mathit{mem}_{1n}\rangle]; \sigma'_1; \mu_1\rangle)$$

where $\sigma'_1 = \sigma_1[\mathit{ECh} \mapsto \xi_1(\mathit{tr}_{10})(\mathit{ECh})]$. Let

$$\mathit{tr}_2 = \mathit{trace\text{-}of}(\mathit{rtr}_{20} \cdot \langle [\langle \mathit{prog}_{21}; \mathit{mem}_{21}\rangle, \ldots, \langle \mathit{prog}_{2n}; \mathit{mem}_{2n}\rangle]; \sigma'_2; \mu_2\rangle)$$

where $\sigma'_2 = \sigma_2[\mathit{ECh} \mapsto \xi_2(\mathit{tr}_{20})(\mathit{ECh})]$. From (6), we have $\mathit{tr}_{10} \simeq_{\mathit{Tr}} \mathit{tr}_{20}$. Thus we have $\sigma'_1 \simeq_{\Sigma} \sigma'_2$ from (10), and the hypothesis $\xi_1 \simeq_{\Xi} \xi_2$.

On this basis, it is not difficult to establish $(\mathit{tr}_1, \mathit{tr}_2) \in R_{\mathit{dprog}}$.

The induction on $|\mathit{tr}_1|$ above completes the proof. □

We define pairings $(\mathbb{C}[\blacktriangle_1, ..., \blacktriangle_n], \varphi(\blacktriangle_1, ..., \blacktriangle_n))$ of secure contexts and conditions, with holes $\blacktriangle_1, \ldots, \blacktriangle_n$ to be filled in with programs.

$$\begin{aligned}
&(\mathbb{C}[\blacktriangle_1, ..., \blacktriangle_n], \varphi(\blacktriangle_1, ..., \blacktriangle_n)) \ ::= \\
&(\mathsf{skip}, \mathsf{true}) \\
&\mid (x := e, \mathit{lev}\langle e\rangle \sqsubseteq \mathit{lev}(x)) \\
&\mid (\mathsf{send}(ch, e),\ \mathit{lev}\langle e\rangle \sqsubseteq \mathit{lev}^{\bullet}(ch, \emptyset)) \\
&\mid ({}^{as}\mathsf{recv}(ch, x),\ (\mathit{NE} \notin as \Rightarrow \mathit{lev}^{\circ}(ch, as) = \mathbb{L}) \wedge \mathit{lev}^{\circ}(ch, as) \sqcup \mathit{lev}^{\bullet}(ch, as) \sqsubseteq \mathit{lev}(x)) \\
&\mid ({}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}}),\ (\mathit{NE} \notin as \Rightarrow \mathit{lev}^{\circ}(ch, as) \sqsubseteq \mathit{lev}(x_{\mathrm{b}})) \wedge \mathit{lev}^{\circ}(ch, as) \sqcup \mathit{lev}^{\bullet}(ch, as) \sqsubseteq \mathit{lev}(x)) \\
&\mid (\mathsf{if}\ e\ \mathsf{then}\ \blacktriangle_1\ \mathsf{else}\ \blacktriangle_2\ \mathsf{fi},\ \mathit{lev}\langle e\rangle = \mathbb{H} \Rightarrow \blacktriangle_1 \sim \blacktriangle_2) \\
&\mid (\mathsf{while}\ e\ \mathsf{do}\ \blacktriangle_1\ \mathsf{od},\ \mathit{lev}\langle e\rangle = \mathbb{L}) \\
&\mid (\blacktriangle_1; \blacktriangle_2,\ \mathsf{true})
\end{aligned}$$

We prove the following hook-up property of $\mathit{LSec}(\cdot)$.

Proposition 3. *For all indices $n \geq 0$, pairs $(\mathbb{C}[\blacktriangle_1, ..., \blacktriangle_n], \varphi(\blacktriangle_1, ..., \blacktriangle_n))$, programs $prog_1$, ..., $prog_n$, we have $LSec(\mathbb{C}[prog_1, ..., prog_n])$, if we have $\varphi(prog_1, ..., prog_n)$ and $\forall j \in \{1, ..., n\} : LSec(prog_j)$.*

Proof. We first consider the **non-composite cases** $(\mathbb{C}[prog_1, ..., prog_n], \varphi(...)) = (prog, \varphi(...))$ where $prog$ is one of skip, $x := e$, $\mathsf{send}(ch, e)$, ${}^{as}\mathsf{recv}(ch, x)$, and ${}^{as}\mathsf{if\text{-}recv}(ch, x, x_{\mathrm{b}})$.
For each of the non-composite cases, we construct the relation

$$R_{prog} = \{(\langle prog; mem_1\rangle, \langle prog; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\} \cup \tag{20}$$

$$\{(\langle \mathsf{stop}; mem_1\rangle, \langle \mathsf{stop}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\} \tag{21}$$

It is trivial that for all mem_1 and mem_2 such that $mem_1 =_{\mathbb{L}} mem_2$, we have $\langle prog; mem_1\rangle$ R_{prog} $\langle prog; mem_2\rangle$.
We proceed to show that R_{prog} is an assumption-aware bisimulation. It is not difficult to see that R_{prog} is symmetric. Pick arbitrary pair $(\langle prog_1; mem_1\rangle, \langle prog_2; mem_2\rangle)$ from R_{prog}. We have $mem_1 =_{\mathbb{L}} mem_2$. We also have $prog_1 = \mathsf{stop} \Leftrightarrow prog_2 = \mathsf{stop}$.
Pick arbitrary channel states σ_1 and σ_2 such that $\sigma_1 \simeq_{\Sigma} \sigma_2$.
We show how the remaining proof obligations are discharged for the interesting cases for $prog$ below.

Case $\mathsf{send}(ch, e)$. We assume the hypothesis

$$lev\langle e\rangle \sqsubseteq lev^{\bullet}(ch, \emptyset) \tag{22}$$

We also assume $\sigma_1 \overset{\emptyset}{=\!=} \sigma_2$, $\sigma_1 \ltimes^{NE} \emptyset$, and $\sigma_2 \ltimes^{NE} \emptyset$.

Suppose for some σ_1' we have

$$\langle\langle \mathsf{send}(ch, e); mem_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathsf{stop}; mem_1\rangle; \sigma_1'\rangle$$

It is not difficult to see that there is some σ_2' such that

$$\langle\langle \mathsf{send}(ch, e); mem_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathsf{stop}; mem_2\rangle; \sigma_2'\rangle$$

We have $\langle \mathsf{stop}; mem_1\rangle$ R_{prog} $\langle \mathsf{stop}; mem_2\rangle$ by $mem_1 =_{\mathbb{L}} mem_2$. We proceed to show $\sigma_1' \simeq_{\Sigma} \sigma_2'$ with a case analysis on whether $ch \in ECh$.

Sub-case $ch \in ECh$. We make a further case analysis on the class of ch.

- Suppose $ch \in PriCh$. We have $ob(ch, \sigma_1') = ob(ch, \sigma_2') = \circledcirc$. It is not difficult to see $\sigma_1' \simeq_{\Sigma} \sigma_2'$, since the message queues for the other channels are unchanged in σ_1' and σ_2'.
- Suppose $ch \in EncCh$. From $\sigma_1 \simeq_{\Sigma} \sigma_2$, we have $ob(ch, \sigma_1) = ob(ch, \sigma_2) = \odot^n$ for some n. We then have $ob(ch, \sigma_1') = ob(ch, \sigma_2') = \odot^{n+1}$. It is not difficult to see that $\sigma_1' \simeq_{\Sigma} \sigma_2'$.
- Suppose $ch \in PubCh$. From $\sigma_1 \simeq_{\Sigma} \sigma_2$, we have $ob(ch, \sigma_1) = ob(ch, \sigma_2) = vlst$ for some list $vlst$ of values. We have $lev^{\bullet}(ch, \emptyset) = \mathbb{L}$. We thus have $lev\langle e\rangle = \mathbb{L}$ by (22). Thus we have $[\![e]\!]_{mem_1} = [\![e]\!]_{mem_2}$ by $mem_1 =_{\mathbb{L}} mem_2$. We have $ob(ch, \sigma_1') = ob(ch, \sigma_2') = vlst \cdot v$, where $v = [\![e]\!]_{mem_1} = [\![e]\!]_{mem_2}$. It is not difficult to see that $\sigma_1' \simeq_{\Sigma} \sigma_2'$.

Sub-case $ch \in ICh$. Since the message queues of all the external channels are unaffected, we trivially have $\sigma_1' \simeq_{\Sigma} \sigma_2'$.

Case ${}^{as}\mathsf{recv}(ch, x)$. We assume the following hypotheses

$$NE \notin as \Rightarrow lev^{\circ}(ch, as) = \mathbb{L} \tag{23}$$

$$lev^{\circ}(ch, as) \sqcup lev^{\bullet}(ch, as) \sqsubseteq lev(x) \tag{24}$$

We also assume

$$\sigma_1 \overset{as \times \{ch\}}{=\!=\!=\!=} \sigma_2 \tag{25}$$

$$\sigma_1 \ltimes^{NE} (as \times \{ch\}) \;\wedge\; \sigma_2 \ltimes^{NE} (as \times \{ch\}) \tag{26}$$

Suppose for some mem_1' and σ_1' we have

$$\langle\langle {}^{as}\mathsf{recv}(ch, x); mem_1\rangle; \sigma_1\rangle \to \langle\langle \mathsf{stop}; mem_1'\rangle; \sigma_1'\rangle \tag{27}$$

We make a case analysis on whether $ch \in ECh$ to discharge the remaining proof obligations.

Sub-case $ch \in ECh$. We make a further case analysis on whether $NE \in as$.

– Suppose $NE \in as$.

By (26), we have $|\sigma_2(ch)| > 0$. Hence there exists some σ_2' and mem_2' such that

$$\langle\langle {}^{as}\mathsf{recv}(ch, x); mem_2\rangle; \sigma_2\rangle \to \langle\langle \mathsf{stop}; mem_2'\rangle; \sigma_2'\rangle$$

We make a further case analysis on the class of ch.

- Suppose $ch \in PriCh$. By (24), we have $lev(x) = \mathbb{H}$. Thus we vacuously have $lev(x) = \mathbb{L} \Rightarrow mem_1'(x) = mem_2'(x)$. It is not difficult to establish $mem_1' =_{\mathbb{L}} mem_2'$ since the values for the other variables are unchanged in mem_1' and mem_2'. Hence we have $\langle \mathsf{stop}; mem_1'\rangle\; R_{prog}\; \langle \mathsf{stop}; mem_2'\rangle$.

  We have $ob(ch, \sigma_1') = ob(ch, \sigma_2') = \circledcirc$. Thus, it is not difficult to establish $\sigma_1' \simeq_{\Sigma} \sigma_2'$, since the message queues for the other channels are unchanged in σ_1' and σ_2'.

- Suppose $ch \in EncCh$. That $\langle \mathsf{stop}; mem_1'\rangle\; R_{prog}\; \langle \mathsf{stop}; mem_2'\rangle$ can be established analogously to the case where $ch \in PriCh$.

  From $\sigma_1 \simeq_{\Sigma} \sigma_2$, we have $ob(ch, \sigma_1) = ob(ch, \sigma_2) = \odot^{n}$ for some n, where $n > 0$ since $NE \in as$. Hence, $ob(ch, \sigma_1') = ob(ch, \sigma_2') = \odot^{n-1}$. On this basis, it is not difficult to see that $\sigma_1' \simeq_{\Sigma} \sigma_2'$.

- Suppose $ch \in PubCh$.

  From $\sigma_1 \simeq_{\Sigma} \sigma_2$, we have $ob(ch, \sigma_1) = ob(ch, \sigma_2) = vlst$, where $vlst$ is a non-empty list of values since $NE \in as$.

  Suppose $vlst = v{\cdot}vlst'$ for some $v \in Val$ and $vlst' \in Val^{*}$. We have $mem_1'(x) = mem_2'(x) = v$. On this basis, it is not difficult to see that $mem_1' =_{\mathbb{L}} mem_2'$ holds. Hence we have $\langle \mathsf{stop}; mem_1'\rangle\; R_{prog}\; \langle \mathsf{stop}; mem_2'\rangle$. In addition, we have $\sigma_1'(ch) = \sigma_2'(ch) = vlst'$. On this basis, it is not difficult to establish $\sigma_1' \simeq_{\Sigma} \sigma_2'$.

– Suppose $NE \notin as$. By (23), we have $lev^{\circ}(ch, as) = \mathbb{L}$. Hence $ch \in PubCh$ or $ch \in EncCh$. By (27), we have $|\sigma_1(ch)| > 0$. Thus we have $|\sigma_2(ch)| > 0$ by $\sigma_1 \simeq_{\Sigma} \sigma_2$ and the possible classes of ch. Hence there exists some σ_2' and mem_2' such that

$$\langle\langle {}^{as}\mathsf{recv}(ch, x); mem_2\rangle; \sigma_2\rangle \to \langle\langle \mathsf{stop}; mem_2'\rangle; \sigma_2'\rangle$$

The remaining proof obligations can be discharged by a case analysis on whether $ch \in PubCh$ or $ch \in EncCh$, analogously to the case where $NE \in as$.

Sub-case $ch \in ICh$. We make a further case analysis on whether $NE \in as$.

– Suppose $NE \in as$.

By (26), we have $|\sigma_2(ch)| > 0$. Hence there exist some σ_2' and mem_2' such that

$$\langle\langle {}^{as}\mathsf{recv}(ch, x); mem_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathsf{stop}; mem_2'\rangle; \sigma_2'\rangle$$

Since no message queues for the external channels are affected in σ_1' or σ_2', we trivially have $\sigma_1' \simeq_\Sigma \sigma_2'$.

We make a further case analysis on whether $\mathbb{L}^\circ$ and $\mathbb{L}^\bullet$ are in as, to show $\langle \mathsf{stop}; mem_1'\rangle\ R_{prog}\ \langle \mathsf{stop}; mem_2'\rangle$, which boils down to showing $mem_1' =_\mathbb{L} mem_2'$.

- Suppose $\mathbb{L}^\bullet \notin as$ or $\mathbb{L}^\circ \notin as$.

  From (24), we have $lev(x) = \mathbb{H}$. Hence it vacuously holds that $lev(x) = \mathbb{L} \Rightarrow mem_1'(x) = mem_2'(x)$. Thus we have $mem_1' =_\mathbb{L} mem_2'$ from $mem_1 =_\mathbb{L} mem_2$, and the fact that the values of the other variables are unchanged in mem_1' and mem_2'.

- Suppose $\mathbb{L}^\bullet \in as$ and $\mathbb{L}^\circ \in as$.

  By (26) and $NE \in as$, we have $|\sigma_1(ch)| > 0$, and $|\sigma_2(ch)| > 0$. Hence we can obtain $\mathit{first}(\sigma_1(ch)) = \mathit{first}(\sigma_2(ch))$ using (25). Hence we can derive $mem_1'(x) = mem_2'(x)$. On this basis, it is not difficult to derive $mem_1' =_\mathbb{L} mem_2'$.

– Suppose $NE \notin as$.

Using (23), we derive $\mathbb{L}^\circ \in as$. By (27), we have $|\sigma_1(ch)| > 0$. Thus we have $|\sigma_2(ch)| > 0$ by (25). Hence there exist some σ_2' and mem_2' such that

$$\langle\langle {}^{as}\mathsf{recv}(ch, x); mem_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathsf{stop}; mem_2'\rangle; \sigma_2'\rangle$$

Since no message queues for the external channels are affected in σ_1' or σ_2', we trivially have $\sigma_1' \simeq_\Sigma \sigma_2'$.

We make a case analysis on whether $\mathbb{L}^\bullet \in as$ to show $\langle \mathsf{stop}; mem_1'\rangle\ R_{prog}\ \langle \mathsf{stop}; mem_2'\rangle$, which boils down to showing $mem_1' =_\mathbb{L} mem_2'$.

- Suppose $\mathbb{L}^\bullet \notin as$.

  By (24), we have $lev(x) = \mathbb{H}$. Hence we have $lev(x) = \mathbb{L} \Rightarrow mem_1'(x) = mem_2'(x)$. On this basis, $mem_1' =_\mathbb{L} mem_2'$ can be derived.

- Suppose $\mathbb{L}^\bullet \in as$.

  By (25), $|\sigma_1(ch)| > 0$, and $|\sigma_2(ch)| > 0$, we have $\mathit{first}(\sigma_1(ch)) = \mathit{first}(\sigma_2(ch))$. Hence, we have $mem_1'(x) = mem_2'(x)$. On this basis, $mem_1' =_\mathbb{L} mem_2'$ can be derived.

Case ${}^{as}\mathsf{if\text{-}recv}(ch, x, x_\mathrm{b})$. The reasoning can be structured similarly to the case for ${}^{as}\mathsf{recv}(ch, x)$.

For each of the **composite cases**, we construct a specific relation that is shown to be an assumption-aware bisimulation, which serves to justify the local security of the corresponding program.

Case $\mathsf{if}\ e\ \mathsf{then}\ prog_{\mathrm{a}}\ \mathsf{else}\ prog_{\mathrm{b}}\ \mathsf{fi}$. We construct the following relation where $\underline{\mathsf{if}}$ is a shorthand for the program $\mathsf{if}\ e\ \mathsf{then}\ prog_{\mathrm{a}}\ \mathsf{else}\ prog_{\mathrm{b}}\ \mathsf{fi}$.

$$R_{\underline{\mathsf{if}}} = \{(\langle \underline{\mathsf{if}}; mem_1\rangle, \langle \underline{\mathsf{if}}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\} \cup \approx$$

For all mem_1, mem_2 such that $mem_1 =_{\mathbb{L}} mem_2$, we have $\langle \underline{\mathsf{if}}; mem_1\rangle\ R_{\underline{\mathsf{if}}}\ \langle \underline{\mathsf{if}}; mem_2\rangle$. We proceed to show that $R_{\underline{\mathsf{if}}}$ is an assumption-aware bisimulation. It is obvious that $R_{\underline{\mathsf{if}}}$ is symmetric. Pick an arbitrary pair $p = (\langle prog_1; mem_1\rangle, \langle prog_2; mem_2\rangle)$ from $R_{\underline{\mathsf{if}}}$. We can also derive $mem_1 =_{\mathbb{L}} mem_2$ and $prog_1 = \mathsf{stop} \Leftrightarrow prog_2 = \mathsf{stop}$.

To discharge the remaining proof obligations, we make a case analysis on which part of $R_{\underline{\mathsf{if}}}$ p belongs to. We assume the following hypotheses.

$$LSec(prog_{\mathrm{a}}) \tag{28}$$
$$LSec(prog_{\mathrm{b}}) \tag{29}$$
$$lev\langle e\rangle = \mathbb{H} \Rightarrow prog_{\mathrm{a}} \sim prog_{\mathrm{b}} \tag{30}$$

Sub-case $p \in \{(\langle \underline{\mathsf{if}}; mem_1\rangle, \langle \underline{\mathsf{if}}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$. We have $prog_1 = prog_2 = \underline{\mathsf{if}}$.

Pick arbitrary channel states σ_1 and σ_2 such that $\sigma_1 \simeq_{\Sigma} \sigma_2$, $\sigma_1 \overset{\emptyset}{=\!=} \sigma_2$, $\sigma_1 \ltimes^{NE} \emptyset$, and $\sigma_2 \ltimes^{NE} \emptyset$. Suppose without loss of generality that

$$\langle\langle \underline{\mathsf{if}}; mem_1\rangle; \sigma_1\rangle \to \langle\langle prog_{\mathrm{a}}; mem_1\rangle; \sigma_1\rangle \tag{31}$$

We make a case analysis on whether $[\![e]\!]_{mem_2}$ is equal to $[\![e]\!]_{mem_1}$.

– Suppose $[\![e]\!]_{mem_2} \neq [\![e]\!]_{mem_1}$. We have

$$\langle\langle \underline{\mathsf{if}}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog_{\mathrm{b}}; mem_2\rangle; \sigma_2\rangle$$

From $mem_1 =_{\mathbb{L}} mem_2$, we have $lev\langle e\rangle = \mathbb{H}$. Hence we have $prog_{\mathrm{a}} \sim prog_{\mathrm{b}}$ by (30). Hence we have $\langle prog_{\mathrm{a}}; mem_1\rangle \approx \langle prog_{\mathrm{b}}; mem_2\rangle$, which gives $\langle prog_{\mathrm{a}}; mem_1\rangle\ R_{\underline{\mathsf{if}}}\ \langle prog_{\mathrm{b}}; mem_2\rangle$. We also have $\sigma_1 \simeq_{\Sigma} \sigma_2$, which is preserved from before.

– Suppose $[\![e]\!]_{mem_2} = [\![e]\!]_{mem_1}$. We have

$$\langle\langle \underline{\mathsf{if}}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog_{\mathrm{a}}; mem_2\rangle; \sigma_2\rangle$$

By (28) and $mem_1 =_{\mathbb{L}} mem_2$, we have $\langle prog_{\mathrm{a}}; mem_1\rangle \approx \langle prog_{\mathrm{a}}; mem_2\rangle$, which directly gives $\langle prog_{\mathrm{a}}; mem_1\rangle\ R_{\underline{\mathsf{if}}}\ \langle prog_{\mathrm{a}}; mem_2\rangle$. We also have $\sigma_1 \simeq_{\Sigma} \sigma_2$, which is preserved from before.

Sub-case $p \in \approx$. Straightforward from the fact that $\approx$ is an assumption-aware bisimulation.

Case $\mathsf{while}\ e\ \mathsf{do}\ prog_{\mathrm{a}}\ \mathsf{od}$. We construct the relation $R_{\underline{\mathsf{while}}} = R_0 \cup R_1 \cup R_2$, where $\underline{\mathsf{while}}$ is a shorthand for $\mathsf{while}\ e\ \mathsf{do}\ prog_{\mathrm{a}}\ \mathsf{od}$, and

$$R_0 = \{(\langle \underline{\mathsf{while}}; mem_1\rangle, \langle \underline{\mathsf{while}}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$
$$R_1 = \{(\langle prog_1; \underline{\mathsf{while}}; mem_1\rangle, \langle prog_2; \underline{\mathsf{while}}; mem_2\rangle) \mid \langle prog_1; mem_1\rangle \approx \langle prog_2; mem_2\rangle\}$$
$$R_2 = \{(\langle \mathsf{stop}; mem_1\rangle, \langle \mathsf{stop}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$

For all mem_1 and mem_2 such that $mem_1 =_{\mathbb{L}} mem_2$, we have $\langle \underline{\mathsf{while}}; mem_1\rangle\ R_{\underline{\mathsf{while}}}\ \langle \underline{\mathsf{while}}; mem_2\rangle$. We proceed to show that $R_{\underline{\mathsf{while}}}$ is an assumption-aware bisimulation. We have that $R_{\underline{\mathsf{while}}}$ is symmetric. Pick an arbitrary pair $p = (\langle \mathit{prog}_1; \mathit{mem}_1\rangle, \langle \mathit{prog}_2; \mathit{mem}_2\rangle)$ from $R_{\underline{\mathsf{while}}}$. We have $\mathit{mem}_1 =_{\mathbb{L}} \mathit{mem}_2$, and $\mathit{prog}_1 \neq \mathsf{stop} \Leftrightarrow \mathit{prog}_2 \neq \mathsf{stop}$.

To discharge the remaining proof obligations, we make a case analysis on which part of $R_{\underline{\mathsf{while}}}$ p belongs to. We assume the following hypotheses.

$$\mathit{LSec}(\mathit{prog}_{\mathrm{a}}) \tag{32}$$

$$\mathit{lev}\langle e\rangle = \mathbb{L} \tag{33}$$

Sub-case $p \in R_0$. Pick arbitrary channel states σ_1 and σ_2 such that $\sigma_1 \simeq_{\Sigma} \sigma_2$, $\sigma_1 \overset{\emptyset}{=\!=} \sigma_2$, $\sigma_1 \ltimes^{NE} \emptyset$, and $\sigma_2 \ltimes^{NE} \emptyset$.

We make a further case analysis on whether $\underline{\mathsf{while}}$ continues or terminates in one step.

- Suppose $\langle\langle \underline{\mathsf{while}}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathit{prog}_{\mathrm{a}}; \underline{\mathsf{while}}; \mathit{mem}_1\rangle; \sigma_1\rangle$.

  By (33) and $\mathit{mem}_1 =_{\mathbb{L}} \mathit{mem}_2$, we have $[\![e]\!]_{\mathit{mem}_2} = [\![e]\!]_{\mathit{mem}_1}$. Hence we have

  $$\langle\langle \underline{\mathsf{while}}; \mathit{mem}_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathit{prog}_{\mathrm{a}}; \underline{\mathsf{while}}; \mathit{mem}_2\rangle; \sigma_2\rangle$$

  By (32) and $\mathit{mem}_1 =_{\mathbb{L}} \mathit{mem}_2$, we have $\langle \mathit{prog}_{\mathrm{a}}; \mathit{mem}_1\rangle \approx \langle \mathit{prog}_{\mathrm{a}}; \mathit{mem}_2\rangle$. Hence, we have $\langle \mathit{prog}_{\mathrm{a}}; \underline{\mathsf{while}}; \mathit{mem}_1\rangle\ R_{\underline{\mathsf{while}}}\ \langle \mathit{prog}_{\mathrm{a}}; \underline{\mathsf{while}}; \mathit{mem}_2\rangle$. We also have $\sigma_1 \simeq_{\Sigma} \sigma_2$, which is trivially preserved.

- Suppose $\langle\langle \underline{\mathsf{while}}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathsf{stop}; \mathit{mem}_1\rangle; \sigma_1\rangle$. This case is straightforward.

Sub-case $p \in R_1$. We have $\mathit{prog}_1 = \mathit{prog}_{10}; \underline{\mathsf{while}}$, and $\mathit{prog}_2 = \mathit{prog}_{20}; \underline{\mathsf{while}}$ for some prog_{10} and prog_{20} such that

$$\langle \mathit{prog}_{10}; \mathit{mem}_1\rangle \approx \langle \mathit{prog}_{20}; \mathit{mem}_2\rangle \tag{34}$$

Pick arbitrary channel states σ_1 and σ_2 such that

$$\begin{aligned} &\sigma_1 \simeq_{\Sigma} \sigma_2 \ \wedge\ \sigma_1 \overset{\mathit{asm\text{-}of}(\mathit{prog}_{10}) \cup \mathit{asm\text{-}of}(\mathit{prog}_{20})}{=\!=\!=\!=\!=\!=\!=\!=} \sigma_2 \\ &\wedge\ \sigma_1 \ltimes^{NE} \mathit{asm\text{-}of}(\mathit{prog}_{10}) \ \wedge\ \sigma_2 \ltimes^{NE} \mathit{asm\text{-}of}(\mathit{prog}_{20}) \end{aligned} \tag{35}$$

We make a further case analysis on whether prog_{10} continues or terminates after one step.

- Suppose $\langle\langle \mathit{prog}_{10}; \underline{\mathsf{while}}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathit{prog}'_{10}; \underline{\mathsf{while}}; \mathit{mem}'_1\rangle; \sigma'_1\rangle$, with $\langle\langle \mathit{prog}_{10}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathit{prog}'_{10}; \mathit{mem}'_1\rangle; \sigma'_1\rangle$, and $\mathit{prog}'_{10} \neq \mathsf{stop}$. By (34) and (35), there exist prog'_{20}, mem'_2, and σ'_2 such that $\langle\langle \mathit{prog}_{20}; \mathit{mem}_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathit{prog}'_{20}; \mathit{mem}'_2\rangle; \sigma'_2\rangle$, $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle \mathit{prog}'_{10}; \mathit{mem}'_1\rangle \approx \langle \mathit{prog}'_{20}; \mathit{mem}'_2\rangle$. Hence, we have $\mathit{prog}'_{10} = \mathsf{stop} \Leftrightarrow \mathit{prog}'_{20} = \mathsf{stop}$, which gives $\mathit{prog}'_{20} \neq \mathsf{stop}$.

  Thus, we have $\langle\langle \mathit{prog}_{20}; \underline{\mathsf{while}}; \mathit{mem}_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathit{prog}'_{20}; \underline{\mathsf{while}}; \mathit{mem}'_2\rangle; \sigma'_2\rangle$, $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle \mathit{prog}'_{10}; \underline{\mathsf{while}}; \mathit{mem}'_1\rangle\ R_{\underline{\mathsf{while}}}\ \langle \mathit{prog}'_{20}; \underline{\mathsf{while}}; \mathit{mem}'_2\rangle$.

- Suppose $\langle\langle \mathit{prog}_{10}; \underline{\mathsf{while}}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \underline{\mathsf{while}}; \mathit{mem}'_1\rangle; \sigma'_1\rangle$, with $\langle\langle \mathit{prog}_{10}; \mathit{mem}_1\rangle; \sigma_1\rangle \rightarrow \langle\langle \mathsf{stop}; \mathit{mem}'_1\rangle; \sigma'_1\rangle$. By (34) and (35), we know that there exist prog'_{20}, mem'_2, and σ'_2 such that $\langle\langle \mathit{prog}_{20}; \mathit{mem}_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \mathit{prog}'_{20}; \mathit{mem}'_2\rangle; \sigma'_2\rangle$, $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle \mathsf{stop}; \mathit{mem}'_1\rangle \approx \langle \mathit{prog}'_{20}; \mathit{mem}'_2\rangle$. Hence, we have $\mathit{mem}'_1 =_{\mathbb{L}} \mathit{mem}'_2$, and $\mathit{prog}'_{20} = \mathsf{stop}$.

  Thus, we have $\langle\langle \mathit{prog}_{20}; \underline{\mathsf{while}}; \mathit{mem}_2\rangle; \sigma_2\rangle \rightarrow \langle\langle \underline{\mathsf{while}}; \mathit{mem}'_2\rangle; \sigma'_2\rangle$, $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, as well as $\langle \underline{\mathsf{while}}; \mathit{mem}'_1\rangle\ R_{\underline{\mathsf{while}}}\ \langle \underline{\mathsf{while}}; \mathit{mem}'_2\rangle$.

Sub-case $p \in R_2$. Trivial.

Case $prog_{\mathrm{a}}; prog_{\mathrm{b}}$. We construct the relation $R_; = R_0 \cup \approx$, where

$$R_0 = \{(\langle prog_{10}; prog_{\mathrm{b}}; mem_1\rangle, \langle prog_{20}; prog_{\mathrm{b}}; mem_2\rangle) \mid \langle prog_{10}; mem_1\rangle \approx \langle prog_{20}; mem_2\rangle\}$$

We assume the following hypotheses

$$LSec(prog_{\mathrm{a}}) \tag{36}$$

$$LSec(prog_{\mathrm{b}}) \tag{37}$$

Pick arbitrary mem_1 and mem_2 such that $mem_1 =_{\mathbb{L}} mem_2$. By (36), we have $\langle prog_{\mathrm{a}}; mem_1\rangle \approx \langle prog_{\mathrm{a}}; mem_2\rangle$. Hence we have $\langle prog_{\mathrm{a}}; prog_{\mathrm{b}}; mem_1\rangle \approx \langle prog_{\mathrm{a}}; prog_{\mathrm{b}}; mem_2\rangle$. We proceed to show that $R_;$ is an assumption-aware bisimulation. It is obvious that $R_;$ is symmetric. Pick arbitrary pair $p = (\langle prog_1; mem_1\rangle, \langle prog_2; mem_2\rangle)$ from $R;$. We have $mem_1 =_{\mathbb{L}} mem_2$, and $prog_1 = \mathsf{stop} \Leftrightarrow prog_2 = \mathsf{stop}$.

We discharge the remaining proof obligations with a case analysis on which part of $R_;$ p belongs.

Sub-case $p \in R_0$. We have $prog_1 = prog_{10}; prog_{\mathrm{b}}$, and $prog_2 = prog_{20}; prog_{\mathrm{b}}$, for some $prog_{10}$ and $prog_{20}$ such that

$$\langle prog_{10}; mem_1\rangle \approx \langle prog_{20}; mem_2\rangle \tag{38}$$

Pick arbitrary channel states σ_1 and σ_2 such that

$$\begin{aligned}&\sigma_1 \simeq_{\Sigma} \sigma_2 \ \wedge\ \sigma_1 \overset{asm\text{-}of(prog_{10}) \cup asm\text{-}of(prog_{20})}{=\!=\!=\!=\!=\!=} \sigma_2 \\ &\wedge\ \sigma_1 \ltimes^{NE} asm\text{-}of(prog_{10}) \ \wedge\ \sigma_2 \ltimes^{NE} asm\text{-}of(prog_{20})\end{aligned} \tag{39}$$

We make a case analysis on whether $prog_{10}$ is exhausted in the next step of $prog_{10}; prog_{\mathrm{b}}$.

- Suppose for some $prog'_{10}$, mem'_1, and σ'_1, we have

  $$\langle\langle prog_{10}; prog_{\mathrm{b}}; mem_1\rangle; \sigma_1\rangle \to \langle\langle prog'_{10}; prog_{\mathrm{b}}; mem'_1\rangle; \sigma'_1\rangle$$

  with $\langle\langle prog_{10}; mem_1\rangle; \sigma_1\rangle \to \langle\langle prog'_{10}; mem'_1\rangle; \sigma'_1\rangle$, and $prog'_{10} \neq \mathsf{stop}$.

  By (38), and (39), there exist $prog'_{20}$, mem'_2, σ'_2 such that

  $$\langle\langle prog_{20}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog'_{20}; mem'_2\rangle; \sigma'_2\rangle$$

  $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle prog'_{10}; mem'_1\rangle \approx \langle prog'_{20}; mem'_2\rangle$. Hence $prog'_{10} = \mathsf{stop} \Leftrightarrow prog'_{20} = \mathsf{stop}$. Hence $prog'_{20} \neq \mathsf{stop}$.

  Thus, we have $\langle\langle prog_{20}; prog_{\mathrm{b}}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog'_{20}; prog_{\mathrm{b}}; mem'_2\rangle; \sigma'_2\rangle$, $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle prog'_{10}; prog_{\mathrm{b}}; mem'_1\rangle \approx \langle prog'_{20}; prog_{\mathrm{b}}; mem'_2\rangle$.

- Suppose for some mem'_1, and σ'_1, we have

  $$\langle\langle prog_{10}; prog_{\mathrm{b}}; mem_1\rangle; \sigma_1\rangle \to \langle\langle prog_{\mathrm{b}}; mem'_1\rangle; \sigma'_1\rangle$$

  with $\langle\langle prog_{10}; mem_1\rangle; \sigma_1\rangle \to \langle\langle \mathsf{stop}; mem'_1\rangle; \sigma'_1\rangle$.

  By (38), and (39), there exist $prog'_{20}$, mem'_2, σ'_2 such that

  $$\langle\langle prog_{20}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog'_{20}; mem'_2\rangle; \sigma'_2\rangle$$

  $\sigma'_1 \simeq_{\Sigma} \sigma'_2$, and $\langle \mathsf{stop}; mem'_1\rangle \approx \langle prog'_{20}; mem'_2\rangle$. Hence $mem'_1 =_{\mathbb{L}} mem'_2$, and $prog'_{20} = \mathsf{stop}$. Hence we have $\langle\langle prog_{20}; prog_{\mathrm{b}}; mem_2\rangle; \sigma_2\rangle \to \langle\langle prog_{\mathrm{b}}; mem'_2\rangle; \sigma'_2\rangle$. By (37) and $mem'_1 =_{\mathbb{L}} mem'_2$, we also have $\langle prog_{\mathrm{b}}; mem'_1\rangle \approx \langle prog_{\mathrm{b}}; mem'_2\rangle$. Thus $\langle prog_{\mathrm{b}}; mem'_1\rangle \ R_; \ \langle prog_{\mathrm{b}}; mem'_2\rangle$.

Sub-case $p \in \approx$. Trivial.

The above case analysis on the pairs of programs with holes and conditions completes the proof. $\square$

We next prove Theorem 2.

Theorem 2. *If $lev \vdash prog$, then $LSec(prog)$.*

Proof. The proof is by induction on the derivation of $lev \vdash prog$. In each case we build on the hook-up property of $LSec(\cdot)$ (Proposition 3) to obtain the desired result. We only present one base case and one inductive case below.

Case ${}^{as}\mathsf{recv}(ch, x)$: By $lev \vdash {}^{as}\mathsf{recv}(ch, x)$, we have

$$NE \notin as \Rightarrow lev^{\circ}(ch, as) = \mathbb{L} \tag{40}$$

$$lev^{\circ}(ch, as) \sqcup lev^{\bullet}(ch, as) \sqsubseteq lev(x) \tag{41}$$

Instantiating Proposition 3 with $n = 0$, and the fourth pair of context and condition, we know that $LSec({}^{as}\mathsf{recv}(ch, x))$ holds, if $(NE \notin as \Rightarrow lev^{\circ}(ch, as) = \mathbb{L}) \wedge lev^{\circ}(ch, as) \sqcup lev^{\bullet}(ch, as) \sqsubseteq lev(x)$, which is ensured by (40), (41).

Case $\mathsf{if}\ e\ \mathsf{then}\ prog_1\ \mathsf{else}\ prog_2\ \mathsf{fi}$: By $lev \vdash \mathsf{if}\ e\ \mathsf{then}\ prog_1\ \mathsf{else}\ prog_2\ \mathsf{fi}$, we have

$$lev \vdash prog_1 \tag{42}$$

$$lev \vdash prog_2 \tag{43}$$

$$lev\langle e\rangle = \mathbb{H} \Rightarrow prog_1 \sim prog_2 \tag{44}$$

Instantiating Proposition 3 with $n = 2$, the sixth pair of context and condition, $prog_1$, and $prog_2$, we have $LSec(\mathsf{if}\ e\ \mathsf{then}\ prog_1\ \mathsf{else}\ prog_2\ \mathsf{fi})$ holds if $lev\langle e\rangle = \mathbb{H} \Rightarrow prog_1 \sim prog_2$, $LSec(prog_1)$, and $LSec(prog_2)$. Using the induction hypothesis on (42) and (43), we can derive $LSec(prog_1)$ and $LSec(prog_2)$. Combining this with (44), it is not difficult to see that all the conditions for $LSec(\mathsf{if}\ e\ \mathsf{then}\ prog_1\ \mathsf{else}\ prog_2\ \mathsf{fi})$ to be established are satisfied.

In the remaining cases, the premises of the typing rules give rise to the premises of the corresponding statements in the hook-up property in an analogous fashion to the cases above. The induction completes the proof of this theorem. $\square$

$$\frac{\begin{array}{c} \mathit{NE} \notin \mathit{as} \Rightarrow \mathit{lev}^{\circ}(\mathit{ch}, \mathit{as}) = \mathbb{L} \\ \mathit{lev}^{\circ}(\mathit{ch}, \mathit{as}) \sqcup \mathit{lev}^{\bullet}(\mathit{ch}, \mathit{as}) \sqsubseteq \mathit{lev}(x) \end{array}}{\mathit{lev} \vdash_{\mathrm{S}} {}^{\mathit{as}}\mathsf{recv}(\mathit{ch}, x)} \qquad \frac{\begin{array}{c} \mathit{NE} \notin \mathit{as} \Rightarrow \mathit{lev}^{\circ}(\mathit{ch}, \mathit{as}) \sqsubseteq \mathit{lev}(x_{\mathrm{b}}) \\ \mathit{lev}^{\circ}(\mathit{ch}, \mathit{as}) \sqcup \mathit{lev}^{\bullet}(\mathit{ch}, \mathit{as}) \sqsubseteq \mathit{lev}(x) \end{array}}{\mathit{lev} \vdash_{\mathrm{S}} {}^{\mathit{as}}\mathsf{if\text{-}recv}(\mathit{ch}, x, x_{\mathrm{b}})}$$

$$\frac{\mathit{lev}\langle e \rangle \sqsubseteq \mathit{lev}^{\bullet}(\mathit{ch}, \emptyset)}{\mathit{lev} \vdash_{\mathrm{S}} \mathsf{send}(\mathit{ch}, e)} \qquad \frac{\mathit{lev}\langle e \rangle \sqsubseteq \mathit{lev}(x)}{\mathit{lev} \vdash_{\mathrm{S}} x := e} \qquad \frac{}{\mathit{lev} \vdash_{\mathrm{S}} \mathsf{skip}}$$

$$\frac{\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_1 \quad \mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_2 \quad \mathit{lev}\langle e \rangle = \mathbb{H} \Rightarrow \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2)}{\mathit{lev} \vdash_{\mathrm{S}} \mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi}}$$

$$\frac{\mathit{lev}\langle e \rangle = \mathbb{L} \quad \mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}}{\mathit{lev} \vdash_{\mathrm{S}} \mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}} \qquad \frac{\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_1 \quad \mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_2}{\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_1; \mathit{prog}_2}$$

$$\mathit{lev}^{\bullet}(\mathit{ch}, \mathit{as}) = \begin{cases} \mathbb{L} & \text{if } \mathit{ch} \in \mathit{ICh} \wedge \mathbb{L}^{\bullet} \in \mathit{as} \\ & \quad \vee \mathit{ch} \in \mathit{PubCh} \\ \mathbb{H} & \text{otherwise} \end{cases} \qquad \mathit{lev}^{\circ}(\mathit{ch}, \mathit{as}) = \begin{cases} \mathbb{L} & \text{if } \mathit{ch} \in \mathit{ICh} \wedge \mathbb{L}^{\circ} \in \mathit{as} \vee \\ & \quad \mathit{ch} \in \mathit{PubCh} \vee \mathit{ch} \in \mathit{EncCh} \\ \mathbb{H} & \text{otherwise} \end{cases}$$

Fig. 1: The Fully Syntactic Security Type System

3 Fully Syntactic Security Type System

We present a security type system that is sound wrt. local security, yet has no semantic side condition. This type system establishes the judgment $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}$, saying that the program prog is (fully syntactically) typable in the environment lev.
The typing rules are presented in Fig. 1. Except for the if rule, all the rules in Fig. 1 are identical to the rules of our original type system (in Fig. 1 of [1]) for the same language constructs.
For the if rule, the side condition $\mathit{lev}\langle e \rangle = \mathbb{H} \Rightarrow \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2)$ is used to replace the semantic side condition $\mathit{lev}\langle e \rangle = \mathbb{H} \Rightarrow \mathit{prog}_1 \sim \mathit{prog}_2$ in our original type system. Here, the function $\mathit{low\text{-}slice\text{-}of} : \mathit{Prog} \to \mathit{LS} \cup \{\bot\}$ gives the *low slice* of each program (defined in Fig. 2). The set LS of low slices is the same as the set of programs, except that the set of expressions is extended with the special element $\diamond$, which represents an expression that is not contained in a low slice. Intuitively, the low slice of a program is a slice that captures the effects on the low parts of the memory, and the public content or presence of messages over communication channels, created by executing the program. Hence, the side condition $\mathit{lev}\langle e \rangle = \mathbb{H} \Rightarrow \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2)$ captures the requirement that the execution of the two branches of the if should have the same effects on the public parts of the memory and channel states.
We have the following theoretical results about the syntactic security type system.

Theorem 3. *If $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}$, then $\mathit{lev} \vdash \mathit{prog}$.*

Corollary 1. *If $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}$, then $\mathit{LSec}(\mathit{prog})$.*

The theorem says that the fully syntactic security type system is sound wrt. the security type system presented in Fig. 1 of the paper. The corollary says that the fully syntactic security type system is sound wrt. local security. Hence, the fully syntactic security type system can be used as a replacement of the security type system presented in the paper to the benefit of fully operational type checking, and verification of local security.
Theorem 3 can be shown with a straightforward structural induction on prog once the following lemma is shown. Corollary 1 then follows immediately from Theorem 2 and Theorem 3.

$$\mathit{low\text{-}slice\text{-}of}(\mathsf{send}(ch,e)) = \begin{cases} \mathsf{send}(ch,e) & \text{if } ch \in \mathit{PubCh} \\ \mathsf{skip} & \text{if } ch \in \mathit{PriCh} \cup \mathit{ICh} \\ \mathsf{send}(ch,\diamond) & \text{if } ch \in \mathit{EncCh} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}({}^{as}\mathsf{recv}(ch,x)) = \begin{cases} {}^{as}\mathsf{recv}(ch,x) & \text{if } \mathit{lev}^{\circ}(ch,\mathit{as}) = \mathbb{L} \wedge \mathit{lev}^{\bullet}(ch,\mathit{as}) = \mathbb{L} \\ \mathsf{skip} & \text{if } \mathit{lev}^{\circ}(ch,\mathit{as}) = \mathbb{H} \wedge \mathit{lev}^{\bullet}(ch,\mathit{as}) = \mathbb{H} \\ {}^{as}\mathsf{recv}(ch,\diamond) & \text{otherwise} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}({}^{as}\mathsf{if\text{-}recv}(ch,x,x_{\mathrm{b}})) = \begin{cases} {}^{as}\mathsf{if\text{-}recv}(ch,x,x_{\mathrm{b}}) & \text{if } \mathit{lev}^{\circ}(ch,\mathit{as}) = \mathbb{L} \wedge \mathit{lev}^{\bullet}(ch,\mathit{as}) = \mathbb{L} \\ \mathsf{skip} & \text{if } \mathit{lev}^{\circ}(ch,\mathit{as}) = \mathbb{H} \wedge \mathit{lev}^{\bullet}(ch,\mathit{as}) = \mathbb{H} \\ & \quad \wedge\, \mathit{lev}(x_{\mathrm{b}}) = \mathbb{H} \\ {}^{as}\mathsf{if\text{-}recv}(ch,\diamond,x_{\mathrm{b}}) & \text{otherwise} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}(x := e) = \begin{cases} x := e & \text{if } \mathit{lev}(x) = \mathbb{L} \\ \mathsf{skip} & \text{otherwise} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}(\mathsf{skip}) = \mathsf{skip}$$

$$\mathit{low\text{-}slice\text{-}of}(\mathsf{if}\ e\ \mathsf{then}\ \mathit{prog}_1\ \mathsf{else}\ \mathit{prog}_2\ \mathsf{fi})$$

$$= \begin{cases} \begin{pmatrix} \mathsf{if}\ e\ \mathsf{then} \\ \quad \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) \\ \mathsf{else} \\ \quad \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2) \\ \mathsf{fi} \end{pmatrix} & \text{if } \mathit{lev}\langle e\rangle = \mathbb{L} \\ \mathsf{skip};\mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) & \text{if } \mathit{lev}\langle e\rangle = \mathbb{H}\ \wedge\ \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2) \\ \bot & \text{otherwise} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}(\mathsf{while}\ e\ \mathsf{do}\ \mathit{prog}\ \mathsf{od}) = \begin{cases} \mathsf{while}\ e\ \mathsf{do}\ \mathit{low\text{-}slice\text{-}of}(\mathit{prog})\ \mathsf{od} & \text{if } \mathit{lev}\langle e\rangle = \mathbb{L} \\ \bot & \text{otherwise} \end{cases}$$

$$\mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1;\mathit{prog}_2) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1);\mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2)$$

Fig. 2: The Definition of the Function *low-slice-of*

Lemma 9. *If* $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_1$*,* $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}_2$*, and* $\mathit{low\text{-}slice\text{-}of}(\mathit{prog}_1) = \mathit{low\text{-}slice\text{-}of}(\mathit{prog}_2)$*, then* $\mathit{prog}_1 \sim \mathit{prog}_2$*.*

This lemma can be shown with a structural induction on prog_1, using the two following lemmas in the case for if. The proofs of these two lemmas are straightforward and omitted here.

Lemma 10. *If* $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}$*,* $\mathit{low\text{-}slice\text{-}of}(\mathit{prog}) = \mathsf{skip}$*,* $\sigma \ltimes^{NE} \mathit{asm\text{-}of}(\mathit{prog})$*, then for all mem, there exist* mem' *and* σ'*, such that* $\langle\langle \mathit{prog};\mathit{mem}\rangle;\sigma\rangle \rightarrow \langle\langle \mathsf{stop};\mathit{mem}'\rangle;\sigma'\rangle$*.*

Lemma 11. *If* $\mathit{lev} \vdash_{\mathrm{S}} \mathit{prog}$*,* $\mathit{low\text{-}slice\text{-}of}(\mathit{prog}) = \mathsf{skip}$*, and* $\langle\langle \mathit{prog};\mathit{mem}\rangle;\sigma\rangle \rightarrow \langle\langle \mathit{prog}';\mathit{mem}'\rangle;\sigma'\rangle$*, then the following statements hold:*

1. $\forall x \in \mathit{Var} : \mathit{mem}'(x) \neq \mathit{mem}(x) \Rightarrow \mathit{lev}(x) = \mathbb{H}$*,*
2. $\forall ch \in \mathit{ECh} : \sigma'(ch) \neq \sigma(ch) \Rightarrow ch \in \mathit{PriCh}$*.*

4 Typability of the Auction Example and Authentication Example

Type-Checking Using the Type System from the Paper. As outlined in the paper, the authentication example and the auction example can successfully be type-checked using the type system presented in Fig. 1 of the paper.
While for the authentication example the semantic condition in the premise of the if-rule is not used in the type-checking, the auction example contains two high if-branches where the semantic condition is used. We provide the bisimulation relations that can be used to established the semantic condition for the two if-branches, respectively.
Client Program. With the symmetric closure of the following relation the conditional branching in *auct-cl* can be type-checked.

$$\{(\langle \mathsf{send}(\underline{int}_1, calc(min, thres)); mem_1\rangle, \langle \mathsf{skip}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$
$$\cup \{(\langle \mathsf{stop}; mem_1\rangle, \langle \mathsf{stop}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$

Server Program. With the symmetric closure of the following relation the conditional branching in *auct-srv* can be type-checked.

$$\{(\langle \mathsf{send}(\underline{pri}, bid); mem_1\rangle, \langle \mathsf{skip}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$
$$\cup \{(\langle \mathsf{stop}; mem_1\rangle, \langle \mathsf{stop}; mem_2\rangle) \mid mem_1 =_{\mathbb{L}} mem_2\}$$

Type-Checking Using the Fully-Syntactic Type System of this Addendum. The fully-syntactic type system provided in Sect. 3 of this addendum allows one to type-check both the auction example and the authentication example without constructing bisimulation relations.
For the authentication example, the type-checking can be conducted analogously to the type-checking using the type system from the paper. This is because neither the client program nor the server program contains a high conditional branching. Hence, the semantic condition and also its syntactic replacement are not used in the type-checking.
For the auction example, the type-checking of each high conditional branching can be conducted by a comparison of the low-slices of the two branches. Since for each high conditional branching these two low-slices are equal, both the client program and the server program can be successfully typed. We give the computed low-slices below.
Client Program. For the client program, the relevant low-slices are the following:

$$low\text{-}slice\text{-}of(\mathsf{send}(\underline{int}_1, calc(min, thres))) = \mathsf{skip}$$
$$low\text{-}slice\text{-}of(\mathsf{skip}) = \mathsf{skip}$$

Server Program. For the server program, the relevant low-slices are the following:

$$low\text{-}slice\text{-}of(\mathsf{send}(\underline{pri}, bid)) = \mathsf{skip}$$
$$low\text{-}slice\text{-}of(\mathsf{skip}) = \mathsf{skip}$$

It is worth mentioning that all the example programs from Sect. 2 of our paper [1] are also typable using the fully syntactic security type system.

References

1. Ximeng Li, Heiko Mantel, and Markus Tasch. Taming message-passing communication in compositional reasoning about confidentiality. In *15th Asian Symposium on Programming Languages and Systems (APLAS)*, 2017. Accepted for publication.