Addendum to the Article ”Assumptions and Guarantees for Compositional Noninterference” – Proofs of Theorems and Type System for Establishing Locally Sound Use of Modes

Heiko Mantel[1], David Sands[2], and Henning Sudbrock[1]

[1]Department of Computer Science
TU Darmstadt, Germany
{*mantel,sudbrock*}@*cs.tu-darmstadt.de*

[2]Department of Computer Science and Engineering
Chalmers University of Technology, Sweden
dave@chalmers.se

This document contains proofs for the theorems from the article “Assumptions and Guarantees for Compositional Noninterference” [1] (in Sections I–IX). Moreover, as an addendum to that article it contains a type system for establishing the locally sound use of modes (in Section X).

I. Proof of Proposition 1

Proposition 1. Assume that the multi-threaded program executing the commands $c_1, \ldots, c_n$ is SIFUM-secure. Then whenever $mem_1 =_{low} mem_2$ and

$$\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_1\rangle \rightarrow^* \langle\langle(\mathsf{stop}, mds_0), \ldots, (\mathsf{stop}, mds_0)\rangle, mem'_1\rangle$$

there exists mem'_2 such that $mem'_1 =_{low} mem'_2$ and

$$\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_2\rangle \rightarrow^* \langle\langle(\mathsf{stop}, mds_0), \ldots, (\mathsf{stop}, mds_0)\rangle, mem'_2\rangle.$$

Proof: By the definition of SIFUM-security the number of execution steps of SIFUM-secure programs, and, hence, their termination behavior does not differ when executing the program in two low-equal initial memory states. Hence, we obtain mem'_2 such that

$$\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_2\rangle \rightarrow^* \langle\langle(\mathsf{stop}, mds_0), \ldots, (\mathsf{stop}, mds_0)\rangle, mem'_2\rangle$$

and $mem'_1 =^{mds_0}_{low} mem'_2$ from the definition of SIFUM-security. Since no variable has mode *asm-noread* in mds_0, it follows that $mem'_1 =_{low} mem'_2$. ■

II. Proof of Proposition 2

Proposition 2. Assume that the global configuration $\langle\langle(c_1, mds_1), \ldots, (c_n, mds_n)\rangle, mem\rangle$ ensures a sound use of modes and that

$$\langle\langle(c_1, mds_1), \ldots, (c_n, mds_n)\rangle, mem\rangle \rightarrow^* \langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle.$$

Then $\langle c'_i, mds'_i, mem'\rangle \in lReach(\langle c_i, mds_i, mem\rangle)$ holds for all $i \in \{1, \ldots, n\}$.

Proof: The proof is by induction on the number n of transitions of

$$\langle\langle(c_1, mds_1), \ldots, (c_n, mds_n)\rangle, mem\rangle \rightarrow^* \langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle.$$

In the inductive proof, we prove in addition to the property stated by the theorem (in the proof called “property (1)”) that $\langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle$ ensures a sound use of modes (in the proof called “property (2)”).

1) Assume that $n = 0$. Then $mem = mem'$ and $(c_i, mds_i) = (c'_i, mds'_i)$ for all $i \in \{1, \ldots, n\}$ hold. Hence, $\langle c'_i, mds'_i, mem'\rangle \in lReach(\langle c_i, mds_i, mem\rangle)$ holds for each $i \in \{1, \ldots, n\}$ due to Item (1) in Definition 11, which proves property (1). Property (2) follows as here both global configurations are equal.
2) Assume that $n > 0$. Then there exists $gc'' = \langle\langle(c''_1, mds''_1), \ldots, (c''_n, mds''_n)\rangle, mem''\rangle$ such that
   (a) $\langle\langle(c_1, mds_1), \ldots, (c_n, mds_n)\rangle, mem\rangle \rightarrow^* gc''$ and
   (b) $gc'' \rightarrow \langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle$.
   Due to (a) we may apply the induction hypothesis to gc'', obtaining that
   (c) $\langle c''_i, mds''_i, mem''\rangle \in lReach(\langle c_i, mds_i, mem\rangle)$ for each $i \in \{1, \ldots, n\}$, and that
   (d) gc'' ensures a sound use of modes.
   From (b) and (d) it follows that
   (e) $\langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle$ ensures a globally sound use of modes.
   From (b) and the definition of the relation $\rightarrow$, there exists $k \in \{1, \ldots, n\}$ such that
   (f) $\langle c''_k, mds''_k, mem''\rangle \rightarrowtriangle \langle c'_k, mds'_k, mem'\rangle$ and
   (g) $(c''_i, mds''_i) = (c'_i, mds'_i)$ for all $i \in \{1, \ldots, n\} \setminus \{k\}$.
   From (c), (f), and Item (2) in Definition 11 we obtain
   (h) $\langle c'_k, mds'_k, mem'\rangle \in lReach(\langle c_k, mds_k, mem\rangle)$.
   Let $i \neq k$ and $x \in asm\text{-}nowrite(mds''_i)$. As by (e) $\langle\langle(c'_1, mds'_1), \ldots, (c'_n, mds'_n)\rangle, mem'\rangle$ ensures a globally sound use of modes, we obtain from (a) that
   (i) $guar\text{-}nowrite \in mds''_k(x)$.
   As $\langle c_k, mds_k, mem\rangle$ ensures a locally sound use of

modes, we obtain from (c) and (i) that
(j) c''_k does not modify x.
Hence, we have $mem'(x) = mem''(x)$ for all $x \in mds''_i(asm\text{-}nowrite)$ Thus, by Item (3) of Definition 11, we conclude that $\langle c'_i, mds'_i, mem'\rangle \in lReach(\langle c_i, mds_i, mem\rangle)$ for $i \neq k$, and, hence, for all $i \in \{1, \ldots, n\}$ due to (h).
Thus, property (1) is satisfied.
From property (1), we obtain that $lReach(\langle c'_i, mds'_i, mem'\rangle) \subseteq lReach(\langle c_i, mds_i, mem\rangle)$. Hence, $\langle c'_i, mds'_i, mem'\rangle$ ensures a locally sound use of modes as $\langle c_i, mds_i, mem\rangle$ ensures a locally sound use of modes. This fact together with (e) results in property (2), which concludes the proof. ■

III. Proof of Theorem 1

Theorem 1 (Compositionality). Let $c_1, \ldots, c_n$ be SIFUM-secure commands such that $\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem\rangle$ ensures a sound use of modes for every memory state mem. Then the multi-threaded program executing the commands $c_1, \ldots, c_n$ is SIFUM-secure.

For proving Theorem 1, we first establish two lemmata.

Lemma 1. Let c, c' be commands, mds, mds' be mode states, mem_1, mem'_1, mem_2 be memory states, and $x_1, \ldots, x_k$ be variables. Assume that $\langle c, mds, mem_1\rangle \rightarrowtriangle \langle c', mds', mem'_1\rangle$, that c does not read x_i for all $i \in \{1, \ldots, k\}$, and that $mem_1(x) = mem_2(x)$ for all $x \in Var \setminus \{x_1, \ldots, x_k\}$.

Then there exists a memory state mem'_2 such that $\langle c, mds, mem_2\rangle \rightarrowtriangle \langle c', mds', mem'_2\rangle$ and such that $mem'_1(x) = mem'_2(x)$ for all $x \notin \{x_1, \ldots, x_k\}$.

Moreover, if $mem_1(x) \neq mem'_1(x)$ or $mem_2(x) \neq mem'_2(x)$ for $x \in \{x_1, \ldots, x_k\}$, then $mem'_1(x) = mem'_2(x)$.

Proof: The proof is by induction on k.

Let firstly $k = 0$. In this case, $mem_1 = mem_2$, and we conclude by setting $mem'_2 = mem'_1$.

Now let $k > 0$. Let $mem_3 = mem_2[x_k \mapsto mem_1(x_k)]$. Then mem_1 and mem_3 differ only on the variables $x_1, \ldots, x_{k-1}$. Hence, by the induction hypothesis there exists mem'_3 such that $\langle c, mds, mem_3\rangle \rightarrowtriangle \langle c', mds', mem'_3\rangle$ and such that mem'_1 and mem'_3 differ only on the variables $x \in \{x_1, \ldots, x_{k-1}\}$ for which $mem_1(x) = mem'_1(x)$ or $mem_3(x) = mem'_3(x)$. By construction of mem_3, there exists $v \in Val$ such that $mem_2 = mem_3[x_k \mapsto v]$. As c does not read x_k, due to $\langle c, mds, mem_3\rangle \rightarrowtriangle \langle c', mds', mem'_3\rangle$ one of the following two conditions holds:

- $\langle c, mds, mem_2\rangle \rightarrowtriangle \langle c', mds', mem'_3[x_k \mapsto v]\rangle$
- $\langle c, mds, mem_2\rangle \rightarrowtriangle \langle c', mds', mem'_3\rangle$

In the first case define $mem'_2 = mem'_3[x_k \mapsto v]$, and in the second case define $mem'_2 = mem'_3$. Then mem'_2 and mem'_3 differ at most on x_k. Moreover, if $mem_2(x_k) \neq mem'_2(x_k)$, then $mem'_2(x_k) \neq v$, and hence we must have the second case. But then $mem'_2(x_k) = mem'_3(x_k)$. Finally, if $mem_3(x_k) \neq mem'_3(x_k)$, we can turn the reasoning around (considering $mem_3 = mem_2[x_k \mapsto v']$) to conclude.

Hence, mem'_1 and mem'_2 differ only on $x_1, \ldots, x_k$, and they do not differ on those $x \in \{x_1, \ldots, x_k\}$ that have been modified in one of the two execution steps. ■

Lemma 2. Let $\langle\langle(c_{1,1}, mds_1), \ldots, (c_{1,n}, mds_n)\rangle, mem_1\rangle$ and $\langle\langle(c_{2,1}, mds_1), \ldots, (c_{2,n}, mds_n)\rangle, mem_2\rangle$ be global configurations that ensure a sound use of modes. Whenever $\langle\langle(c_{1,1}, mds_1), \ldots, (c_{1,n}, mds_n)\rangle, mem_1\rangle \rightarrow \langle\langle(c'_{1,1}, mds'_1), \ldots, (c'_{1,n}, mds'_n)\rangle, mem'_1\rangle$ and there exist $mem_{1,i}$ and $mem_{2,i}$ for all $i \in \{1, \ldots, n\}$ with

- $\langle c_{1,i}, mds_i, mem_{1,i}\rangle \approx \langle c_{2,i}, mds_i, mem_{2,i}\rangle$ and
- $mem_{1,i}(x) = mem_1(x)$ and $mem_{2,i}(x) = mem_2(x)$ whenever $\big[(\mathcal{L}(x) = high) \vee (mem_1(x) = mem_2(x)) \vee (\forall j \in \{1, \ldots, n\} : x \notin mds_j(asm\text{-}noread))\big]$ holds,

then there exist $c'_{2,1} \ldots, c'_{2,n}$ and mem'_2 such that

(1) $\langle\langle(c_{2,1}, mds_1), \ldots, (c_{2,n}, mds_n)\rangle, mem_2\rangle \rightarrow \langle\langle(c'_{2,1}, mds'_1), \ldots, (c'_{2,n}, mds'_n)\rangle, mem'_2\rangle$ and
(2) for all $i \in \{1, \ldots, n\}$ there are $mem'_{1,i}$ and $mem'_{2,i}$ with
- $\langle c'_{1,i}, mds'_i, mem'_{1,i}\rangle \approx \langle c'_{2,i}, mds'_i, mem'_{2,i}\rangle$ and
- $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$ whenever $\big[(\mathcal{L}(x) = high) \vee (mem'_1(x) = mem'_2(x)) \vee (\forall j \in \{1, \ldots, n\} : x \notin mds'_j(asm\text{-}noread))\big]$ holds.

Proof: In the first step, we construct a global configuration $\langle\langle(c'_{2,1}, mds'_1), \ldots, (c'_{2,n}, mds'_n)\rangle, mem'_2\rangle$ such that conclusion (1) stated by the lemma is satisfied, i.e.,

$$\langle\langle(c_{2,1}, mds_1), \ldots, (c_{2,n}, mds_n)\rangle, mem_2\rangle \rightarrow \langle\langle(c'_{2,1}, mds'_1), \ldots, (c'_{2,n}, mds'_n)\rangle, mem'_2\rangle.$$

In the second step, we prove that conclusion (2) stated by the lemma is satisfied for the global configuration constructed in the first step.

Step 1. We show that the execution step in the global configuration $\langle\langle(c_{1,1}, mds_1), \ldots, (c_{1,n}, mds_n)\rangle, mem_1\rangle$ can be matched by an execution step in the global configuration $\langle\langle(c_{2,1}, mds_1), \ldots, (c_{2,n}, mds_n)\rangle, mem_2\rangle$.

By the assumption of the theorem,

$$\langle\langle(c_{1,1}, mds_1), \ldots, (c_{1,n}, mds_n)\rangle, mem_1\rangle \rightarrow \langle\langle(c'_{1,1}, mds'_1), \ldots, (c'_{1,n}, mds'_n)\rangle, mem'_1\rangle.$$

Hence, by the definition of the transition relation $\rightarrow$ there exists some $j \in \{1, \ldots, n\}$ such that

(a) $\langle c_{1,j}, mds_j, mem_1\rangle \rightarrowtriangle \langle c'_{1,j}, mds'_j, mem'_1\rangle$ and
(b) $(c_{1,i}, mds_i) = (c'_{1,i}, mds'_i)$ for all $i \neq j$.

By assumption (2) of the lemma there hence exist memory states $mem_{1,j}$ and $mem_{2,j}$ such that for all $x \in Var$ we have

(c) $[\mathcal{L}(x) = high \vee mem_1(x) = mem_2(x) \vee \forall k \neq j : x \notin mds_k(asm\text{-}noread)] \implies mem_{1,j}(x) = mem_1(x)$

and

(d) $[\mathcal{L}(x) = \mathit{high} \vee \mathit{mem}_1(x) = \mathit{mem}_2(x) \vee \forall k \neq j : x \notin \mathit{mds}_k(\mathit{asm\text{-}noread})] \implies \mathit{mem}_{2,j}(x) = \mathit{mem}_2(x)$,

and moreover

(e) $\langle c_{1,j}, \mathit{mds}_j, \mathit{mem}_{1,j}\rangle \approx \langle c_{2,j}, \mathit{mds}_j, \mathit{mem}_{2,j}\rangle$.

From (e), we directly obtain

(f) $\mathit{mem}_{1,j} =_{low}^{\mathit{mds}_j} \mathit{mem}_{2,j}$

by the definition of strong low bisimulation modulo modes.

Due to (c), mem_1 and $\mathit{mem}_{1,j}$ differ only in variables x with $\mathcal{L}(x) = \mathit{low}$, $\mathit{mem}_1(x) \neq \mathit{mem}_2(x)$, and $x \in \mathit{mds}_k(\mathit{asm\text{-}noread})$ for some $k \neq j$. As by assumption $\langle\langle(c_{1,1}, \mathit{mds}_1), \ldots, (c_{1,n}, \mathit{mds}_n)\rangle, \mathit{mem}_1\rangle$ ensures a globally sound use of modes, $x \in \mathit{mds}_j(\mathit{guar\text{-}noread})$ holds for these variables. Moreover, as by assumption $\langle c_{1,j}, \mathit{mds}_j, \mathit{mem}_1\rangle$ ensures a locally sound use of modes, $c_{1,j}$ does not read the variables with mode $\mathit{guar\text{-}noread}$, i.e., in particular $c_{1,j}$ does not read the variables whose values differ in $\mathit{mem}_{1,j}$ and mem_1. We may hence apply Lemma 1 for the transition in (a) and the memory state $\mathit{mem}_{1,j}$ This yields a memory state $\mathit{mem}'_{1,j}$ with

(g) $\langle c_{1,j}, \mathit{mds}_j, \mathit{mem}_{1,j}\rangle \rightharpoonup \langle c'_{1,j}, \mathit{mds}'_j, \mathit{mem}'_{1,j}\rangle$ and

(h) for all $x \in \mathit{Var}$, $[\mathcal{L}(x) = \mathit{high} \vee \mathit{mem}_1(x) = \mathit{mem}_2(x) \vee \forall k \neq j : x \notin \mathit{mds}_k(\mathit{asm\text{-}noread})] \implies \mathit{mem}'_{1,j}(x) = \mathit{mem}'_1(x)$.

As $\langle c_{1,j}, \mathit{mds}_j, \mathit{mem}_{1,j}\rangle$ and $\langle c_{2,j}, \mathit{mds}_j, \mathit{mem}_{2,j}\rangle$ are strong low bisimilar modulo modes (compare (e)), we obtain due to (g) that there exist $c'_{2,j}$ and $\mathit{mem}'_{2,j}$ with

(i) $\langle c_{2,j}, \mathit{mds}_j, \mathit{mem}_{2,j}\rangle \rightharpoonup \langle c'_{2,j}, \mathit{mds}'_j, \mathit{mem}'_{2,j}\rangle$ and

(j) $\langle c'_{1,j}, \mathit{mds}'_j, \mathit{mem}'_{1,j}\rangle \approx \langle c'_{2,j}, \mathit{mds}'_j, \mathit{mem}'_{2,j}\rangle$.

From (j) and the definition of strong low bisimulation modulo modes we directly obtain

(k) $\mathit{mem}'_{1,j} =_{low}^{\mathit{mds}'_j} \mathit{mem}'_{2,j}$.

Due to (d), we may exploit global and local soundness as before applying Lemma 1 earlier in this prove, and apply Lemma 1 for the transition in (i) and the memory state mem_2. This yields a memory state mem'_2 such that

(l) $\langle c_{2,j}, \mathit{mds}_j, \mathit{mem}_2\rangle \rightharpoonup \langle c'_{2,j}, \mathit{mds}'_j, \mathit{mem}'_2\rangle$ and

(m) for all $x \in \mathit{Var}$, $[\mathcal{L}(x) = \mathit{high} \vee \mathit{mem}_1(x) = \mathit{mem}_2(x) \vee \forall k \neq j : x \notin \mathit{mds}_k(\mathit{asm\text{-}noread})] \implies \mathit{mem}'_2(x) = \mathit{mem}'_{2,j}(x)$.

I.e., we now have constructed both $c'_{2,j}$ and mem'_2. It remains to construct $c'_{2,i}$ for $i \neq j$. To this end, we define

(n) $c'_{2,i} := c_{2,i}$ for all $i \neq j$.

Then, due to (l), we have

$$\langle\langle(c_{2,1}, \mathit{mds}_1), \ldots, (c_{2,n}, \mathit{mds}_n)\rangle, \mathit{mem}_2\rangle \to \langle\langle(c'_{2,1}, \mathit{mds}'_1), \ldots, (c'_{2,n}, \mathit{mds}'_n)\rangle, \mathit{mem}'_2\rangle$$

by the definition of the semantics for global transitions.

Step 2. In this step, we show that for all $i \in \{1, \ldots, n\}$ there exist memory states $\mathit{mem}'_{1,i}$ and $\mathit{mem}'_{2,i}$ with $\mathit{mem}'_{1,i}(x) = \mathit{mem}'_1(x)$ and $\mathit{mem}'_{2,i}(x) = \mathit{mem}'_2(x)$ for all $x \in \mathit{Var}$ that do not satisfy $\mathcal{L}(x) = \mathit{low}$, $\mathit{mem}'_1(x) \neq \mathit{mem}'_2(x)$, and $x \in \mathit{mds}'_k(\mathit{asm\text{-}noread})$ for some $k \neq i$, and with $\langle c'_{1,i}, \mathit{mds}'_i, \mathit{mem}'_{1,i}\rangle \approx \langle c'_{2,i}, \mathit{mds}'_i, \mathit{mem}'_{2,i}\rangle$.

We distinguish the two cases $i = j$ and $i \neq j$ (where j is the index of the command performing the execution step, as exhibited in Step 1).

Case 1: $i = j$: The memory states $\mathit{mem}'_{1,j}$ and $\mathit{mem}'_{2,j}$ have already been constructed in Step 1. That $\langle c'_{1,j}, \mathit{mds}'_j, \mathit{mem}'_{1,j}\rangle \approx \langle c'_{2,j}, \mathit{mds}'_j, \mathit{mem}'_{2,j}\rangle$ has been established in (j). It remains to show that whenever $\mathcal{L}(x) = \mathit{high}$, $\mathit{mem}'_1(x) = \mathit{mem}'_2(x)$, or $\forall k \neq j : x \notin \mathit{mds}'_k(\mathit{asm\text{-}noread})$ holds, both $\mathit{mem}'_{1,j}(x) = \mathit{mem}'_1(x)$ and $\mathit{mem}'_{2,j}(x) = \mathit{mem}'_2(x)$ hold.

Assume firstly that $\mathcal{L}(x) = \mathit{high}$. But then the required equalities follow directly from (h) and (m).

Assume now that $\forall k \neq j : x \notin \mathit{mds}'_k(\mathit{asm\text{-}noread})$. As for $k \neq j$ we have $\mathit{mds}'_k = \mathit{mds}_k$ (compare (b)), this is equivalent to $\forall k \neq j : x \notin \mathit{mds}_k(\mathit{asm\text{-}noread})$. But then again, the required equalities follow directly from (h) and (m).

Assume finally that $\mathit{mem}'_1(x) = \mathit{mem}'_2(x)$. If $\mathit{mem}_1(x) = \mathit{mem}_2(x)$ would also hold, then the required equalities would again directly follow from (h) and (m). Hence, we assume that $\mathit{mem}_1(x) \neq \mathit{mem}_2(x)$. But then the execution step from (a) or the execution step from (g) modifies x. As both execution steps have been obtained with Lemma 1, we obtain that x is modified to the same value in $\mathit{mem}'_{1,j}$ and mem'_1 respectively $\mathit{mem}'_{2,j}$ and mem'_2, and we can conclude.

Case 2: $i \neq j$: We define the memory states $\mathit{mem}'_{1,i}$ and $\mathit{mem}'_{2,i}$ as follows for all $x \in \mathit{Var}$:

(o) If $\mathcal{L}(x) = \mathit{high}$, or if $\mathit{mem}'_1(x) = \mathit{mem}'_2(x)$, or if $x \notin \mathit{mds}'_k(x)(\mathit{asm\text{-}noread})$ for all $k \neq i$, then $\mathit{mem}'_{1,i}(x) = \mathit{mem}'_1(x)$ and $\mathit{mem}'_{2,i}(x) = \mathit{mem}'_2(x)$.

(p) If $\mathcal{L}(x) = \mathit{low}$, $\mathit{mem}'_1(x) \neq \mathit{mem}'_2(x)$, and there exists $k \neq i$ such that $x \in \mathit{mds}'_k(x)(\mathit{asm\text{-}noread})$, then $\mathit{mem}'_{1,i}(x) = \mathit{mem}_{1,i}(x)$ and $\mathit{mem}'_{2,i}(x) = \mathit{mem}_{2,i}(x)$.

We firstly show that $\mathit{mem}'_{1,i}(x) = \mathit{mem}'_1(x)$ and $\mathit{mem}'_{2,i}(x) = \mathit{mem}'_2(x)$ for all $x \in \mathit{Var}$ satisfying $\mathcal{L}(x) = \mathit{high}$, $\mathit{mem}'_1(x) = \mathit{mem}'_2(x)$, or $\forall k \neq i : x \notin \mathit{mds}'_k(\mathit{asm\text{-}noread})$. Let x be a variable with these properties. Then case (o) applies and we obtain directly that $\mathit{mem}'_{1,i}(x) = \mathit{mem}'_1(x)$ and $\mathit{mem}'_{2,i}(x) = \mathit{mem}'_2(x)$.

We now show that $\langle c'_{1,i}, \mathit{mds}'_i, \mathit{mem}'_{1,i}\rangle \approx \langle c'_{2,i}, \mathit{mds}'_i, \mathit{mem}'_{2,i}\rangle$. As $j \neq i$, we have $c_{1,i} = c'_{1,i}$, $c_{2,i} = c'_{2,i}$, and $\mathit{mds}_i = \mathit{mds}'_i$. Hence, we need to show that $\langle c_{1,i}, \mathit{mds}_i, \mathit{mem}'_{1,i}\rangle \approx \langle c_{2,i}, \mathit{mds}_i, \mathit{mem}'_{2,i}\rangle$. By assumption (2) of the lemma, we know that $\langle c_{1,i}, \mathit{mds}_i, \mathit{mem}_{1,i}\rangle \approx \langle c_{2,i}, \mathit{mds}_i, \mathit{mem}_{2,i}\rangle$. As $\approx$ is closed under globally consistent changes, we will conclude by showing that $\mathit{mem}'_{1,i}$ and $\mathit{mem}'_{2,i}$ can be obtained from $\mathit{mem}_{1,i}$ and $\mathit{mem}_{2,i}$ using the closure conditions from

Definition 3.

Firstly note that, due to the definition in (o) and (p), $mem'_{1,i}(x) \neq mem_{1,i}(x)$ or $mem'_{2,i}(x) \neq mem_{2,i}(x)$ only if $\mathcal{L}(x) = high$, $mem'_1(x) = mem'_2(x)$, or $x \notin mds'_k(asm\text{-}noread)$ for all $k \neq i$. Hence, we only need to consider variables for which one of these three conditions holds in the following. Note furthermore that then, due to (o), $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$) hold.

Consider firstly a variable x such that $x \notin mds_i(asm\text{-}nowrite)$. If $\mathcal{L}(x) = high$, then evidently the new values of x can be set by a globally consistent change, as globally consistent changes allow arbitrary new values for high variables. If $\mathcal{L}(x) = low$, then by our assumptions in the previous paragraph either $mem'_1(x) = mem'_2(x)$, or $x \notin mds'_k(asm\text{-}noread)$ for all $k \neq i$. But then $x \notin mds'_j(asm\text{-}noread)$, and hence, due to (k), $mem'_{1,j}(x) = mem'_{2,j}(x)$. But then, by (h) and (m), also $mem'_1(x) = mem'_2(x)$. We can now conclude as globally consistent changes require equal values for low variables.

Consider now a variable x with $x \in mds_i(asm\text{-}nowrite)$. In this case, $x \in mds_j(guar\text{-}nowrite)$ due to global soundness. But then, due to local soundness, $c_{1,j}$ and $c_{2,j}$ do not modify x. Hence, due to the definition of "does not modify", $mem_1(x) = mem'_1(x)$ and $mem_2(x) = mem'_2(x)$.

Assume firstly that $mem_1(x) = mem_{1,i}(x)$ and $mem_2(x) = mem_{2,i}(x)$. Then $mem'_{1,i}(x) = mem_{1,i}(x)$ as well as $mem'_{2,i}(x) = mem_{2,i}(x)$ due to our assumptions made above, i.e., the value of x remains unchanged.

Assume now contrarily that $mem_1(x) \neq mem_{1,i}(x)$ or that $mem_2(x) \neq mem_{2,i}(x)$. Then, by assumption (2) of the lemma, $\mathcal{L}(x) = low$, $mem_1(x) \neq mem_2(x)$, and $x \in mds_k(asm\text{-}noread)$ for some $k \neq i$. If in this case $mem'_1(x) = mem'_2(x)$ would hold, this would contradict that the value of x remains unchanged by $c_{1,j}$ and $c_{2,j}$. Hence, let us assume that $mem'_1(x) \neq mem'_2(x)$. But then, due to our assumptions made above, it must hold that $x \notin mds'_k(asm\text{-}noread)$ for all $k \neq i$. But as $x \in mds_k(asm\text{-}noread)$ for some $k \neq i$, and $mds_k \neq mds'_k$ only holds for $k = j$, we obtain that $x \in mds_j(asm\text{-}noread)$ and $x \notin mds'_j(asm\text{-}noread)$. But this contradicts the fact that $mem_{1,j} =^{mds_j}_{low} mem_{2,j}$ and $mem'_{1,j} =^{mds'_j}_{low} mem'_{2,j}$, while both $c_{1,j}$ and $c_{2,j}$ leave the value of x unchanged.

Hence, we know that $mem_{1,i}(x) = mem'_1(x)$ and $mem_{2,i}(x) = mem'_2(x)$ for all variables with $x \in mds'_i(asm\text{-}nowrite)$, and that we hence must not apply globally consistent changes to these variables. ■

Now we prove Theorem 1:

Proof: We want to apply Lemma 2 for the commands $c_1, \ldots, c_n$ and the mode state mds_0. By the definition of SIFUM-security for commands (Definition 5), we have $\langle c_i, mds_0, mem_1\rangle \approx \langle c_i, mds_0, mem_2\rangle$ for all $i \in \{1, \ldots, n\}$. Hence, when setting $mem_{1,i} = mem_1$ and $mem_{2,i} = mem_2$, all the preconditions of Lemma 2 are satisfied, and we obtain that each execution step in $\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_1\rangle$ can be matched by an execution step in $\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_2\rangle$. Moreover, the preconditions of Lemma 2 are again satisfied after this first execution step. This is on the one hand due to the fact that the assertions of Lemma 2 for the global configurations after an execution step correspond to its preconditions when applying the theorem to the global configurations after the execution step, and on the other hand this is due to the fact that both locally and globally sound use of capabilities are preserved after an execution step (due to Proposition 2).

We can thus inductively apply Lemma 2 k times. In doing so, we obtain $c''_1, \ldots, c''_n$ and mem'_2 such that

$$\langle\langle(c_1, mds_0), \ldots, (c_n, mds_0)\rangle, mem_2\rangle \to^k \langle\langle(c''_1, mds_1), \ldots, (c''_n, mds_n)\rangle, mem'_2\rangle,$$

and such that there exist (for all i) $mem'_{1,i}$ and $mem'_{2,i}$ with the properties in condition (2) of Lemma 2. It remains to show that $mem'_1(x) = mem'_2(x)$ whenever $\mathcal{L}(x) = low$ and $x \notin mds_i(asm\text{-}noread)$ for all $i \in \{1, \ldots, n\}$. But for such x, we have $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$, and, hence, $mem'_1(x) = mem'_2(x)$ as $\langle c'_1, mds_i, mem'_{1,i}\rangle \approx \langle c''_1, mds_i, mem'_{2,i}\rangle$. ■

IV. Proof of Theorem 2

Theorem 2. Assume that $\mathcal{L}(\mathsf{log}) = \mathcal{L}(\mathsf{debug}) = low$ and $\mathcal{L}(\mathsf{secret}) = high$. Then c_{debug} is SIFUM-secure.

Proof: We iteratively construct a strong low bisimulation modulo modes such that $\langle c_{debug}, mds_0, mem_1\rangle \ \mathcal{R} \ \langle c_{debug}, mds_0, mem_2\rangle$ for all $mem_1 =_{low} mem_2$. Due to the definition of strong low bisimulations modulo modes, the configurations containing the program obtained from c_{debug} by removing the annotated assignment to debug and the mode state where debug has mode *asm-nowrite* must be related to themselves by $\mathcal{R}$ for all low-equal memory states that assign False to debug (note that globally consistent changes will not modify debug due to its mode). As the value of debug is False, the guard of the if-then-else statement evaluates to False, and, hence, the definition of strong low bisimulation modulo modes requires that configurations containing the command stop and the mode state mds_0 must be related to themselves. As stop cannot perform any execution step, we do not have to add any more pairs to the relation $\mathcal{R}$ – we added enough pairs to make it a strong low bisimulation modulo modes. Hence, the program c_{debug} is SIFUM-secure. ■

V. Proof of Theorem 3

Theorem 3. Assume that $\mathcal{L}(\mathsf{key1}) = \mathcal{L}(\mathsf{key2}) = \mathit{high}$ and $\mathcal{L}(\mathsf{pub1}) = \mathcal{L}(\mathsf{pub2}) = \mathcal{L}(\mathsf{temp}) = \mathit{low}$. Then c_{temp} is SIFUM-secure.

Proof: We could prove Theorem 3 as in the proof of Theorem 2 by iteratively constructing a strong low bisimulation modulo modes that relates configurations containing the command c_{temp}, the mode state mds_0, and low-equal memory states. However, it also suffices to show that c_{temp} is typable, as then the SIFUM-security of c_{temp} follows from Theorem 5. A typing of c_{temp} can be established by applying the typing rule [anno] twice, the rule [assign$_2$] twice, and the rule [assign$_1$] four times. ■

VI. Proof of Theorem 4

Theorem 4. Assume that $\mathcal{L}(\mathsf{secretData}) = \mathcal{L}(\mathsf{localout}) = \mathit{high}$ and $\mathcal{L}(x) = \mathit{low}$ for all remaining variables. Then c_{srv} is SIFUM-secure.

Proof: This theorem can be proved (as in the proof of Theorem 2) by iteratively constructing a strong low bisimulation modulo modes that relates configurations containing the command c_{srv}, the mode state mds_0, and low-equal memory states. In the construction, the modes *asm-nowrite* are exploited to ensure that the values of variables are not modified by globally consistent changes, and the modes *asm-noread* are exploited as they allow that low variables store secrets (as the variable answer). ■

VII. Proof of Theorem 5

Theorem 5. Assume that $\vdash \Gamma\ \{c\}\ \Gamma'$ is derivable, and let mds be a mode state that is consistent with Γ. Then $\langle c, mds, mem_1\rangle \approx \langle c, mds, mem_2\rangle$ holds for all $mem_1, mem_2 \in Mem$ that satisfy $mem_1(x) = mem_2(x)$ for all $x \in Var$ with $\Gamma\langle x\rangle = low$.

Proof:

Outline: We firstly construct a family of relations $\mathcal{R}^{\Gamma}$ parametrized by type environments such that $\langle c, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c, mds, mem_2\rangle$. We then prove that $\mathcal{R}^{\Gamma}$ is a strong low bisimulation modulo modes for each type environment Γ. This establishes that $\langle c, mds, mem_1\rangle \approx \langle c, mds, mem_2\rangle$, as $\approx$ is the union of all strong low bisimulations modulo modes.

Step 1: Construction of the family of relations $\mathcal{R}^{\Gamma}$. For defining the family of relations, we introduce an auxiliary typing rule for the command stop. This allows us to omit special cases stating that a command is either typable, or equals stop. As adding this typing rule enlarges the set of typable commands, we actually prove a stronger result as the theorem, as we also include the stop-command in its statement. The new rule is as follows:

$$[\mathsf{stop}]\,\frac{}{\vdash \Gamma\ \{\mathsf{stop}\}\ \Gamma}$$

We furthermore write $mem_1 =_{\Gamma} mem_2$ if $mem_1(x) = mem_2(x)$ for all $x \in Var$ with $\Gamma\langle x\rangle = low$. Finally, we denote the set of mode states that are compatible with Γ with $comp(\Gamma)$.

We define $\mathcal{R}^{\Gamma'} := \mathcal{R}_1^{\Gamma'} \cup \mathcal{R}_2^{\Gamma'} \cup \mathcal{R}_3^{\Gamma'}$, where

$$\mathcal{R}_1^{\Gamma'} = \{(\langle c, mds, mem_1\rangle\ ,\ \langle c, mds, mem_2\rangle) \mid \exists\Gamma :\ \vdash \Gamma\ \{c\}\ \Gamma' \wedge mds \in comp(\Gamma) \wedge mem_1 =_{\Gamma} mem_2\},$$

$$\begin{aligned}\mathcal{R}_2^{\Gamma'} = \{(\langle c_1, mds, mem_1\rangle\ ,\ \langle c_2, mds, mem_2\rangle) \mid\ & \langle c_1, mds, mem_1\rangle \approx \langle c_2, mds, mem_2\rangle \wedge \\ & \forall x \in dom(\Gamma') : \Gamma'(x) = high \wedge \\ & \exists \Gamma_1, \Gamma_2 : \vdash \Gamma_1\ \{c_1\}\ \Gamma' \wedge \vdash \Gamma_2\ \{c_2\}\ \Gamma' \wedge \\ & mds \in comp(\Gamma_1) \wedge mds \in comp(\Gamma_2)\}, \text{and}\end{aligned}$$

$$\mathcal{R}_3^{\Gamma'} = \{(\langle c_1; c, mds, mem_1\rangle\ ,\ \langle c_2; c, mds, mem_2\rangle) \mid \exists\Gamma :\ \langle c_1, mds, mem_1\rangle \mathcal{R}_2^{\Gamma} \langle c_2, mds, mem_2\rangle \wedge \vdash \Gamma\ \{c\}\ \Gamma'\}.$$

Step 2: Show that $\langle c, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c, mds, mem_2\rangle$ holds. By the assumptions of the theorem, we evidently have $\langle c, mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c, mds, mem_2\rangle$. Hence, $\langle c, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c, mds, mem_2\rangle$.

Step 3: Show that $\mathcal{R}^{\Gamma'}$ is a strong low bisimulation modulo modes for each type environment Γ'.

It is clear from its definition that $\mathcal{R}^{\Gamma'}$ is a symmetric relation.

Now we show that $\mathcal{R}^{\Gamma'}$ is closed under globally consistent changes. Assume hence that $\langle c_1, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c_2, mds, mem_2\rangle$. We must show that for all $x \in Var$ with $x \notin mds(\mathit{asm\text{-}nowrite})$ the following hold:

(1) If $\mathcal{L}(x) = low$, then $\langle c_1, mds, mem_1[x \mapsto v]\rangle \mathcal{R}^{\Gamma'} \langle c_2, mds, mem_2[x \mapsto v]\rangle$ for all $v \in Val$.

(2) If $\mathcal{L}(x) = high$, then $\langle c_1, mds, mem_1[x \mapsto v_1]\rangle \mathcal{R}^{\Gamma'} \langle c_2, mds, mem_2[x \mapsto v_1]\rangle$ for all $v_1, v_2 \in Val$.

We distinguish between the three cases arising from the definition of $\mathcal{R}^{\Gamma'}$.

- Case 1: $\langle c_1, mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c_2, mds, mem_2\rangle$: Let Γ be a type environment with the properties stated in the definition of $\mathcal{R}_1^{\Gamma'}$. We need to show that the memory states are still related by $=_{\Gamma}$ after the modification of x. In (1), this is evident as x is set to an equal value on both sides of the relation. In (2), we know that $\mathcal{L}(x) = high$. Hence, $\Gamma\langle x\rangle = low$ only if $x \in dom(\Gamma)$. But then, as $mds \in comp(\Gamma)$, we have $x \in mds(\mathit{asm\text{-}nowrite})$. But this would contradict our assumption that $x \notin mds(\mathit{asm\text{-}nowrite})$, and hence $\Gamma\langle x\rangle = \mathit{high}$. Hence, $\mathit{mem}_1[x \mapsto v_1] =_\Gamma \mathit{mem}_2[x \mapsto v_2]$ holds also in (2).

- Case 2: $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}_2^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$: By the definition of $\mathcal{R}_2^{\Gamma'}$, the two local configurations are related by $\approx$. As $\approx$ is a strong low bisimulation modulo modes, it is closed under globally consistent changes.
- Case 3: $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}_3^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$: As the two local configurations are related by $\mathcal{R}_2^{\Gamma'}$, we ca conclude as in Case 2.

Now we show that whenever $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$, then $\mathit{mem}_1 =^{\mathit{mds}}_{\mathit{low}} \mathit{mem}_2$. We distinguish three cases, namely that the two local configurations are related via $\mathcal{R}_1^{\Gamma'}$, via $\mathcal{R}_2^{\Gamma'}$, or via $\mathcal{R}_3^{\Gamma'}$. In the last two cases, by the definition of the relations we know that the local configurations are related by the strong low bisimulation modulo modes $\approx$, and, hence, the statement follows directly from the definition of strong low bisimulations modulo modes.

Assume hence that $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}_1^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$. Let Γ be a type environment with the properties stated in the definition of $\mathcal{R}_1^{\Gamma'}$. To show $\mathit{mem}_1 =^{\mathit{mds}}_{\mathit{low}} \mathit{mem}_2$, let $x \in \mathit{Var}$ with $\mathcal{L}(x) = \mathit{low}$ and $x \notin \mathit{mds}(\mathit{asm\text{-}noread})$. But then, as $\mathit{mds} \in \mathit{comp}(\Gamma)$ by the definition of $\mathcal{R}_1^{\Gamma'}$, we have $x \notin \mathit{dom}(\Gamma)$. Hence, $\Gamma\langle x\rangle = \mathit{low}$. Then, $\mathit{mds}_1(x) = \mathit{mds}_2(x)$ follows directly from $\mathit{mds}_1 =_\Gamma \mathit{mds}_2$.

We finally show that whenever $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$ and $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \twoheadrightarrow \langle c_1', \mathit{mds}', \mathit{mem}_1'\rangle$, then there exist c_2' and mem_2' such that both $\langle c_2, \mathit{mds}, \mathit{mem}_2\rangle \twoheadrightarrow \langle c_2', \mathit{mds}', \mathit{mem}_2'\rangle$ and $\langle c_1', \mathit{mds}', \mathit{mem}_1'\rangle \mathcal{R}^{\Gamma'} \langle c_2', \mathit{mds}', \mathit{mem}_2'\rangle$ hold.

For showing this, we distinguish the cases that $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle$ and $\langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$ are related by $\mathcal{R}_1^{\Gamma'}$, $\mathcal{R}_2^{\Gamma'}$, respectively $\mathcal{R}_3^{\Gamma'}$. We treat these three cases in the remainder of the proof.

Case 1: $\langle c_1, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}_1^{\Gamma'} \langle c_2, \mathit{mds}, \mathit{mem}_2\rangle$. In this case, $c_1 = c_2$ (we will write c for both commands in the following), and there exists a type environment Γ such that $\vdash \Gamma\ \{c\}\ \Gamma'$, $\mathit{mds} \in \mathit{comp}(\Gamma)$, and $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$.

I.e., we must prove that for all c, c', all $\mathit{mds}, \mathit{mds}'$, all $\mathit{mem}_1, \mathit{mem}_2, \mathit{mem}_1'$, and all Γ, Γ' that satisfy $\langle c, \mathit{mds}, \mathit{mem}_1'\rangle \twoheadrightarrow \langle c', \mathit{mds}', \mathit{mem}_1'\rangle$, $\vdash \Gamma\ \{c\}\ \Gamma'$, $\mathit{mds} \in \mathit{comp}(\Gamma)$, and $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$, there exist c'' and mem_2' such that $\langle c, \mathit{mds}, \mathit{mem}_2\rangle \twoheadrightarrow \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$ and $\langle c', \mathit{mds}', \mathit{mem}_1'\rangle \mathcal{R}^{\Gamma'} \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$ are satisfied. We perform the proof of this statement by induction on the derivation of $\vdash \Gamma\ \{c\}\ \Gamma'$.

Rule [anno]: In this case, c is of the form $/\!\!/\mathit{ann}/\!\!/\ c_1$. As $\vdash \Gamma\ \{c\}\ \Gamma'$ is derived with the rule [anno], we know that when defining $\Gamma'' = \Gamma \oplus \mathit{ann}$, then $\vdash \Gamma''\ \{c_1\}\ \Gamma'$ and $\forall x \in \mathit{Var} : \Gamma\langle x\rangle \sqsubseteq \Gamma''\langle x\rangle$ hold.

As $\langle c, \mathit{mds}, \mathit{mem}_1'\rangle \twoheadrightarrow \langle c', \mathit{mds}', \mathit{mem}_1'\rangle$, we know by the definition of the operational semantics that $\langle c_1, \mathit{update}(\mathit{mds}, \mathit{ann}), \mathit{mem}_1\rangle \twoheadrightarrow \langle c', \mathit{mds}', \mathit{mem}_1'\rangle$. By the definitions of update and $\oplus$, we obtain that $\mathit{update}(\mathit{mds}, \mathit{ann}) \in \mathit{comp}(\Gamma'')$. As $\Gamma\langle x\rangle \sqsubseteq \Gamma''\langle x\rangle$ for all $x \in \mathit{Var}$ and as $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$, we also have $\mathit{mem}_1 =_{\Gamma''} \mathit{mem}_2$. Hence, we can apply the induction hypothesis obtained from $\vdash \Gamma''\ \{c_1\}\ \Gamma'$ to obtain c'' and mem_2' such that $\langle c_1, \mathit{update}(\mathit{mds}, \mathit{ann}), \mathit{mem}_2\rangle \twoheadrightarrow \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$ and such that $\langle c', \mathit{mds}', \mathit{mem}_1'\rangle \mathcal{R}^{\Gamma'} \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$. By the definition of the operational semantics, we then obtain also that $\langle /\!\!/\mathit{ann}/\!\!/\ c_1, \mathit{mds}, \mathit{mem}_2\rangle \twoheadrightarrow \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$, which concludes the proof for this case.

Rule [skip]: In this case, $c = \mathsf{skip}$. Due to the operational semantics for the command skip, we have $c' = \mathsf{stop}$, $\mathit{mds}' = \mathit{mds}$, and $\mathit{mem}_1' = \mathit{mem}_1$. We set $c'' = \mathsf{stop}$ and $\mathit{mem}_2' = \mathit{mem}_2$. Then, evidently, $\langle c, \mathit{mds}, \mathit{mem}_2\rangle \twoheadrightarrow \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$. It remains to show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}^{\Gamma'} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2\rangle$. By the definition of the rule [skip] we have $\Gamma' = \Gamma$, and it hence suffices to show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}^{\Gamma} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2\rangle$. For this, it suffices to show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1\rangle \mathcal{R}_1^{\Gamma} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2\rangle$. This is straightforward, as $\vdash \Gamma\ \{\mathsf{stop}\}\ \Gamma$, and $\mathit{mds} \in \mathit{comp}(\Gamma)$ as well as $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$ are satisfied by assumption.

Rule [assign$_1$]: In this case, $c = x{:=}e$. Let $v_1, v_2 \in \mathit{Val}$ be those values such that $\langle e, \mathit{mem}_1\rangle \downarrow v_1$ and $\langle e, \mathit{mem}_2\rangle \downarrow v_2$. Then $c' = \mathsf{stop}$, $\mathit{mds}' = \mathit{mds}$, and $\mathit{mem}_1' = \mathit{mem}_1[x \mapsto v_1]$. We set $c'' = \mathsf{stop}$ and $\mathit{mem}_2' = \mathit{mem}_2[x \mapsto v_2]$. Then, by the operational semantics, $\langle c, \mathit{mds}, \mathit{mem}_2\rangle \twoheadrightarrow \langle c'', \mathit{mds}', \mathit{mem}_2'\rangle$. As $\Gamma' = \Gamma$ in rule [assign$_1$], it remains to show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1[x \mapsto v_1]\rangle \mathcal{R}^{\Gamma} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2[x \mapsto v_2]\rangle$. For this, it suffices to show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1[x \mapsto v_1]\rangle \mathcal{R}_1^{\Gamma} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2[x \mapsto v_2]\rangle$. As $\vdash \Gamma\ \{\mathsf{stop}\}\ \Gamma$, for this it suffices to show that $\mathit{mem}_1[x \mapsto v_1] =_\Gamma \mathit{mem}_2[x \mapsto v_2]$. As $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$, we must hence show that $v_1 = v_2$ if $\Gamma\langle x\rangle = \mathit{low}$. As $x \notin \mathit{dom}(\Gamma)$ by the conditions of rule [assign$_1$], we then have $\mathcal{L}(x) = \mathit{low}$. By the remaining conditions of rule [assign$_1$], we then have $\Gamma \vdash e : \mathit{low}$. Hence, $\Gamma\langle x'\rangle = \mathit{low}$ for all $x' \in \mathit{vars}(e)$. As $\mathit{mem}_1 =_\Gamma \mathit{mem}_2$ and the value of e depends only on the values of variables in $\mathit{vars}(e)$, we hence have $v_1 = v_2$.

Rule [assign$_2$]: The proof in this case in analogous to the previous case, but for the fact that here $\Gamma'' = \Gamma[x \mapsto t]$ where $\Gamma \vdash e : t$ and we hence must show that $\langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_1[x \mapsto v_1]\rangle \mathcal{R}_1^{\Gamma[x \mapsto t]} \langle \mathsf{stop}, \mathit{mds}, \mathit{mem}_2[x \mapsto v_2]\rangle$. As $\vdash \Gamma[x \mapsto t]\ \{\mathsf{stop}\}\ \Gamma[x \mapsto t]$, it remains to show that $\mathit{mds} \in \mathit{comp}(\Gamma[x \mapsto t])$ and that $\mathit{mem}_1 =_{\Gamma[x \mapsto t]} \mathit{mem}_2$. The first statement is evident as $\mathit{dom}(\Gamma) = \mathit{dom}(\Gamma[x \mapsto t])$ $\mathit{mds}' = \mathit{mds}$. For the second statement, we have to show that $\mathit{mem}_1(x) = \mathit{mem}_2(x)$ if $t = \mathit{low}$. But if $t = \mathit{low}$, then (as $\Gamma \vdash e : t$ is a precondition of rule [assign$_2$]) the value of e is equal in mem_1 and mem_2 due to $mem_1 =_\Gamma mem_2$, and, hence, $v_1 = v_2$.

Rule [if]: Here, $c = \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\ \mathsf{fi}$. Then either $c' = c_1$ or $c' = c_2$. Moreover, $mds' = mds$ and $mem'_1 = mem_1$. We distinguish the two cases $\Gamma \vdash e : low$ and $\Gamma \vdash e : high$.

Let us firstly assume that $\Gamma \vdash e : low$. Then, due to $mem_1 =_\Gamma mem_2$, the value of e is equal in both mem_1 and mem_2. We set $c'' = c'$ and $mem'_2 = mem_2$. Then, evidently, $\langle c, mds, mem_2\rangle \rightsquigarrow \langle c'', mds, mem'_2\rangle$. It remains to show that $\langle c', mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c', mds, mem_2\rangle$. As due to the conditions of rule [if] we have $\vdash \Gamma\ \{c_1\}\ \Gamma'$ and $\vdash \Gamma\ \{c_2\}\ \Gamma'$, we evidently have $\langle c', mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c', mds, mem_2\rangle$ and we can conclude.

Let us now assume that $\Gamma \vdash e : high$. If the value of e is equal in mem_1 and in mem_2, we can proceed as in the previous case. Hence, we assume that the value is not equal. Hence, the transition of c in mem_1 and mem_2 results in c_1 for the one memory state, and in c_2 for the other memory state. We assume without loss of generality that $c' = c_1$, and set $c'' = c_2$ as well as $mem'_2 = mem_2$. Then, evidently, $\langle c, mds, mem_2\rangle \rightsquigarrow \langle c'', mds, mem'_2\rangle$, and it remains to show that $\langle c_1, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c_2, mds, mem_2\rangle$. We will show that this holds because $\langle c_1, mds, mem_1\rangle \mathcal{R}_2^{\Gamma'} \langle c_2, mds, mem_2\rangle$ holds. From the conditions of rule [if], we know that $\vdash \Gamma\ \{c_1\}\ \Gamma'$ and that $\vdash \Gamma\ \{c_2\}\ \Gamma'$ hold. As the mode state remains unchanged in the transition, we have $mds' \in comp(\Gamma)$ as well as $mds' \in comp(\Gamma)$. Moreover, we know from the conditions of rule [if] that $\Gamma'(x) = high$ for all $x \in dom(\Gamma')$. It remains to show that $\langle c_1, mds, mem_1\rangle \approx \langle c_2, mds, mem_2\rangle$. Again, from the conditions of rule [if] we know that $c_1 \sim_{low}^{mds} c_2$ holds for all $mds \in comp(\Gamma)$. Hence, if $mem_1 =_{low}^{mds} mem_2$ then $\langle c_1, mds, mem_1\rangle \approx \langle c_2, mds, mem_2\rangle$ by the definition of the relation $\sim_{low}^{mds}$. To show this, let $x \in Var$ with $\mathcal{L}(x) = low$ and $x \notin mds(asm\text{-}noread)$. Then, as $mds \in comp(\Gamma)$, $x \notin dom(\Gamma)$ and hence $\Gamma\langle x\rangle = low$. As $mem_1 =_\Gamma mem_2$, it follows that $mem_1(x) = mem_2(x)$. In consequence, $mem_1 =_{low}^{mds} mem_2$.

Rule [while]: In this case, $c = \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}$, $\Gamma' = \Gamma$, $\Gamma \vdash e : low$, and $\vdash \Gamma\ \{c\}\ \Gamma$. Hence, $c' = \mathsf{if}\ e\ \mathsf{then}\ c; \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\ \mathsf{else}\ \mathsf{stop}\ \mathsf{fi}$, $mds' = mds$, and $mem'_1 = mem_1$. We set $c'' = c'$ and $mem'_2 = mem_2$. We will show that $\langle c', mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c'', mds', mem''_2\rangle$ by showing that $\langle c', mds', mem'_1\rangle \mathcal{R}_1^{\Gamma'} \langle c'', mds', mem''_2\rangle$, i.e., that

$$\langle \mathsf{if}\ e\ \mathsf{then}\ c; \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\ \mathsf{else}\ \mathsf{stop}\ \mathsf{fi}, mds, mem_1\rangle \mathcal{R}_1^{\Gamma}$$
$$\langle \mathsf{if}\ e\ \mathsf{then}\ c; \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\ \mathsf{else}\ \mathsf{stop}\ \mathsf{fi}, mds', mem_2\rangle.$$

For this, it suffices to show that $\vdash \Gamma\ \{\mathsf{if}\ e\ \mathsf{then}\ c; \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\ \mathsf{else}\ \mathsf{stop}\ \mathsf{fi}\}\ \Gamma$ is derivable. As by our assumptions $\Gamma \vdash e : low$, $\vdash \Gamma\ \{c\}\ \Gamma$, $\vdash \Gamma\ \{\mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\}\ \Gamma$, and as $\vdash \Gamma\ \{\mathsf{stop}\}\ \Gamma$, this follows directly by applying the typing rules [seq] and [if].

Rule [seq]: Here, $c = c_1; c_2$. By the conditions of rule [seq], there is a type environment Γ'' such that $\vdash \Gamma\ \{c_1\}\ \Gamma''$ and such that $\vdash \Gamma''\ \{c_2\}\ \Gamma'$. In the following, we distinguish two cases, namely $c_1 = \mathsf{stop}$ and $c_1 \neq \mathsf{stop}$.

Let us firstly assume that $c_1 = \mathsf{stop}$. Then, by the operational semantics, we have $c' = c_2$, $mds' = mds$, and $mem'_1 = mem_1$. We set $c'' = c_2$ and $mem'_2 = mem_2$. Then, evidently, $\langle c, mds, mem_2\rangle \rightsquigarrow \langle c_2, mds, mem'_2\rangle$. It remains to show that $\langle c_2, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c_2, mds, mem_2\rangle$. For showing this, we show that $\langle c_2, mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c_2, mds, mem_2\rangle$. As $\vdash \Gamma\ \{c_1\}\ \Gamma''$, by the conditions of the rule [stop] we have that $\Gamma = \Gamma''$. Hence, $\vdash \Gamma\ \{c_2\}\ \Gamma'$. As, by assumption, $mds \in comp(\Gamma)$ and $mem_1 =_\Gamma mem_2$, this concludes the case $c_1 = \mathsf{stop}$.

Now assume that $c_1 \neq \mathsf{stop}$. Then, by the definition of the operational semantics, there exists c'_1 such that $\langle c_1, mds, mem_1\rangle \rightsquigarrow \langle c'_1, mds', mem'_1\rangle$ and such that $c' = c'_1; c_2$. We apply the induction hypothesis obtained from $\vdash \Gamma\ \{c_1\}\ \Gamma''$. This provides us with c''_1 and mem''_2 such that $\langle c_1, mds, mem_2\rangle \rightsquigarrow \langle c''_1, mds', mem''_2\rangle$ and $\langle c'_1, mds', mem'_1\rangle \mathcal{R}^{\Gamma''} \langle c''_1, mds', mem''_2\rangle$. We set $c'' = c''_1; c_2$ and $mem'_2 = mem''_2$. Then, by the definition of the operational semantics, we have $\langle c, mds, mem_2\rangle \rightsquigarrow \langle c'', mds', mem'_2\rangle$. It remains to show that $\langle c'_1; c_2, mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c''_1; c_2, mds', mem'_2\rangle$. For showing this, we distinguish whether $\langle c'_1, mds', mem'_1\rangle$ and $\langle c''_1, mds', mem'_2\rangle$ are related by $\mathcal{R}_1^{\Gamma''}$, $\mathcal{R}_2^{\Gamma''}$, or $\mathcal{R}_3^{\Gamma''}$.

Assume firstly that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}_1^{\Gamma''} \langle c''_1, mds', mem'_2\rangle$. Then $c'_1 = c''_1$ and there exists Γ''' such that $\vdash \Gamma'''\ \{c'_1\}\ \Gamma''$, $mds' \in comp(\Gamma''')$ and $mem'_1 =_{\Gamma'''} mem'_2$. But then, we also have $\vdash \Gamma'''\ \{c'_1; c_2\}\ \Gamma'$ by the rule [seq], and hence $\langle c'_1; c_2, mds', mem'_1\rangle \mathcal{R}_1^{\Gamma'} \langle c''_1; c_2, mds', mem'_2\rangle$.

Assume now that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}_2^{\Gamma''} \langle c''_1, mds', mem'_2\rangle$. As $\vdash \Gamma''\ \{c_2\}\ \Gamma'$, we obtain directly that in this case $\langle c'_1; c_2, mds', mem'_1\rangle \mathcal{R}_3^{\Gamma'} \langle c''_1; c_2, mds', mem'_2\rangle$.

Assume finally that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}_3^{\Gamma''} \langle c''_1, mds', mem'_2\rangle$. Then, $c'_1 = c'^*_1; c^*$ and $c''_1 = c''^*_1; c^*$ for some commands c'^*_1, c''^*_1, and c^*. Moreover, there exists a type environment Γ''' such that $\langle c'^*_1, mds', mem'_1\rangle \mathcal{R}_2^{\Gamma'''} \langle c''^*_1, mds', mem'_2\rangle$ and $\vdash \Gamma'''\ \{c^*\}\ \Gamma''$. As $\vdash \Gamma''\ \{c_2\}\ \Gamma'$, we obtain $\vdash \Gamma'''\ \{c^*; c_2\}\ \Gamma'$. In consequence, $\langle c'^*_1; c^*; c_2, mds', mem'_1\rangle \mathcal{R}_3^{\Gamma'} \langle c''^*_1; c^*; c_2, mds', mem'_2\rangle$, i.e., $\langle c', mds', mem'_1\rangle \mathcal{R}_3^{\Gamma'} \langle c'', mds', mem'_2\rangle$.

Rule [sub]: In this case, there are type environments Γ_1 and Γ'_1 such that $\vdash \Gamma_1\ \{c\}\ \Gamma'_1$, $\Gamma \sqsubseteq \Gamma_1$, and $\Gamma'_1 \sqsubseteq \Gamma'$.

Firstly note that $comp(\Gamma_1) = comp(\Gamma)$ and that $comp(\Gamma'_1) = comp(\Gamma')$, as the domains of type environments related by $\sqsubseteq$ are equal.

From the induction hypothesis obtained from $\vdash \Gamma_1\ \{c\}\ \Gamma'_1$, we know that whenever

$\langle c, mds, mem'_1\rangle \twoheadrightarrow c', mds', mem'_1$, $mds \in comp(\Gamma_1)$, and $mem_1 =_{\Gamma_1} mem_2$, then there exist c'' and mem'_2 such that $\langle c, mds, mem_2\rangle \twoheadrightarrow \langle c'', mds', mem'_2\rangle$ and $\langle c', mds', mem'_1\rangle \mathcal{R}^{\Gamma'_1} \langle c'', mds', mem'_2\rangle$ hold. To apply this induction hypothesis, we establish that $mds \in comp(\Gamma_1)$, and $mem_1 =_{\Gamma_1} mem_2$. The first statement is evident as $comp(\Gamma_1) = comp(\Gamma)$ and $mds \in comp(\Gamma)$ by assumption. For the second statement, let $x \in Var$ such that $\Gamma_1\langle x\rangle = low$. As $\Gamma \sqsubseteq \Gamma_1$, this implies $\Gamma\langle x\rangle = low$, and, due to the assumption that $mem_1 =_\Gamma mem_2$ we obtain that $mem_1(x) = mem_2(x)$. Hence, we can apply the induction hypothesis and obtain c'' and mem'_2 as stated above. To conclude, we must show that $\langle c', mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c'', mds', mem'_2\rangle$. As $\langle c', mds', mem'_1\rangle \mathcal{R}^{\Gamma'_1} \langle c'', mds', mem'_2\rangle$ holds, is suffices to show that $\mathcal{R}^{\Gamma'_1} \subseteq \mathcal{R}^{\Gamma'}$ whenever $\Gamma'_1 \sqsubseteq \Gamma'$. To this end, we must show that $comp(\Gamma'_1) \subseteq comp(\Gamma_1)$ (which is evident as $comp(\Gamma'_1) = comp(\Gamma_1)$ and that $mem_1 =_{\Gamma'_1} mem_2$ implies that $mem_1 =_{\Gamma'} mem_2$. For this, let $x \in Var$ such that $\Gamma'_1\langle x\rangle = low$. Then, as $\Gamma'_1 \sqsubseteq \Gamma'$, also $\Gamma'\langle x\rangle = low$, and hence $mem_1(x) = mem_2(x)$. This concludes the case for rule [sub].

Case 2: $\langle c_1, mds, mem_1\rangle \mathcal{R}_2^{\Gamma'} \langle c_2, mds, mem_2\rangle$**.** In this case, we know that $\langle c_1, mds, mem_1\rangle \approx \langle c_2, mds, mem_2\rangle$ and $\forall x \in dom(\Gamma') : \Gamma'(x) = high$, and that there exist type environments Γ_1 and Γ_2 such that $\vdash \Gamma_1 \; \{c_1\} \; \Gamma'$, $\vdash \Gamma_2 \; \{c_2\} \; \Gamma'$, $mds \in comp(\Gamma_1)$, and $mds \in comp(\Gamma_2)$. Assume that $\langle c_1, mds, mem_1\rangle \twoheadrightarrow \langle c'_1, mds', mem'_1\rangle$. We must show that there exist c'_2 and mem'_2 such that $\langle c_2, mds, mem_2\rangle \twoheadrightarrow \langle c'_2, mds', mem'_2\rangle$ and such that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c'_2, mds', mem'_2\rangle$.

As $\langle c_1, mds, mem_1\rangle \approx \langle c_2, mds, mem_2\rangle$, there are c'_2 and mem'_2 with $\langle c_2, mds, mem_2\rangle \twoheadrightarrow \langle c'_2, mds', mem'_2\rangle$ and $\langle c'_1, mds', mem'_1\rangle \approx \langle c'_2, mds', mem'_2\rangle$. We will show that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}_2^{\Gamma'} \langle c'_2, mds', mem'_2\rangle$. For this, it remains to show that there are type environments Γ'_1 and Γ'_2 such that $\vdash \Gamma'_1 \; \{c'_1\} \; \Gamma'$, $\vdash \Gamma'_2 \; \{c'_2\} \; \Gamma'$, $mds' \in comp(\Gamma'_1)$, and $mds' \in comp(\Gamma'_2)$. But this we have already shown for all possible transitions in the proof of Case 1 (i.e., the case $\langle c_1, mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c_2, mds, mem_2\rangle$).

Case 3: $\langle c_1, mds, mem_1\rangle \mathcal{R}_3^{\Gamma'} \langle c_2, mds, mem_2\rangle$**.** In this case, we know that there are commands c_1^*, c_2^*, and c^* such that $c_1 = c_1^*; c^*$ and $c_2 = c_2^*; c^*$, and that there exists Γ'' such that $\langle c_1^*, mds, mem_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2^*, mds, mem_2\rangle$ and such that $\vdash \Gamma'' \; \{c^*\} \; \Gamma'$. Assume that $\langle c_1, mds, mem_1\rangle \twoheadrightarrow \langle c'_1, mds', mem'_1\rangle$. We need to show that then there exist c'_2 and mem'_2 such that $\langle c_2, mds, mem_2\rangle \twoheadrightarrow \langle c'_2, mds', mem'_2\rangle$ and such that $\langle c'_1, mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c'_2, mds', mem'_2\rangle$.

We distinguish the two cases that $c_1^* \neq \mathsf{stop}$ and that $c_1^* = \mathsf{stop}$.

Let us assume firstly that $c_1^* \neq \mathsf{stop}$. Then we know that there exists $c_1'^*$ such that $\langle c_1^*, mds, mem_1\rangle \twoheadrightarrow \langle c_1'^*, mds', mem'_1\rangle$, where $c'_1 = c_1'^*; c^*$. But then we know from $\langle c_1^*, mds, mem_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2^*, mds, mem_2\rangle$ and the previous case (Case 2) that then there exists $c_2'^*$ and mem'_2 such that $\langle c_2^*, mds, mem_2\rangle \twoheadrightarrow \langle c_2'^*, mds', mem'_2\rangle$ and such that $\langle c_1'^*, mds', mem'_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2'^*, mds', mem'_2\rangle$. We set $c'_2 = c_2'^*; c^*$. Then, by the operational semantics, $\langle c_2, mds, mem_2\rangle \twoheadrightarrow \langle c'_2, mds', mem'_2\rangle$. Moreover, $\langle c'_1, mds', mem'_1\rangle \mathcal{R}_3^{\Gamma'} \langle c'_2, mds', mem'_2\rangle$ holds as $\langle c_1'^*, mds', mem'_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2'^*, mds', mem'_2\rangle$ and $\vdash \Gamma'' \; \{c^*\} \; \Gamma'$ both hold. Hence, $\langle c'_1, mds', mem'_1\rangle \mathcal{R}^{\Gamma'} \langle c'_2, mds', mem'_2\rangle$.

Let us now assume that $c_1^* = \mathsf{stop}$. As $c_1^* = \mathsf{stop}$, we know that $com'_1 = c^*$, $mds' = mds$, and that $mem'_1 = mem_1$. We set $c'_2 = c^*$ and $mem'_2 = mem_2$. We also have $c_2^* = \mathsf{stop}$, as $\langle c_1^*, mds, mem_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2^*, mds, mem_2\rangle$ and we know from Case (2) that if a local configuration containing c_2^* can make a step, then every configuration related by $\mathcal{R}_2^{\Gamma''}$ can make a step. As $c_1^* = \mathsf{stop}$, we know that $com'_1 = c^*$. Hence, $\langle c_2, mds, mem_2\rangle \twoheadrightarrow \langle c^*, mds', mem'_2\rangle$. It remains to show that $\langle c^*, mds, mem_1\rangle \mathcal{R}^{\Gamma'} \langle c^*, mds, mem_2\rangle$. We show this by showing that $\langle c^*, mds, mem_1\rangle \mathcal{R}_1^{\Gamma'} \langle c^*, mds, mem_2\rangle$. We set the type environment required by the definition of $\mathcal{R}_1^{\Gamma'}$ to Γ''. We must hence show that $\vdash \Gamma'' \; \{c^*\} \; \Gamma'$, that $mds \in comp(\Gamma'')$, and that $mem_1 =_{\Gamma''} mem_2$. The first property is clear by assumption. Moreover, $\langle c_1^*, mds, mem_1\rangle \mathcal{R}_2^{\Gamma''} \langle c_2^*, mds, mem_2\rangle$ implies that $mds \in comp(\Gamma'')$, $\langle c_1^*, mds, mem_1\rangle \approx \langle c_2^*, mds, mem_2\rangle$, and $\Gamma''(x) = high$ for all $x \in dom(\Gamma'')$. Hence, $mem_1 =_{low}^{mds} mem_2$. We finally show that $mem_1 =_{\Gamma''} mem_2$. For this, let $x \in Var$ such that $\Gamma''\langle x\rangle = low$. Hence, $x \notin dom(\Gamma'')$ as $\Gamma''(x) = high$ for all $x \in dom(\Gamma'')$. In consequence, $\mathcal{L}(x) = low$ and $mem_1(x) = mem_2(x)$ holds due to $mem_1 =_{low}^{mds} mem_2$ ■

VIII. Proof of Theorem 6

Theorem 6. Let $c_1, \ldots, c_n$ be commands such that the judgment $\vdash c_1, \ldots, c_n$ is derivable. Then the program consisting of the commands $c_1, \ldots, c_n$ is SIFUM-secure.

Proof: By applying Theorems 5 (for each c_i) and Theorem 1. ■

IX. Proof of Theorem 7

Theorem 7. Let $p = \langle (c_1, mds_0), \ldots, (c_n, mds_0)\rangle$ be a list of pairs of commands and mode states such that $c_1, \ldots, c_n$ are SIFUM-secure and $\langle p, mem\rangle$ ensures a sound use of modes for all mem. Then for all k, all p', all $mem_1 =_{low} mem_2$, and all mem'_1 with $\langle p, mem_1\rangle \rightarrow^k \langle p', mem'_1\rangle$ there exist p'' and mem'_2 with $\langle p, mem_2\rangle \rightarrow^k \langle p'', mem'_2\rangle$ and $mem'_1(\mathsf{out}) = mem'_2(\mathsf{out})$.

Proof: The theorem follows directly from Proposition 1. ■

X. Type System for Locally Sound Use of Modes

We use typing judgments of the form $\vdash mds\ \{c\}\ mds'$, where c is a command and mds, mds' are mode states. The intuition is that if c is executed in mode state mds, then mds' safely describes the mode state after the command has executed, where "safe" means an upper bound on each mode. The type system is displayed in Figure 1. In the typing rules, we use a subset relation on mode states, where $mds \subseteq mds'$ if and only if $mds(m) \subseteq mds'(m)$ for all modes m. Moreover, we write $annos$ for a (possibly empty) sequence $/\!\!/ann_1/\!\!/ \ldots /\!\!/ann_n/\!\!/$ of annotations, and $update(mds, annos)$ for the mode state obtained by updating mds subsequently with the annotations $ann_1, \ldots, ann_n$. Besides approximating the mode state in a sound way, the type system also ensures that all guarantees given in the mode state are adhered to by the command, i.e., the command ensures a locally sound use of modes.

Theorem 8. Let c be a command. Assume that $\vdash mds_1\ \{c\}\ mds_1'$ is derivable in the type system. Then for all mode states mds_2, mds_2' with $mds_2 \subseteq mds_1$ and all memory states mem, mem', if $\langle c, mds_2, mem\rangle \rightarrow^* \langle \mathsf{stop}, mds_2', mem'\rangle$ then the following hold:

(1) $mds_2' \subseteq mds_1'$,
(2) For all memory states $mem \in Mem$, $\langle c, mds_2, mem\rangle$ ensures a locally sound use of modes

Proof: The proof is by induction on the derivation of the judgment $\vdash mds\ \{c\}\ mds'$.

Rule [skip]. In this case, $c = annos$ skip, and $mds_1' = update(mds_1, annos)$. Moreover, by the operational semantics, we have $mds_2' = update(mds_2, annos)$. Hence, we need to show that $update(mds_2, annos) \subseteq update(mds_1, annos)$. This follows directly from the definition of the function $update$ and the assumption that $mds_2 \subseteq mds_1$. Moreover, we have to show that $\langle \mathsf{skip}, mds_2, mem\rangle$ ensures a locally sound use of modes. This follows from the fact that for all $x \in Var$, skip does not read x and skip does not modify x, and that the commands in all local configurations that are locally reachable from $\langle \mathsf{skip}, mds_2, mem\rangle$ equal either skip or stop, which neither read nor modify x.

Rule [assign]. In this case, $c = annos\ x{:=}e$. Showing that $mds_2' \subseteq mds_1'$ is as in the proof for rule [skip]. It remains to show that $lc = \langle x{:=}e, mds_2, mem\rangle$ ensures a locally sound use of modes. Let $\langle c^*, mds^*, mem^*\rangle \in lReach(lc)$. Then either $c^* = \mathsf{stop}$ (a command that neither reads nor modifies any variable) or $c^* = x{:=}e$ and $mds' = mds$. By the preconditions of rule [assign] we know that $x \notin mds_1(guar\text{-}nowrite)$. Hence, we must show that for all $x' \neq x$ the assignment $x{:=}e$ does not read x'. But this is evident from the operational semantics. Moreover, by the preconditions of rule [assign] we know that $vars(e) \cap mds(guar\text{-}noread) = \{\}$. Hence, we must show for all $x' \notin vars(e)$ that the assignment $x{:=}e$ does not read x'. This follows from the operational semantics for assignments and the fact that the value of the expression e is unchanged when modifying the values of variables that are not contained in $vars(e)$.

Rule [if]. In this case, $c = annos$ if e then c_1 else c_2 fi. Hence, $\langle c, mds_2, mem\rangle \rightarrow \langle c_i, update(mds_2, annos), mem\rangle$ and $\langle c_i, update(mds_2, annos), mem\rangle \rightarrow^* \langle \mathsf{stop}, mds_2', mem'\rangle$ for either $i = 1$ or $i = 2$. As $mds_2 \subseteq mds_1$, we have $update(mds_2, annos) \subseteq update(mds_1, annos)$. As we have $\vdash update(mds_1, annos)\ \{c_i\}\ mds_1'$ for both $i = 1$ and $i = 2$ by the conditions of rule [if], we can apply an induction hypothesis for both $i = 1$ and $i = 2$ and obtain that $mds_2' \subseteq mds_1'$. Moreover, we obtain that $\langle c_i, update(mds_2, annos), mem\rangle$ ensures a sound use of modes for all mem. Hence, it remains to show that if e then c_1 else c_2 fi does not read variables x if $x \in mds_2(guar\text{-}noread)$ and does not modify variables x with $x \in mds_2(guar\text{-}nowrite)$. The second statement is clear as the evaluation of the guard does not modify any variables (compare the operational semantics). If $x \in mds_2(guar\text{-}noread)$, then by the conditions of rule [if] and the fact that $mds_2 \subseteq mds_1$ we have $x \notin vars(e)$. But then the value of e does not depend on x. Hence, if $\langle c, mds_2, mem\rangle \rightarrow \langle c_i, update(mds_2, annos), mem\rangle$, then the value of i is not modified when modifying the value of x. Hence, c does not read x.

Rule [while]. In this case, $c = annos$ while e do c_1 od. We do the proof by induction on the number n of loop iterations (the number is finite, as $\langle c, mds_2, mem\rangle \rightarrow^* \langle \mathsf{stop}, mds_2', mem'\rangle$). If $n = 0$, the proof is evident (arguing as for rule [if] that c does not read any variables with mode $guar\text{-}noread$, as these variables do not occur in $vars(e)$). If $n > 0$, then by the definition of the semantics of while loops, we have $\langle c, mds_2, mem\rangle \rightarrow \langle$if e then c'; while e do c' od else stop fi$, mds_2'', mem'\rangle \rightarrow \langle c';$ while e do c' od$, mds_2'', mem'\rangle \rightarrow^* \langle \mathsf{stop}, mds_2', mem'\rangle$, where $mds_2'' = update(mds_2, annos)$. From the conditions of rule [while] we have $\vdash mds_2''\ \{c'\}\ mds_2''$. Hence, we can apply the induction hypothesis and obtain that $\langle c', mds_2'', mem\rangle$ ensures a locally sound use of modes for all mem, and that if $\langle c', mds_2'', mem\rangle \rightarrow^* \langle \mathsf{stop}, mds_2''', mem_2''\rangle$ we have $mds_2''' \subseteq mds_2$. Hence, we can apply the induction hypothesis for $n-1$ for the while loop, and obtain that $\langle$while e do c' od$, mds_2''', mem''\rangle$ ensures a locally sound use of modes, and that if $\langle$while e do c' od$, mds_2''', mem''\rangle \rightarrow \langle \mathsf{stop}, mds_2', mem'\rangle$ then $mem_2' \subseteq mem_1$.

Rule [seq]. In this case, $c = c_1; c_2$ and there is a mode state mds_1'' such that $\vdash mds_1\ \{c_1\}\ mds_1''$ and $\vdash mds_1''\ \{c_2\}\ mds_1'$ are derivable in the type system. Assume that $\langle c_1; c_2, mds_2, mem\rangle \rightarrow^* \langle \mathsf{stop}, mds_2', mem'\rangle$. Then,

$$\text{[skip]}\frac{mds' = update(mds, annos)}{\vdash mds\ \{annos\ \mathsf{skip}\}\ mds'} \qquad \text{[assign]}\frac{\begin{array}{c} mds' = update(mds, annos) \\ x \notin mds(guar\text{-}nowrite) \quad vars(e) \cap mds(guar\text{-}noread) = \{\} \end{array}}{\vdash mds\ \{annos\ x{:=}e\}\ mds'}$$

$$\text{[if]}\frac{\begin{array}{c} mds' = update(mds, annos) \\ \vdash mds'\ \{c_1\}\ mds'' \quad \vdash mds'\ \{c_2\}\ mds'' \\ vars(e) \cap mds(guar\text{-}noread) = \{\} \end{array}}{\vdash mds\ \{annos\ \mathsf{if}\ e\ \mathsf{then}\ c_1\ \mathsf{else}\ c_2\ \mathsf{fi}\}\ mds''} \qquad \text{[while]}\frac{\begin{array}{c} mds' = update(mds, annos) \\ \vdash mds'\ \{c\}\ mds' \\ vars(e) \cap mds'(guar\text{-}noread) = \{\} \end{array}}{\vdash mds\ \{annos\ \mathsf{while}\ e\ \mathsf{do}\ c\ \mathsf{od}\}\ mds'}$$

$$\text{[seq]}\frac{\vdash mds\ \{c_1\}\ mds' \quad \vdash mds'\ \{c_2\}\ mds''}{\vdash mds'\ \{c_1; c_2\}\ mds''} \qquad \text{[sub]}\frac{\begin{array}{c} \vdash mds_2\ \{c\}\ mds'_2 \\ mds_1 \subseteq mds_2 \quad mds'_2 \subseteq mds'_1 \end{array}}{\vdash mds_1\ \{c\}\ mds'_1}$$

Figure 1. Type system for locally sound use of modes

by the operational semantics, there exist mds''_2 and mem'' such that $\langle c_1, mds_2, mem\rangle \rightarrow\!\!\triangleright^* \langle \mathsf{stop}, mds''_2, mem''\rangle$ and $\langle c_2, mds''_2, mem''\rangle \rightarrow\!\!\triangleright^* \langle \mathsf{stop}, mds'_2, mem'\rangle$. We firstly apply the induction hypothesis for c_1 and obtain that $mds''_2 \subseteq mds''_1$. Hence, we can apply the induction hypothesis for c_1 and obtain that $mds'_2 \subseteq mds'_1$. It remains to show that $\langle c, mds_2, mem\rangle$ ensures a locally sound use of modes for all mem. But this follows also from the induction hypotheses, which guarantees that $\langle c_1, mds_2, mem\rangle$ and $\langle c_2, mds''_2, mem\rangle$ ensure a locally sound use of modes for all mem.

Rule [sub]. From the conditions of rule [sub] we know that there exist mode states mds_3, mds'_3 such that $\vdash mds_3\ \{c\}\ mds'_3$, $mds_1 \subseteq mds_3$, and $mds'_3 \subseteq mds'_1$. From $mds_2 \subseteq mds_1$ we hence obtain $mds_2 \subseteq mds_3$. We apply the induction hypothesis obtained from $\vdash mds_3\ \{c\}\ mds'_3$, and obtain that $mds'_2 \subseteq mds'_3$. Hence, $mds'_2 \subseteq mds'_1$ follows from $mds'_3 \subseteq mds'_1$. From the induction hypothesis, we also obtain that $\langle c, mds_2, mem\rangle$ ensures a locally sound use of modes for all $mem \in Mem$. ■

References

[1] H. Mantel, D. Sands, and H. Sudbrock, "Assumptions and Guarantees for Compositional Noninterference," in *Proceedings of the IEEE Computer Security Foundations Symposium*. Vaux le Cernay, France: IEEE Computer Society, 2011, in press.