A Proofs for the Theorems in the Paper

A.1 Soundness of the DPN-based Analysis

In this subsection, we provide a proof for our soundness theorem for the DPN analysis (Theorem 1). The following table lists the dependencies between lemmas and theorems in this subsection.

Lemma/Theorem	Depends on lemmas/theorems
Lemma 1	none
Lemma 2	Lemma 1
Theorem 1	Lemmas 1, 2

In a first step, we proof that the semantics of a DPN $\mathcal{M}_{ccnf}$ for a control configuration $ccnf \in CCnf$ simulates the semantics of the any global configuration $\langle [ccnf], mem \rangle$. For this purpose, we define a simulation relation that relates global configurations to potential DPN configurations.

Definition 5. *The function* $toDPN : GCnf \to (CCnf\{\#\})^+$ *is defined by*

$$toDPN(\langle [ccnf_1, \ldots, ccnf_n], mem \rangle) = ccnf_1\# \ldots ccnf_n\#.$$

Lemma 1. *For* $ccnf \in CCnf$*, let* $\mathcal{M}_{ccnf} = (P_{ccnf}, \Gamma_{ccnf}, A_{ccnf}, \Delta_{ccnf})$ *be a DPN as described above. If* $toDPN(gcnf) = s$*,* $s \in DCnf$*, and* $gcnf \twoheadrightarrow gcnf'$*, then exists* s' *with* $toDPN(gcnf') = s'$*,* $s' \in DCnf$ *and* $s \twoheadrightarrow s'$*.*

Proof. Let $gcnf = \langle [ccnf_1, \ldots, ccnf_n], mem \rangle$. From $toDPN(gcnf) = s$ and $s \in DCnf$ we have $ccnf_i \in P_{ccnf}$ for all $1 \leq i \leq n$.

We consider the following two cases for $gcnf \twoheadrightarrow gcnf'$:

1. If $gcnf' = \langle [ccnf_1, \ldots, ccnf'_i, \ldots, ccnf_n], mem' \rangle$ for some $1 \leq i \leq n$ and there exists $\alpha \notin \{\nearrow_{ccnf} |\ ccnf \in CCnf\}$ with $(ccnf_i, mem) \xrightarrow{\alpha} (ccnf'_i, mem')$. Then also $ccnf'_i \in P_{ccnf}$, $\alpha \in A_{ccnf}$ and $(ccnf_i\#, \alpha, ccnf'_i\#) \in \Delta_{ccnf}$.
   From this we get $ccnf_1\# \ldots ccnf'_i\# \ldots ccnf_n\# \in DCnf$
   and $ccnf_1\# \ldots ccnf_i\# \ldots ccnf_n\# \twoheadrightarrow ccnf_1\# \ldots ccnf'_i\# \ldots ccnf_n\#$, since for $\alpha = l$ we have $l \notin locks([ccnf_1, \ldots, ccnf_n])$ and thus also
   $l \notin locks(ccnf_1\# \ldots ccnf_n\#)$.
   Furthermore $toDPN(gcnf') = ccnf_1\# \ldots ccnf'_i\# \ldots ccnf_n\#$ and thus the hypothesis.
2. If $gcnf' = \langle [ccnf_1, \ldots, ccnf_s, ccnf'_i, \ldots, ccnf_n], mem' \rangle$ for some $1 \leq i \leq n$ and there exists $\alpha = \nearrow_{ccnf_s}$ with $(ccnf_i, mem) \xrightarrow{\alpha} (ccnf'_i, mem')$. Then also $ccnf_s, ccnf'_i \in P_{ccnf}$, $\nearrow_{ccnf_s,\#} \in A_{ccnf}$ and $(ccnf_i\#, \nearrow_{ccnf_s,\#}, ccnf'_i\#) \in \Delta_{ccnf}$.
   From this we get $ccnf_1\# \ldots ccnf_s\#ccnf'_i\# \ldots ccnf_n\# \in DCnf$ and
   $ccnf_1\# \ldots ccnf_i\# \ldots ccnf_n\# \twoheadrightarrow ccnf_1\# \ldots ccnf_s\#ccnf'_i\# \ldots ccnf_n\#$. Furthermore $toDPN(gcnf') = ccnf_1\# \ldots ccnf_s\#ccnf'_i\# \ldots ccnf_n\#$ and thus the hypothesis.

□

Corollary 2. *For all* $gcnf \in gReach(\langle ccnf, mem \rangle)$ *there exists* $s \in DCnf$*, with* $toDPN(gcnf) = s$ *and* $ccnf\# \twoheadrightarrow^* s$*.*

Proof. This follows by simple induction from Lemma 1 and the fact that $ccnf \in P_{ccnf}$, $ccnf\# \in DCnf_{ccnf}$ and $toDPN(\langle [ccnf], mem \rangle) = ccnf\#$.

In the second step, we show that for all reachable configurations that violates sound use of modes there is a DPN configuration that is accepted by the automaton $\mathcal{A}_{ccnf}$.

Lemma 2. *Let command $c \in Com$, $gcnf \in GCnf$, and $\overrightarrow{mdst} \in MdSt^*$ be arbitrary. If $gcnf \in gReach(\langle (c, \emptyset, mdst_\perp), mem\rangle)$, $\overrightarrow{mdst}$ is the list of mode states in $gcnf$, and the list of mode states $\overrightarrow{mdst}$ violates at least one of the conditions*

- $\forall x, i, j. i \neq j \wedge x \in \overrightarrow{mdst}[i](\mathsf{A\text{-}NR}) \implies x \in \overrightarrow{mdst}[j](\mathsf{G\text{-}NR})$*, or*
- $\forall x, i, j. i \neq j \wedge x \in \overrightarrow{mdst}[i](\mathsf{A\text{-}NW}) \implies x \in \overrightarrow{mdst}[j](\mathsf{G\text{-}NW})$*,*

then $toDPN(gcnf) \in \mathcal{L}(\mathcal{A}_{\mathcal{M}_c})$.

Proof. Let command $c \in Com$, $gcnf \in GCnf$, and $\overrightarrow{mdst} \in MdSt^*$ be arbitrary such that $gcnf \in gReach(\langle (c, \emptyset, mdst_\perp), mem\rangle)$, $\overrightarrow{mdst}$ is the list of mode states in $gcnf$, and $\overrightarrow{mdst}$ violates at least one of the following conditions

C1: $\forall x, i, j. i \neq j \wedge x \in \overrightarrow{mdst}[i](\mathsf{A\text{-}NR}) \implies x \in \overrightarrow{mdst}[j](\mathsf{G\text{-}NR})$, or
C2: $\forall x, i, j. i \neq j \wedge x \in \overrightarrow{mdst}[i](\mathsf{A\text{-}NW}) \implies x \in \overrightarrow{mdst}[j](\mathsf{G\text{-}NW})$.

From the fact that $\overrightarrow{mdst}$ violates one of the two conditions C1 and C2, we know that there is a smallest index n such that there is an index m with $m < n$ and the pair of indices contributes to the violation, i.e. at least one of the following four conditions holds:

C3: $x \in \overrightarrow{mdst}[m](\mathsf{A\text{-}NR}) \wedge x \notin \overrightarrow{mdst}[n](\mathsf{G\text{-}NR})$ for some $x \in Var$, or
C4: $x \notin \overrightarrow{mdst}[m](\mathsf{G\text{-}NR}) \wedge x \in \overrightarrow{mdst}[n](\mathsf{A\text{-}NR})$ for some $x \in Var$, or
C5: $x \in \overrightarrow{mdst}[m](\mathsf{A\text{-}NW}) \wedge x \notin \overrightarrow{mdst}[n](\mathsf{G\text{-}NW})$ for some $x \in Var$, or
C6: $x \notin \overrightarrow{mdst}[m](\mathsf{G\text{-}NW}) \wedge x \in \overrightarrow{mdst}[n](\mathsf{A\text{-}NW})$ for some $x \in Var$.

From the fact that n is the smallest index such that there is a mode state at a smaller index that violates one of the four conditions, we get from the definition of $\oplus$ (Definition 3) that there is a mode state $mdst' \in MdSt$ with $mdst' = \oplus_{i=0}^{n-1} \overrightarrow{mdst}[i]$, $\overrightarrow{mdst}[m](\mathsf{A\text{-}NR}) \subseteq mdst'(\mathsf{A\text{-}NR})$, $mdst'(\mathsf{G\text{-}NR}) \subseteq \overrightarrow{mdst}[m](\mathsf{G\text{-}NR})$, $\overrightarrow{mdst}[m](\mathsf{A\text{-}NR}) \subseteq mdst'(\mathsf{A\text{-}NW})$, and $mdst'(\mathsf{G\text{-}NW}) \subseteq \overrightarrow{mdst}[m](\mathsf{G\text{-}NW})$. Hence, one of the following conditions holds:

C7: $x \in mdst'(\mathsf{A\text{-}NR}) \wedge x \notin \overrightarrow{mdst}[n](\mathsf{G\text{-}NR})$ for some $x \in Var$, or
C8: $x \notin mdst'(\mathsf{G\text{-}NR}) \wedge x \in \overrightarrow{mdst}[n](\mathsf{A\text{-}NR})$ for some $x \in Var$, or
C9: $x \in mdst'(\mathsf{A\text{-}NW}) \wedge x \notin \overrightarrow{mdst}[n](\mathsf{G\text{-}NW})$ for some $x \in Var$, or
C10: $x \notin mdst'(\mathsf{G\text{-}NW}) \wedge x \in \overrightarrow{mdst}[n](\mathsf{A\text{-}NW})$ for some $x \in Var$.

Thus, we get from definition of $\oplus$ that $mdst' \oplus \overrightarrow{mdst}[n] = \top$ and, in consequence, $\oplus_{i=0}^{len(\overrightarrow{mdst})-1} \overrightarrow{mdst}[i] = \top$ where $len(\overrightarrow{mdst})$ denotes the length of the list of mode states.

By Corollary 2 we obtain $s \in DCnf$ with $toDPN(gcnf) = s$. From the fact that $\overrightarrow{mdst}$ is the list of mode states in $gcnf$, we get by the definition of $toDPN$ that $\overrightarrow{mdst}$ is also the list of mode states of s. Since $s \in DCnf$, all symbols in s are from the correct alphabet for $\mathcal{A}_{(c,\emptyset,mdst_\perp)}$ and from the fact that $\oplus_{i=0}^{len(\overrightarrow{mdst})-1} \overrightarrow{mdst}[i] = \top$ and $\overrightarrow{mdst}$ is the list of mode states in s, we get by the definition of the automaton $\mathcal{A}_{(c,\emptyset,mdst_\perp)}$ (Definition 4) that the word s results in the state $\top$ in the automaton $\mathcal{A}_{(c,\emptyset,mdst_\perp)}$. Since $\top$ is an accepting state, we have $s \in \mathcal{L}(\mathcal{A}_{(c,\emptyset,mdst_\perp)})$. ◻

Finally, we can proof the soundness of the DPN analysis (Theorem 1).

Proof. We prove Theorem 1 by contradiction. Let $c \in \mathit{Com}$ be arbitrary such that $(c, \emptyset, \mathit{mdst}_\perp)\# \notin \mathit{pre}^*(\mathcal{L}(\mathcal{A}_{(c,\emptyset,\mathit{mdst}_\perp)}))$ and c does not use modes globally sound.

From the fact that c does not use modes globally sound, we get by the definition of globally sound use of modes and the definition of *gReach* that there is a global configuration $\mathit{gcnf} \in \mathit{GCnf}$ such that $\mathit{gcnf} \in \mathit{gReach}(\langle(c, \emptyset, \mathit{mdst}_\perp), \mathit{mem}\rangle)$ and the list of mode states $\overrightarrow{\mathit{mdst}}$ in the global configuration violates at least one of the following conditions:

C1: $\forall x, i, j.i \neq j \wedge x \in \overrightarrow{\mathit{mdst}}[i](\mathsf{A\text{-}NR}) \Longrightarrow x \in \overrightarrow{\mathit{mdst}}[j](\mathsf{G\text{-}NR})$, or
C2: $\forall x, i, j.i \neq j \wedge x \in \overrightarrow{\mathit{mdst}}[i](\mathsf{A\text{-}NW}) \Longrightarrow x \in \overrightarrow{\mathit{mdst}}[j](\mathsf{G\text{-}NW})$.

From Lemma 2 and Corollary 2 we obtain $s \in \mathit{DCnf}$ with $\mathit{toDPN}(\mathit{gcnf}) = s$, $s \in \mathcal{L}(\mathcal{A}_{(c,\emptyset,\mathit{mdst}_\perp)})$ and $(c, \emptyset, \mathit{mdst}_\perp)\# \rightarrowtriangle^* s$. This contradicts our initial assumption that $(c, \emptyset, \mathit{mdst}_\perp)\# \notin \mathit{pre}^*(\mathcal{L}(\mathcal{A}_{(c,\emptyset,\mathit{mdst}_\perp)}))$. □

A.2 Soundness of the Guarantee Inference

Lemma/Theorem	Depends on lemmas/theorems
Lemma 3	none
Lemma 4	none
Lemma 5	Lemma 4
Theorem 2	Lemma 3, Lemma 5

In the first step, we show that a command does not read and write the variables for which a mode state provides the noread and nowrite guarantees, if the mode state does not provide the guarantees for the inferred sets of variables.

Lemma 3. *If* $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c'\}\overline{x}_r, \overline{x}_w : c$, $\overline{x}'_r \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$, *and* $\overline{x}'_w \cap \mathit{mdst}(\mathsf{G\text{-}NW}) = \emptyset$ *then* c *provides its guarantees.*

Proof. We prove this lemma by structural induction on $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c\}\overline{x}_r, \overline{x}_w : c'$. We distinguish cases based on the last rule to derive this judgment.

Case (ISK): In this case, we have $c = \mathbf{skip}@\overrightarrow{a}$. From $c = \mathbf{skip}@\overrightarrow{a}$, we get by the rules AN1, SK that c does not read x holds for all $x \in \mathit{Var}$ and c does not write x holds for all $x \in \mathit{Var}$. Thus, c provides its guarantees.

Case (IAS): In this case, we have $c = x' := e@\overrightarrow{a}$. From the rule IAS we get that $\overline{x}'_r = \mathit{vars}(e)$ and $\overline{x}'_w = \{x'\}$. From $c = x' := e@\overrightarrow{a}$, $\overline{x}'_r \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$, $\overline{x}'_w \cap \mathit{mdst}(\mathsf{G\text{-}NW}) = \emptyset$, $\overline{x}'_r = \mathit{vars}(e)$, and $\overline{x}'_w = \{x'\}$, we get by the rules AN1 and AS that c' does not read x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NR})$ and c' does not write x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NW})$. Thus, c provides its guarantees.

Case (ILO): In this case, we have $c = \mathbf{lock}(l)@\overrightarrow{a}$. From $c = \mathbf{lock}(l)@\overrightarrow{a}$, we get by the rules AN1, and LK, that c does not read x for all $x \in \mathit{Var}$ and c does not write x for all $x \in \mathit{Var}$. Thus, c provides its guarantees.

Case (IUL): In this case, we have $c = \mathbf{unlock}(l)@\overrightarrow{a}$. From $c = \mathbf{unlock}(l)@\overrightarrow{a}$, we get by the rules AN1, and ULK, that c does not read x for all $x \in \mathit{Var}$ and c does not write x for all $x \in \mathit{Var}$. Thus, c provides its guarantees.

Case (ISP): In this case, we have $c = \mathbf{spawn}(c_s)@\overrightarrow{a}$ for some $c_s' \in \mathit{Com}$ and some $\overrightarrow{a} \in \mathit{Ann}^*$. From $c = \mathbf{spawn}(c_s)@\overrightarrow{a}$, we get by the rules AN1, and SP, that c does not read x for all $x \in \mathit{Var}$ and c does not write x for all $x \in \mathit{Var}$. Thus, c provides its guarantees.

Case (ISQ): In this case, we have $c = c_1; c_2$. From the rule ISQ, we get that $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o,1}\}\overline{x}_r'', \overline{x}_w'' : c_1$.
From $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o,1}\}\overline{x}_r'', \overline{x}_w'' : c_1$ and $\overline{x}_r' \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$ and $\overline{x}_w' \cap \mathit{mdst}(\mathsf{G\text{-}NW}) = \emptyset$, we get by the induction hypothesis that c_1 provides its guarantess.
From $c = c_1; c_2$ we get by the rules SQ1 and SQ2 and the fact that c_1 provides its guarantees that c provides its guarantees.

Case (IIF): In this case, we have $c = \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{fi}$. From the rule IIF we get that $\mathit{vars}(e) = \overline{x}_r'$.
From $c = \mathbf{if}\ e\ \mathbf{then}\ c_1'\ \mathbf{else}\ c_2'\ \mathbf{fi}$, $\mathit{vars}(e) = \overline{x}_r'$, and $\overline{x}_r' \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$, we get by the rules IFT, and IFF, that c does not read x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NR})$ and c does not write x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NW})$. Thus, c provides its guarantees.

Case (IWH): In this case, we have $c = \mathbf{while}\ e\ \mathbf{do}\ c_1\ \mathbf{od}@\overrightarrow{a}$. From the rule IWH, we get that $\mathit{vars}(e) = \overline{x}_r'$. From $c = \mathbf{while}\ e\ \mathbf{do}\ c_1\ \mathbf{od}@\overrightarrow{a}$, $\mathit{vars}(e) = \overline{x}_r'$, and $\overline{x}_r' \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$, we get by the rules AN2, WHT, AN1, and WHF, that c does not read x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NR})$ and c does not write x holds for all $x \in \mathit{mdst}(\mathsf{G\text{-}NW})$. Thus, c provides its guarantees.

Case (IAN): In this case, we have $c = c_1@\overrightarrow{a}$ with $\overrightarrow{a} = \overrightarrow{a} \restriction_{\mathsf{A}}$. From the rule IAN, we get that $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ is derivable.
From $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ and $\overline{x}_r' \cap \mathit{mdst}(\mathsf{G\text{-}NR}) = \emptyset$ and $\overline{x}_w' \cap \mathit{mdst}(\mathsf{G\text{-}NW}) = \emptyset$, we get by the induction hypothesis that c_1 provides its guarantees.
From $c = c_1@\overrightarrow{a}$ we get by the rules AN1 and AN2 and the fact that c_1 provides its guarantees. that c provides its guarantees.

Next we show that a command that results from an execution step of a command obtained with our guarantee inference again can be obtained with our guarantee inference and if the mode state and variable sets in the inference fit to each other before the step, they also fit to each other after the step.

Lemma 4. *If $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_o\}\overline{x}_r, \overline{x}_w : c$ is derivable, $\mathit{mdst}(\mathsf{G\text{-}NR}) \cap \overline{x}_r' = \emptyset$ and $\mathit{mdst}(\mathsf{G\text{-}NW}) \cap \overline{x}_w' = \emptyset$ hold, and $\langle c, \mathit{lkst}, \mathit{mdst}, \mathit{mem}\rangle \xrightarrow{\alpha} \langle c', \mathit{lkst}', \mathit{mdst}', \mathit{mem}\rangle$ is derivable, then either the following three conditions are satisfied:*

1. *$c' = \mathbf{stop}$ and*
2. *$\mathit{mdst}'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and*
3. *$\mathit{mdst}'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$*

or the following three conditions are satisfied:

1. *$\overline{x}' \vdash \overline{x}_r'', \overline{x}_w''\{c_o'\}\overline{x}_r, \overline{x}_w : c'$ and*
2. *$\mathit{mdst}'(\mathsf{G\text{-}NR}) \cap \overline{x}_r'' = \emptyset$ and*
3. *$\mathit{mdst}'(\mathsf{G\text{-}NW}) \cap \overline{x}_w'' = \emptyset$.*

Moreover, if $\alpha = \nearrow_{\langle c'', \emptyset, \mathit{mdst}_\perp\rangle}$, then $\emptyset \vdash \emptyset, \emptyset\{c_o''\}\emptyset, \emptyset : c''$.

Proof. We prove this lemma by structural induction on $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_o\}\overline{x}_r, \overline{x}_w : c$. We distinguish cases for the last rule used in the derivation of this judgment.

Case (ISK): In this case, we have $c = \mathbf{skip}@\overrightarrow{a}$ with
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$. From $c = \mathbf{skip}@\overrightarrow{a}$ and
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$, we get by the rules AN1, SK, and the definition of *updMds* that $c' = \mathbf{stop}$, $\alpha = \epsilon$, $mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup \overline{x}) \setminus \overline{x}_r$, and $mdst'(\mathsf{G\text{-}NW}) = mdst(\mathsf{G\text{-}NW}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \mathbf{stop}$ and $\alpha = \epsilon$ we can conclude this case.

Case (IAS): In this case, we have $c = x{:=}e@\overrightarrow{a}$ with
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, vars(e) \cup \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \{x\}), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$.
From $c = x{:=}e@\overrightarrow{a}$ and
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, vars(e) \cup \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \{x\}), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$ we get by the rules AN1, AS, and the definition of *updMds* that $c' = \mathbf{stop}$, $\alpha = \epsilon$, $mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup vars(e) \cup \overline{x}) \setminus \overline{x}_r$, and $mdst'(\mathsf{G\text{-}NW}) = (mdst(\mathsf{G\text{-}NW}) \cup \{x\}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$, and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \mathbf{stop}$ and $\alpha = \epsilon$ we can conclude this case.

Case (ILO): In this case, we have $c = \mathbf{lock}(l)@\overrightarrow{a}$ with
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$.
From $c = \mathbf{lock}(l)@\overrightarrow{a}$ and
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$, we get by the rules AN1, LK, and the definition of *updMds* that $c' = \mathbf{stop}$, $\alpha = l$, $mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup \overline{x}) \setminus \overline{x}_r$, and $mdst'(\mathsf{G\text{-}NW}) = mdst(\mathsf{G\text{-}NW}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$, and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \mathbf{stop}$ and $\alpha = l$ we can conclude this case.

Case (IUL): In this case, we have $c = \mathbf{unlock}(l)@\overrightarrow{a}$ with
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$.
From $c = \mathbf{unlock}(l)@\overrightarrow{a}$ and
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$, we get by the rules AN1, ULK, and the definition of *updMds* that $c' = \mathbf{stop}$, $\alpha = \neg l$, $mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup \overline{x}) \setminus \overline{x}_r$, and $mdst'(\mathsf{G\text{-}NW}) = mdst(\mathsf{G\text{-}NW}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$, and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \mathbf{stop}$ and $\alpha = \neg l$ we can conclude this case.

Case (ISP): In this case, we have $c = \mathbf{spawn}(c_s)@\overrightarrow{a}$ with
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$ and
$\emptyset \vdash \emptyset, \emptyset\{c_{o,s}\}\emptyset, \emptyset : c_s$.
From $c = \mathbf{spawn}(c_s)@\overrightarrow{a}$ and
$\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x}), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$, we get by the rules AN1, SP, and the definition of *updMds* that $c' = \mathbf{stop}$,
$\alpha = \nearrow_{\langle c_s, \emptyset, mdst_\perp \rangle}$, $mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup \overline{x}) \setminus \overline{x}_r$, and $mdst'(\mathsf{G\text{-}NW}) = mdst(\mathsf{G\text{-}NW}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$, and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \mathbf{stop}$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$ and $\emptyset \vdash \emptyset, \emptyset\{c_{o,s}\}\emptyset, \emptyset : c_s$ we can conclude this case.

Case (ISQ): In this case, we have $c = c_1; c_2$.
From $c = c_1; c_2$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_o\}\overline{x}_r, \overline{x}_w : c$, we get by the rule ISQ that $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o1}\}\overline{x}''_r, \overline{x}''_w : c_1$ and
$\overline{x}' \vdash \overline{x}''_r, \overline{x}''_w\{c_{o2}\}\overline{x}_r, \overline{x}_w : c_2$.
From $c = c_1; c_2$ we get that the last rule in the derivation of $\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem'\rangle$ must be either SQ1 or SQ2. We distinguish these two cases.

Case (SQ1): From the assumption of this case, we get by the rule SQ1 that $c' = c_1'; c_2$ and $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst', mem'\rangle$ and $c_1' \neq \textbf{stop}$.
From $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o1}\}\overline{x}_r'', \overline{x}_w'' : c_1$ and
$\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst', mem'\rangle$ and $mdst(\mathsf{G\text{-}NR}) \cap \overline{x}_r' = \emptyset$ and $mdst(\mathsf{G\text{-}NW}) \cap \overline{x}_w' = \emptyset$ and $c_1' \neq \textbf{stop}$, we get by the induciton hypothesis that
1. $\overline{x}'' \vdash \overline{x}_r''', \overline{x}_w'''\{c_{o1}'\}\overline{x}_r'', \overline{x}_w'' : c_1'$ and
2. $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r'' = \emptyset$ and
3. $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w'' = \emptyset$.

Moreover we get from the induction hypothesis, that if $\alpha = \nearrow_{\langle c'', \emptyset, mdst_\perp\rangle}$, then $\emptyset \vdash \emptyset, \emptyset\{c_o''\}\emptyset, \emptyset : c''$.
From $c' = c_1'; c_2$ and
$\overline{x}'' \vdash \overline{x}_r''', \overline{x}_w'''\{c_{o1}'\}\overline{x}_r'', \overline{x}_w'' : c_1'$ and $\overline{x}' \vdash \overline{x}_r'', \overline{x}_w''\{c_{o2}\}\overline{x}_r, \overline{x}_w : c_2$ we get by the rule ISQ that $\overline{x}'' \vdash \overline{x}_r''', \overline{x}_w'''\{c_o'\}\overline{x}_r'', \overline{x}_w'' : c'$.

Case (SQ2): From the assumption of this case, we get by the rule SQ2 that $c' = c_2$ and $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst', mem'\rangle$ and $c_1' = \textbf{stop}$.
From $\overline{x} \vdash \overline{x}_r', \overline{x}_w'\{c_{o1}\}\overline{x}_r'', \overline{x}_w'' : c_1$ and
$\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst', mem'\rangle$ and $mdst(\mathsf{G\text{-}NR}) \cap \overline{x}_r' = \emptyset$ and $mdst(\mathsf{G\text{-}NW}) \cap \overline{x}_w' = \emptyset$ and $c_1' = \textbf{stop}$, we get by the induction hypothesis that
1. $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r'' = \emptyset$ and
2. $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w'' = \emptyset$.

Moreover we get from the induction hypothesis, that if $\alpha = \nearrow_{\langle c'', \emptyset, mdst_\perp\rangle}$, then $\emptyset \vdash \emptyset, \emptyset\{c_o''\}\emptyset, \emptyset : c''$.
Since, $\overline{x}' \vdash \overline{x}_r'', \overline{x}_w''\{c_{o2}\}\overline{x}_r, \overline{x}_w : c_2$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r'' = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w'' = \emptyset$ we can conclude this case.

Case (IIF): In this case, we have $c = \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}$ and $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ and $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,2}\}\overline{x}_r, \overline{x}_w : c_2$.
From $c = \textbf{if } e \textbf{ then } c_1 \textbf{ else } c_2 \textbf{ fi}$ we get by the rules IFT and IFF that $c' = c_1$ or $c' = c_2$ and, furthermore, $mdst' = mdst$ and $\alpha = \epsilon$ or $\alpha = \epsilon$
Since $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$
and $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,2}\}\overline{x}_r, \overline{x}_w : c_2$ and $mdst'(\mathsf{G\text{-}NR}) \cap \emptyset = \emptyset$, and $mdst'(\mathsf{G\text{-}NW}) \cap \emptyset = \emptyset$, we can conclude this case.

Case (IWH): In this case, we have $c = \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$ and $\overrightarrow{a} = [\mathsf{acq}(\mathsf{G\text{-}NR}, \overline{x} \cup vars(e)), \mathsf{acq}(\mathsf{G\text{-}NW}, \emptyset), \mathsf{rel}(\mathsf{G\text{-}NR}, \overline{x}_r), \mathsf{rel}(\mathsf{G\text{-}NW}, \overline{x}_w)]$ and $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,1}\}vars(e), \emptyset : c_1$ and $\overline{x} \vdash vars(e), \emptyset\{c_o\}\overline{x}_r, \overline{x}_w : \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$.
From $c = \textbf{while } e \textbf{ do } c_1 \textbf{ od}$ we get by the rules WHF, WHT, AN1 and AN2 that either $c' = \textbf{stop}$ and $\alpha = \epsilon$ or $c' = c_1; \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$ and $\alpha = \epsilon$. We distinguish the two cases.

Case ($c' = \textbf{stop}$): From the assumption of this case we get by the rules WHF and AN1 and the definition of *updMds* that
$mdst'(\mathsf{G\text{-}NR}) = (mdst(\mathsf{G\text{-}NR}) \cup \overline{x} \cup vars(e)) \setminus \overline{x}_r$ and $mdst'(\mathsf{G\text{-}NW}) = mdst(\mathsf{G\text{-}NW}) \setminus \overline{x}_w$. Hence, $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.
Since $c' = \textbf{stop}$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$, we can conclude this case.

Case ($c' = c_1; \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$): From the assumption of this case we get by the rules WHT and AN2 that $mdst' = mdst$.
From $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{\textbf{skip}; c_{o,1}\}vars(e), \emptyset : c_1$ and
$\overline{x} \vdash vars(e), \emptyset\{c_o\}\overline{x}_r, \overline{x}_w : \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$ we get by the rule ISQ that $\overline{x} \cup vars(e) \vdash \emptyset, \emptyset\{c_o\}\overline{x}_r, \overline{x}_w : c_1; \textbf{while } e \textbf{ do } c_1 \textbf{ od@}\overrightarrow{a}$.
Since $mdst(\mathsf{G\text{-}NR}) \cap \emptyset = \emptyset$ and $mdst(\mathsf{G\text{-}NW}) \cap \emptyset = \emptyset$, we can conclude this case.

Case (IAN): In this case, we have $c = c_1@\overrightarrow{a}$ and $\overrightarrow{a} = \overrightarrow{a}' \restriction_{\mathsf{A}}$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$.

From $c = c_1@\overrightarrow{a}$ we get by the rule in the derivation of $\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem'\rangle$ must be either AN1 or AN2. We distinguish these two cases.

Case (AN1): In this case, we get from the rule AN1 that $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst', mdst'', mem'\rangle$ and $c' = \mathbf{stop}$ and $mdst' = updMds(mdst'', \overrightarrow{a})$.

From $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst', mdst'', mem'\rangle$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ and $mdst(\mathsf{G\text{-}NR}) \cap \overline{x}'_r = \emptyset$ and $mdst(\mathsf{G\text{-}NW}) \cap \overline{x}'_w = \emptyset$ and $c' = \mathbf{stop}$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ we get by the induction hypothesis that $mdst''(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst''(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.

From $mdst' = updMds(mdst'', \overrightarrow{a})$ and $\overrightarrow{a} = \overrightarrow{a}' \restriction_{\mathsf{A}}$ we get by definition of $updMds$ and $\restriction_{\mathsf{A}}$ that $mdst''(\mathsf{G\text{-}NR}) = mdst'(\mathsf{G\text{-}NR})$ and $mdst''(\mathsf{G\text{-}NW}) = mdst'(\mathsf{G\text{-}NW})$. Hence, due to $mdst''(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst''(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$ we have $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$.

Moreover we get from the induction hypothesis, that if $\alpha = \nearrow_{\langle c'', \emptyset, mdst_\perp\rangle}$, then $\emptyset \vdash \emptyset, \emptyset\{c''_o\}\emptyset, \emptyset : c''$.

Since $c' = \mathbf{stop}$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}_w = \emptyset$ we can conclude this case.

Case (AN2): In this case, we get from the rule AN2 that $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c'_1, lkst', mdst', mem'\rangle$ and $c'_1 \neq \mathbf{stop}$ and $c' = c'_1@\overrightarrow{a}$.

From $\langle c_1, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c'_1, lkst', mdst', mem'\rangle$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ and $mdst(\mathsf{G\text{-}NR}) \cap \overline{x}'_r = \emptyset$ and $mdst(\mathsf{G\text{-}NW}) \cap \overline{x}'_w = \emptyset$ and $c' \neq \mathbf{stop}$ and $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c_{o,1}\}\overline{x}_r, \overline{x}_w : c_1$ we get by the induction hypothesis that $\overline{x}' \vdash \overline{x}''_r, \overline{x}''_w\{c'_{o,1}\}\overline{x}_r, \overline{x}_w : c'_1$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}''_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}''_w = \emptyset$.

Moreover we get from the induction hypothesis, that if $\alpha = \nearrow_{\langle c'', \emptyset, mdst_\perp\rangle}$, then $\emptyset \vdash \emptyset, \emptyset\{c''_o\}\emptyset, \emptyset : c''$.

From $\overline{x} \vdash \overline{x}''_r, \overline{x}''_w\{c'_{o,1}\}\overline{x}_r, \overline{x}_w : c'_1$ and $\overrightarrow{a} = \overrightarrow{a}' \restriction_{\mathsf{A}}$ we get that $\overline{x} \vdash \overline{x}''_r, \overline{x}''_w\{c'_{o,1}@\overrightarrow{a}'\}\overline{x}_r, \overline{x}_w : c'_1@\overrightarrow{a}$.

Since $\overline{x} \vdash \overline{x}''_r, \overline{x}''_w\{c'_{o,1}@\overrightarrow{a}'\}\overline{x}_r, \overline{x}_w : c'_1@\overrightarrow{a}$ and $mdst'(\mathsf{G\text{-}NR}) \cap \overline{x}''_r = \emptyset$ and $mdst'(\mathsf{G\text{-}NW}) \cap \overline{x}''_w = \emptyset$, we can conclude this case.

□

Lemma 5. *Let* $gcnf = \langle[(c_1, lkst_1, mdst_1), \ldots, (c_n, lkst_n, mdst_n)], mem\rangle$ *be a global configuration such that for all* i *we have either* $c_i = \mathbf{stop}$ *or the following three conditions are satisfied:*

1. $\overline{x}_i \vdash \overline{x}'_{r,i}, \overline{x}'_{w,i}\{c_{o,i}\}\overline{x}_{r,i}, \overline{x}_{w,i} : c_i$, *and*
2. $mdst_i(\mathsf{G\text{-}NR}) \cap \overline{x}'_{r,i} = \emptyset$, *and*
3. $mdst_i(\mathsf{G\text{-}NW}) \cap \overline{x}'_{w,i} = \emptyset$.

If $gcnf \twoheadrightarrow \langle[(c'_1, lkst'_1, mdst'_1), \ldots, (c'_m, lkst'_m, mdst'_m)], mem'\rangle$, *then for all* i *we have either* $c'_i = \mathbf{stop}$ *or the following three conditions are satisfied:*

1. $\overline{x}'_i \vdash \overline{x}'''_{r,i}, \overline{x}'''_{w,i}\{c'_{o,i}\}\overline{x}''_{r,i}, \overline{x}''_{w,i} : c'_i$, *and*
2. $mdst'_i(\mathsf{G\text{-}NR}) \cap \overline{x}'''_{r,i} = \emptyset$, *and*
3. $mdst'_i(\mathsf{G\text{-}NW}) \cap \overline{x}'''_{w,i} = \emptyset$.

Proof. From the definition of the global transition system we know that either $m = n$ or $m = n + 1$ and there are j, j' with $j \leq n$ and $j' = n - j$ such that

- $c'_i = c_i$, $lkst'_i = lkst_i$, $mdst'_i = mdst_i$ for all $i < j$, and
- $c'_{m-i} = c_{n-i}$, $lkst'_{m-i} = lkst_{n-i}$, $mdst'_{m-i} = mdst_{n-i}$ for all $i < j'$.

For these control configurations, we get that the conclusion of the lemma holds directly from the assumptions of the lemma.

From the definition of the global transition system we further get that $\langle c_j, lkst_j, mdst_j, mem\rangle \xrightarrow{\alpha} \langle c'_{m-j'}, lkst'_{m-j'}, mdst'_{m-j'}, mem'\rangle$.

From $\overline{x}_j \vdash \overline{x}'_{r,j}, \overline{x}'_{w,j}\{c_{o,j}\}\overline{x}_{r,j}, \overline{x}_{w,j} : c_j$ and $\langle c_j, lkst_j, mdst_j, mem\rangle \xrightarrow{\alpha} \langle c'_{m-j'}, lkst'_{m-j'}, mdst'_{m-j'}, mem'\rangle$ and $mdst_j(\mathsf{G\text{-}NR}) \cap \overline{x}'_{r,j} = \emptyset$ and $mdst_j(\mathsf{G\text{-}NW}) \cap \overline{x}'_{w,j} = \emptyset$ we get by Lemma 4 that either $c'_{m-j'} = \mathbf{stop}$ or the following three conditions hold:

1. $\overline{x}_{m-j'} \vdash \overline{x}'_{r,m-j'}, \overline{x}'_{w,m-j'}\{c'_{o,m-j'}\}\overline{x}_{r,m-j'}, \overline{x}_{w,m-j'} : c'_{m-j'}$ and
2. $mdst_j(\mathsf{G\text{-}NR}) \cap \overline{x}'_{r,j} = \emptyset$ and
3. $mdst_j(\mathsf{G\text{-}NW}) \cap \overline{x}'_{w,j} = \emptyset$.

It remains to show that c'_j and $mdst'_j$ satisfy the conditions from the lemma. If $m = n$, the cases for j and $m - j'$ coincide. Hence assume $m = n + 1$. Then we get from the definition of the global transition system that $\alpha = \nearrow_{\langle c_s, \emptyset, mdst_\perp\rangle}$ and $c'_j = c_s$ and $mdst_j = mdst_\perp$. In this case, we additionally get from Lemma 4 that $\emptyset \vdash \emptyset, \emptyset\{c'_{o,j}\}\emptyset, \emptyset : c'_j$. Since $mdst_\perp(\mathsf{G\text{-}NR}) \cap \emptyset = \emptyset$ and $mdst_\perp(\mathsf{G\text{-}NW}) \cap \emptyset = \emptyset$, we can conclude the proof.

The following is the proof sketch for Theorem 2.

Proof. Let $c, c' \in Com$ be arbitrary such that $\emptyset \vdash \emptyset, \emptyset\{\mathbf{skip}; c'\}\emptyset, \emptyset : c$ is derivable.

We must proof that $\langle [c, \emptyset, mdst_\perp], mem\rangle$ ensures a locally sound use of modes for all $mem \in Mem$. According to the definition of "ensures a locally sound use of modes" this means we must show that $(c'', lkst'', mdst'')$ provides its guarantees holds for all $(c'', lkst'', mdst'') \in CCnf$, $\overrightarrow{ccnf}, \overrightarrow{ccnf}' \in CCnf^*$, and $mem'' \in Mem$ with $\langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle \in gReach(\langle [c, \emptyset, mdst_\perp], mem\rangle)$.

Let $(c'', lkst'', mdst'') \in CCnf$, $\overrightarrow{ccnf}, \overrightarrow{ccnf}' \in CCnf^*$, and $mem'' \in Mem$ be arbitrary such that $\langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle \in gReach(\langle [c, \emptyset, mdst], mem\rangle)$.
From $\langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle \in gReach(\langle [c, \emptyset, mdst], mem\rangle)$, we get by definition of global reachability that $\langle [c, \emptyset, mdst_\perp], mem\rangle \twoheadrightarrow^* \langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle$. Hence, we have either $\langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle = \langle [c, \emptyset, mdst_\perp], mem\rangle$ or there is k such that $\langle [c, \emptyset, mdst_\perp], mem\rangle \twoheadrightarrow_k \langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle$.

Assume $\langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle = \langle [c, \emptyset, mdst_\perp], mem\rangle$. Hence, $c'' = c$ and $mdst'' = mdst_\perp$. From $\emptyset, \emptyset\{\mathbf{skip}; c'\}\emptyset, \emptyset : c$ we get by the definition of the inference Figure 3 that $c = \mathbf{skip}@\overrightarrow{a}; c_B$ for some $\overrightarrow{a}$ and c_B. Hence, we get from the rules SQ1, AN1, and SK, that c does not read or write any variable and, consequently, c'' also does not read or write any variable. Thus, $((c, \emptyset, mdst_\perp), mem)$ provides its guarantees.

Now assume there is a k such that $\langle [c, \emptyset, mdst_\perp], mem\rangle \twoheadrightarrow_k \langle \overrightarrow{ccnf}{+}{+}[(c'', lkst'', mdst'')]{+}{+}\overrightarrow{ccnf}', mem''\rangle$. Since $\emptyset \vdash \emptyset, \emptyset\{\mathbf{skip}; c'\}\emptyset, \emptyset :$

c and $mdst_\perp(\mathsf{G\text{-}NR}) \cap \emptyset = \emptyset$ and $mdst_\perp(\mathsf{G\text{-}NW}) \cap \emptyset = \emptyset$ and the fact that Lemma 5 re-establishes its prerequisits for the resulting configuation after a global transition, we can apply Lemma 5 k times inductively to obtain that $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c'''\}\overline{x}_r, \overline{x}_w : c''$ and $mdst''(\mathsf{G\text{-}NR}) \cap \overline{x}'_r = \emptyset$ and $mdst''(\mathsf{G\text{-}NW}) \cap \overline{x}'_w = \emptyset$.

From $\overline{x} \vdash \overline{x}'_r, \overline{x}'_w\{c'''\}\overline{x}_r, \overline{x}_w : c''$ and $mdst''(\mathsf{G\text{-}NR}) \cap \overline{x}'_r = \emptyset$ and $mdst''(\mathsf{G\text{-}NW}) \cap \overline{x}'_w = \emptyset$ we get by Lemma 3 that $((c'', lkst'', mdst''), mem'')$ provides its guarantees. □

A.3 Soundness of the Security Type System

In this subsection, we introduce a bisimulation-based security property and prove the soundness of our security type system with respect to this bisimulation-based security property. The following table lists the dependencies between lemmas and theorems in this subsection.

Lemma/Theorem	Depends on lemmas/theorems
Lemma 6	none
Lemma 7	none
Lemma 8	Lemma 7
Lemma 9	none
Lemma 10	Lemma 6, Lemma 7, Lemma 8, Lemma 9
Lemma 11	Lemma 7, Lemma 8, Lemma 10
Lemma 12	none
Theorem 5	Lemma 11, Lemma 12

For the proofs, we extend the type system with an additional typing rule to avoid treating **stop** as special case. Since we add a typing rule, we actually relax the constraints of the type system and thus prove a stronger property that the soundness property of our type system. The additional typing rule is:

$$\textsc{tst}\ \frac{}{\vdash_{lev} \Lambda\{\mathbf{stop}\}\Lambda : \mathbf{stop}}$$

Now we show that whenever a command that is accepted by our type system is also accepted by our type system after lowering some security levels in the initial partial type environment and raising some security levels in the resulting partial type environment.

Lemma 6. *If $\vdash_{lev} \Lambda_1\{c\}\Lambda'_1 : c_s$ is derivable, then $\vdash_{lev} \Lambda_2\{c\}\Lambda'_2 : c_s$ is derivable for all Λ_2 and Λ'_2 with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda'_1 \sqsubseteq \Lambda'_2$.*

Proof. We proof this lemma by induction on $\vdash_{lev} \Lambda_1\{c\}\Lambda'_1 : c_s$. We distinguish the cases for the last rule used in the derivation of this judgment.

Case (TSK): We get by the rule TSK that $c = \mathbf{skip}$ and $c_s = \mathbf{skip}$ and $\Lambda_1 \sqsubseteq \Lambda'_1$. Let Λ_2 and Λ'_2 be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda'_1 \sqsubseteq \Lambda'_2$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda'_1$, and $\Lambda'_1 \sqsubseteq \Lambda'_2$, we get that $\Lambda'_2 \sqsubseteq \Lambda'_2$
Hence, we get from the rule TSK that $\vdash_{lev} \Lambda_2\{c\}\Lambda'_2 : c_s$ is derivable.

Case (TAL): We get by the rule TAL that $c = x{:=}e$ and $c_s = x{:=}e$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $lev(x) = \mathbf{low}$ and $x \notin pre(\Lambda_1)$ and $\Lambda_1 \sqsubseteq \Lambda'_1$. Let Λ_2 and Λ'_2 be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda'_1 \sqsubseteq \Lambda'_2$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ we get that $\vdash_{lev,\Lambda_2} e : \mathbf{low}$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda'_1$, and $\Lambda'_1 \sqsubseteq \Lambda'_2$, we get that $\Lambda'_2 \sqsubseteq \Lambda'_2$.
Hence, we get from the rule TAL that $\vdash_{lev} \Lambda_2\{c\}\Lambda'_2 : c_s$ is derivable.

Case (TAH): We get by the rule TAH that $c = x{:=}e$ and $c_s = \mathbf{skip}$ and $lev(x) = \mathbf{high}$ and $x \notin pre(\Lambda_1)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$, we get that $\Lambda_2' \sqsubseteq \Lambda_2'$.
Hence, we get from the rule TAH that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TFL): We get by the rule TFL that $c = x{:=}e$ and $c_s = x{:=}e$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ we get that $\vdash_{lev,\Lambda_2} e : \mathbf{low}$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ we get that $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1[x \mapsto \mathbf{low}]$. From $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1[x \mapsto \mathbf{low}]$, $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$ we get that $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_2'$.
Hence, we get from the rule TFL that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TFH): We get by the rule TFH that $c = x{:=}e$ and $c_s = \mathbf{skip}$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ we get that $\Lambda_2[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1[x \mapsto \mathbf{high}]$. From $\Lambda_2[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1[x \mapsto \mathbf{high}]$, $\Lambda_1[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$, we get that $\Lambda_2[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_2'$.
Hence, we get from the rule TFH that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TLO): We get by the rule TLO that $c = \mathbf{lock}(l)$ and $c_s = \mathbf{lock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$, we get that $\Lambda_2' \sqsubseteq \Lambda_2'$
Hence, we get from the rule TLO that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TUL): We get by the rule TUL that $c = \mathbf{unlock}(l)$ and $c_s = \mathbf{unlock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$, we get that $\Lambda_2' \sqsubseteq \Lambda_2'$
Hence, we get from the rule TUL that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TSP): We get by the rule TSP that $c = \mathbf{spawn}(c_1)$ and $c_s = \mathbf{spawn}(c_{s1})$ and $\vdash_{lev} c_1 : c_{s1}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$, $\Lambda_1 \sqsubseteq \Lambda_1'$, and $\Lambda_1' \sqsubseteq \Lambda_2'$, we get that $\Lambda_2' \sqsubseteq \Lambda_2'$
Hence, we get from the rule TSP that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TIH): We get by the rule TIH that $c = \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{fi}$ and $c_s = \mathbf{skip}; c_{s1}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ and $\vdash_{lev} \Lambda_1\{c_2\}\Lambda_1' : c_{s2}$ and $c_{s1} = c_{s2}$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ and $\vdash_{lev} \Lambda_1\{c_2\}\Lambda_1' : c_{s2}$ we get by the induction hypothesis that $\vdash_{lev} \Lambda_2\{c_1\}\Lambda_2' : c_{s1}$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$.
Hence, we get from the rule TIH that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TIL): We get by the rule TIL that $c = \mathbf{if}\ e\ \mathbf{then}\ c_1\ \mathbf{else}\ c_2\ \mathbf{fi}$ and $c_s = \mathbf{if}\ e\ \mathbf{then}\ c_{s1}\ \mathbf{else}\ c_{s2}\ \mathbf{fi}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ and $\vdash_{lev} \Lambda_1\{c_2\}\Lambda_2' : c_{s2}$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ and $\vdash_{lev} \Lambda_1\{c_2\}\Lambda_2' : c_{s2}$ we get by the induction hypothesis that $\vdash_{lev} \Lambda_2\{c_1\}\Lambda_2' : c_{s1}$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$.
From $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $\Lambda_2 \sqsubseteq \Lambda_1$ we get that $\vdash_{lev,\Lambda_2} e : \mathbf{low}$
Hence, we get from the rule TIL that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TWL): We get by the rule TWL that $c = \mathbf{while}\ e\ \mathbf{do}\ c_1\ \mathbf{od}$ and $c_s = \mathbf{while}\ e\ \mathbf{do}\ c_{s1}\ \mathbf{od}$ and $\vdash_{lev} \Lambda_1''\{c_1\}\Lambda_1'' : c_{s1}$ and $\vdash_{lev,\Lambda_1''} e : \mathbf{low}$ and $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_1'' \sqsubseteq \Lambda_1'$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_2 \sqsubseteq \Lambda_1$ we get that $\Lambda_2 \sqsubseteq \Lambda_1''$.
From $\Lambda_1'' \sqsubseteq \Lambda_1'$ and $\Lambda_1' \sqsubseteq \Lambda_2'$ we get that $\Lambda_1'' \sqsubseteq \Lambda_2'$.
Hence, we get from the rule TWL that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TSQ): We get by the rule TSQ that $c = c_1; c_2$ and $c_s = c_{s1}; c_{s2}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1'' : c_{s1}$ and $\vdash_{lev} \Lambda_1''\{c_2\}\Lambda_1' : c_{s2}$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1'' : c_{s1}$ we get by the induction hypothesis that $\vdash_{lev} \Lambda_2\{c_1\}\Lambda_1'' : c_{s1}$. From $\Lambda_1' \sqsubseteq \Lambda_2'$ and $\vdash_{lev} \Lambda_1''\{c_2\}\Lambda_1' : c_{s2}$. we get by the induction hypothesis that $\vdash_{lev} \Lambda_1''\{c_2\}\Lambda_2' : c_{s2}$.
Hence, we get from the rule TSQ that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

Case (TAN): We get by the rule TAN that $c = c_1@\overrightarrow{a}$ and $c_s = c_{s1}@\overrightarrow{a} \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1'' : c_{s1}$ and $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}$ and $\forall x.\Lambda_1''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_1'{}_{lev}\langle x\rangle$. Let Λ_2 and Λ_2' be arbitrary with $\Lambda_2 \sqsubseteq \Lambda_1$ and $\Lambda_1' \sqsubseteq \Lambda_2'$.
From $c = c_1@\overrightarrow{a}$ we get that that last rule in any derivation of $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ must be TAN. We now must show that there is Λ_2'' such that $\vdash_{lev} \Lambda_2\{c_1\}\Lambda_2'' : c_{s1}$, $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}$ and $\forall x.\Lambda_2''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_2'{}_{lev}\langle x\rangle$.
From $\Lambda_1' \sqsubseteq \Lambda_2'$ and $\forall x.\Lambda_1''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_1'{}_{lev}\langle x\rangle$ we get that $\forall x.\Lambda_1''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_2'{}_{lev}\langle x\rangle$.
From $\Lambda_2 \sqsubseteq \Lambda_1$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1'' : c_{s1}$ we get by the induction hypothesis that $\vdash_{lev} \Lambda_2\{c_1\}\Lambda_2'' : c_{s1}$ is derivable for all Λ_2'' with $\Lambda_1'' \sqsubseteq \Lambda_2''$. Since, $\forall x.\Lambda_1''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_2'{}_{lev}\langle x\rangle$ holds, this judgment is also derivable for Λ_2'' with $\Lambda_2'(x) = \Lambda_2''{}_{lev}\langle x\rangle$ for all $x \in pre(\Lambda_2')$ and $\Lambda_1''(x) = \Lambda_2''(x)$ for all $x \in pre(\Lambda_2'') \setminus pre(\Lambda_2')$. From $\Lambda_2'(x) = \Lambda_2''{}_{lev}\langle x\rangle$ for all $x \in pre(\Lambda_2')$, $pre(\Lambda_1'') = pre(\Lambda_2'')$, $pre(\Lambda_1') = pre(\Lambda_2')$, and $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}$, we get that $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}$.
From $\Lambda_2'(x) = \Lambda_2''{}_{lev}\langle x\rangle$ for all $x \in pre(\Lambda_2')$ and $\Lambda_1''(x) = \Lambda_2''(x)$ for all $x \in pre(\Lambda_2'') \setminus pre(\Lambda_2')$ and $\forall x.\Lambda_1''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_2'{}_{lev}\langle x\rangle$ we get that $\forall x.\Lambda_2''{}_{lev}\langle x\rangle \sqsubseteq \Lambda_2'{}_{lev}\langle x\rangle$.
Hence, we get from the rule TAN that $\vdash_{lev} \Lambda_2\{c\}\Lambda_2' : c_s$ is derivable.

□

We now define, when a partial type environment is compatible with a domain assignment and a mode state as follows.

Definition 6. *A partial type environment $\Lambda : Var \rightharpoonup Lev$, a mode state $mdst \in MdSt$, and a domain assignment lev are compatible, if and only if*

$$pre(\Lambda) = \left\{ x \in Var \,\middle|\, \begin{array}{l} (lev(x) = \mathbf{low} \wedge x \in mdst(\mathsf{A\text{-}NR})) \\ \vee (lev(x) = \mathbf{high} \wedge x \in mdst(\mathsf{A\text{-}NW})) \end{array} \right\}$$

We denote the set of mode states that is compatible with lev and Λ by $comp(lev, \Lambda)$.

Intuitively, a partial type environment is compatible with a domain assignment and a mode state, if it only tracks flow-sensitive levels for **low** variables for which a no-read assumption is made, and for **high** variables for which a no-write assumption is made.

We further define a notion of memory equivalence that relates exactly those memorys that refer to equal values for all variables that currently have the security level **low** as follows.

Definition 7. *Two memories $mem, mem' \in Mem$ are* **low***-equal wrt. a partial type environment Λ and a domain assignment lev (denoted by: $mem =^{lev,\Lambda}_{\mathbf{low}} mem'$), if and only if the following condition holds:*

- $\Lambda_{lev}\langle x\rangle = \mathbf{low} \implies mem(x) = mem'(x)$ *for all* $x \in Var$.

We now show that whenever the **low** slice of a command is a **skip**, then the memories before and after the execution step of the command are **low**-equal with respect to the partial type environment after the step and the domain assignment, and the command terminates in one step.

Lemma 7. *If* $\vdash_{lev} \Lambda\{c\}\Lambda' : \mathbf{skip}$ *is derivable, then*

$$\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst, mdst, mem'\rangle$$

is derivable with $mem =^{lev,\Lambda'}_{\mathbf{low}} mem'$ *and* $\Lambda \sqsubseteq \Lambda'$ *holds.*

Proof. The last rule in the derivation of $\vdash_{lev} \Lambda\{c\}\Lambda' : \mathbf{skip}$ must be either TSK, TAH, or TFH

We distinguish these three cases.

Case (TSK): We get by the rule TSK that $c = \mathbf{skip}$ and $\Lambda \sqsubseteq \Lambda'$ holds.
From $c = \mathbf{skip}$ we get by the rule SK that

$$\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst, mdst, mem'\rangle$$

is derivable with $mem' = mem$.
From $mem' = mem$ we get that $mem =^{lev,\Lambda'}_{\mathbf{low}} mem'$.

Case (TAH): We get by the rule TAH that $c = x{:=}e$ and $x \notin pre(\Lambda)$ and $lev(x) = \mathbf{high}$ and $\Lambda \sqsubseteq \Lambda'$ holds.
From $c = x{:=}e$ we get by the rule SK that

$$\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst, mdst, mem'\rangle$$

is derivable with $mem' = mem[x \mapsto v]$ for some $v \in Val$.
From $\Lambda \sqsubseteq \Lambda'$ and $x \notin pre(\Lambda)$ we get that $x \notin pre(\Lambda')$.
From $x \notin pre(\Lambda')$ and $lev(x) = \mathbf{high}$ we get $mem =^{lev,\Lambda'}_{\mathbf{low}} mem[x \mapsto v]$. Hence, $mem =^{lev,\Lambda'}_{\mathbf{low}} mem'$.

Case (TFH): We get by the rule TFH that $c = x{:=}e$ and $x \in pre(\Lambda)$ and $\Lambda[x \mapsto \mathbf{high}] \sqsubseteq \Lambda'$ holds.
From $c = x{:=}e$ we get by the rule SK that

$$\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle \mathbf{stop}, lkst, mdst, mem'\rangle$$

is derivable with $mem' = mem[x \mapsto v]$ for some $v \in Val$.
From $\Lambda[x \mapsto \mathbf{high}] \sqsubseteq \Lambda'$ and $Lev = \{\mathbf{low}, \mathbf{high}\}$ and $\mathbf{low} \sqsubseteq \mathbf{high}$ and $\mathbf{high} \sqsubseteq \mathbf{high}$ we get that $\Lambda \sqsubseteq \Lambda'$.
From $\Lambda[x \mapsto \mathbf{high}] \sqsubseteq \Lambda'$ and $Lev = \{\mathbf{low}, \mathbf{high}\}$ and $\mathbf{high} \not\sqsubseteq \mathbf{low}$ we get that $\Lambda'(x) = \mathbf{high}$. From $\Lambda'(x) = \mathbf{high}$ we get that $mem =^{lev,\Lambda'}_{\mathbf{low}} mem[x \mapsto v]$. Hence, $mem =^{lev,\Lambda'}_{\mathbf{low}} mem'$.

□

We now show that, whenever two commands have the same **low**-slice, and partial type environments in the beginning with identical pre-images, then the resulting partial type environments have the same pre-image.

Lemma 8. *If* $pre(\Lambda_1) = pre(\Lambda_2)$, $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_s$ *and* $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$, *then* $pre(\Lambda_1') = pre(\Lambda_2')$.

Proof. We proof this lemma by structural induction on the judgment $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_s$. We make a case distinction on the last rule in the derivation of $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_2 : c_s$.

Case (TSK): We get by the rule TSK that $c_s = \mathbf{skip}$.

From $c_s = \mathbf{skip}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_s$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ we get by Lemma 7 that $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$. From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TAL): We get by the rule TAL that $c_s = x{:=}e$ and $x \notin pre(\Lambda_1)$ and $lev(x) = \mathbf{low}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$

From $x \notin pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that $x \notin pre(\Lambda_2)$. From $c_s = x{:=}e$ and $x \notin pre(\Lambda_2)$ and $lev(x) = \mathbf{low}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TAL. From this rule we get $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TFL): We get by the rule TFL that $c_s = x{:=}e$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$.

From $x \in pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that $x \in pre(\Lambda_2)$. From $c_s = x{:=}e$ and $x \in pre(\Lambda_2)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TFL. From this rule we get $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_2'$.

From $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$ and $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1[x \mapsto \mathbf{low}]) = pre(\Lambda_1')$ and $pre(\Lambda_2[x \mapsto \mathbf{low}]) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1[x \mapsto \mathbf{low}]) = pre(\Lambda_1')$ and $pre(\Lambda_2[x \mapsto \mathbf{low}]) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TAH): We get by the rule TSK that $c_s = \mathbf{skip}$.

From $c_s = \mathbf{skip}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_s$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ we get by Lemma 7 that $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$. From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TFH): We get by the rule TSK that $c_s = \mathbf{skip}$.

From $c_s = \mathbf{skip}$ and $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_s$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ we get by Lemma 7 that $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$. From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TLO): We get by the rule TLO that $c_s = \mathbf{lock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$

From $c_s = \mathbf{lock}(l)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TLO. From this rule we get $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TUL): We get by the rule TLO that $c_s = \mathbf{unlock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$

From $c_s = \mathbf{unlock}(l)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TUL. From this rule we get $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$.

From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TSP): We get by the rule TSP that $c_s = \mathbf{spawn}(c_{sA})$ and $\Lambda_1 \sqsubseteq \Lambda_1'$

From $c_s = \mathbf{spawn}(c_{sA})$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TSP. From this rule we get $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$. From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TSQ): We get by the rule TSQ that $c_1 = c_A; c_B$ and $c_s = c_{sA}; c_{sB}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$.
From $c_s = c_{sA}; c_{sB}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be either TSQ or TIH. We distinguish these two cases.

Case (TSQ): In this case, we get from the rule TSQ that $c_2 = c_C; c_D$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sB}$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : c_{sA}$ we get by the induction hypothesis that $pre(\Lambda_1'') = pre(\Lambda_2'')$. From $pre(\Lambda_1'') = pre(\Lambda_2'')$ and $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$ and and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sB}$ we get by the induction hypothesis that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TIH): In this case, we get from the rule TIH that $c_2 = \textbf{if } e \textbf{ then } c_C \textbf{ else } c_D \textbf{ fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sB}$ and $c_{sA} = \textbf{skip}$.
From $c_{sA} = \textbf{skip}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ we get by Lemma 7 that $\Lambda_1 \sqsubseteq \Lambda_1''$. From $\Lambda_1 \sqsubseteq \Lambda_1''$ we get that $pre(\Lambda_1) = pre(\Lambda_1'')$.
From $pre(\Lambda_1) = pre(\Lambda_1'')$ and $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sB}$ we get by the induction hypothesis that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TIL): We get by the rule TIL that $c_1 = \textbf{if } e \textbf{ then } c_A \textbf{ else } c_B \textbf{ fi}$ and $c_s = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$.
From $c_s = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TIL. From this rule we get that $c_2 = \textbf{if } e \textbf{ then } c_C \textbf{ else } c_D \textbf{ fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ we get by the induction hypothesis that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TIH): We get by the rule TIH that $c_1 = \textbf{if } e \textbf{ then } c_A \textbf{ else } c_B \textbf{ fi}$ and $c_s = \textbf{skip}; c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$.
From $c_s = \textbf{skip}; c_{sA}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be either TIH or TSQ. We distinguish these two cases.

Case (TIH): In this case, we get by the rule TIH that $c_2 = \textbf{if } e \textbf{ then } c_C \textbf{ else } c_D \textbf{ fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ we get by the induction hypothesis that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TSQ): In this case, we get by the rule TSQ that $c_2 = c_C; c_D$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : \textbf{skip}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sA}$.
From $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : \textbf{skip}$ we get that $\Lambda_2 \sqsubseteq \Lambda_2''$. From $\Lambda_2 \sqsubseteq \Lambda_2''$ we get that $pre(\Lambda_2) = pre(\Lambda_2'')$. From $pre(\Lambda_2) = pre(\Lambda_2'')$ and $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sA}$ we get by the induction hypothesis that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TWL): We get by the rule TWL that $c_s = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$ and $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_1'' \sqsubseteq \Lambda_1'$. From $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_1'' \sqsubseteq \Lambda_1'$ we get that $pre(\Lambda_1) = pre(\Lambda_1')$.
From $c_s = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TWL. From this rule we get $\Lambda_2 \sqsubseteq \Lambda_2''$ and $\Lambda_2'' \sqsubseteq \Lambda_2'$. From $\Lambda_2 \sqsubseteq \Lambda_2''$ and $\Lambda_2'' \sqsubseteq \Lambda_2'$ we get that $pre(\Lambda_2) = pre(\Lambda_2')$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $pre(\Lambda_1) = pre(\Lambda_1')$ and $pre(\Lambda_2) = pre(\Lambda_2')$ we get that $pre(\Lambda_1') = pre(\Lambda_2')$.

Case (TAN): We get by the rule TAN that $c_1 = c_A @ \overrightarrow{a}_A$ and $c_s = c_{sA} @ \overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}_A$.
From $c_s = c_{sA} @ \overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_s$ must be TAN. From this rule we get that $c_2 = c_B @ \overrightarrow{a}_B$ and $\vdash_{lev} \Lambda_2\{c_B\}\Lambda_2'' : c_{sA}$ and $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}_B$ and $\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}} = \overrightarrow{a}_B \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_B\}\Lambda_2'' : c_{sA}$ we get by the induction hypothesis that $pre(\Lambda_1'') = pre(\Lambda_2'')$.
From $pre(\Lambda_1'') = pre(\Lambda_2'')$ and $\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}} = \overrightarrow{a}_B \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ and $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}_A$ and $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}_B$ we get by the definition of $\oplus_{lev}$ that $pre(\Lambda_1') = pre(\Lambda_2')$.
□

We define the least upper bound of two partial type environments by a point-wise least upper bound of all variables in the pre-image of the partial type environments.

Definition 8. *The least upper bound Λ of two partial type environments Λ' and Λ'' with $pre(\Lambda') = pre(\Lambda'')$ (denoted by: $\Lambda = \Lambda' \sqcup \Lambda''$) is defined by $pre(\Lambda) = pre(\Lambda')$ and $\Lambda(x) = \Lambda'(x) \sqcup \Lambda''(x)$ for all $x \in pre(\Lambda)$.*

We show that whenever a command has **skip** as **low**-slice and is accepted by our type system for an initial and final partial type environment, then it is also accepted by our type system when using the least upper bound of the initial and final partial type environments with an arbitrary partial type environment.

Lemma 9. *If $\vdash_{lev} \Lambda_1\{c\}\Lambda_1' : \mathbf{skip}$ and $pre(\Lambda_1) = pre(\Lambda_2)$,*
then $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c\}(\Lambda_1' \sqcup \Lambda_2) : \mathbf{skip}$.

Proof. The last rule in the derivation of $\vdash_{lev} \Lambda_1\{c\}\Lambda_1' : \mathbf{skip}$ must be either TSK, TAH, or TFH.

We distinguish these three cases.

Case (TSK): We get by the rule TSK that $c = \mathbf{skip}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$ holds.
From $\Lambda_1 \sqsubseteq \Lambda_1'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$.
From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$ we get by the rule TSK that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c\}(\Lambda_1' \sqcup \Lambda_2) : c_s$.

Case (TAH): We get by the rule TAH that $c = x{:=}e$ and $x \notin pre(\Lambda_1)$ and $lev(x) = \mathbf{high}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$ holds.
From $\Lambda_1 \sqsubseteq \Lambda_1'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$.
From $x \notin pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that $x \notin pre(\Lambda_1 \sqcup \Lambda_2)$.
From $x \notin pre(\Lambda_1 \sqcup \Lambda_2)$ and $lev(x) = \mathbf{high}$ and $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$ we get by the rule TAH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c\}(\Lambda_1' \sqcup \Lambda_2) : c_s$.

Case (TFH): We get by the rule TFH that $c = x{:=}e$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1'$ holds.
From $\Lambda_1[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1'$ and $Lev = \{\mathbf{low}, \mathbf{high}\}$ and $\mathbf{high} \not\sqsubseteq \mathbf{low}$ we get that $\Lambda_1'(x) = \mathbf{high}$. From $\Lambda_1'(x) = \mathbf{high}$ we get that $(\Lambda_1' \sqcup \Lambda_2)(x) = \mathbf{high}$. From $\Lambda_1[x \mapsto \mathbf{high}] \sqsubseteq \Lambda_1'$ and $Lev = \{\mathbf{low}, \mathbf{high}\}$ and $\mathbf{low} \sqsubseteq \mathbf{high}$ and $\mathbf{high} \sqsubseteq \mathbf{high}$ we get that $\Lambda_1 \sqsubseteq \Lambda_1'$. From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$. From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$ and $(\Lambda_1' \sqcup \Lambda_2)(x) = \mathbf{high}$ and $\mathbf{high} \sqsubseteq \mathbf{high}$ we get that $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \mathbf{high}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$.
From $x \in pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that $x \in pre(\Lambda_1 \sqcup \Lambda_2)$.
From $x \in pre(\Lambda_1 \sqcup \Lambda_2)$ and $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \mathbf{high}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2)$ we get by the rule TFH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c\}(\Lambda_1' \sqcup \Lambda_2) : c_s$.

□

Now we show that whenever two commands with identical **low**-slices are accepted by our type system for some, possibly different initial and final partial type environments, then the first command is also accepted by our type system when using the least upper bounds of the respective partial type environments as initial and final partial type environments. Note that the conclusion the lemma establishes also holds for the second command, because all premises are symmetric and, hence, one could simply switch the two typing judgments.

Lemma 10. *If* $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ *and* $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ *and* $c_{s1} = c_{s2}$ *and* $pre(\Lambda_1) = pre(\Lambda_2)$ *and* $pre(\Lambda_1') = pre(\Lambda_2')$*, then* $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Proof. We prove this lemma by structural induction on $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ and $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$. To this end, we distinguish cases based on the last rule applied in the derivation of $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$.

Case (TSK): We get by the rule TSK that $c_1 = c_{s1} = \textbf{skip}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{skip}$ we get $c_{s2} = \textbf{skip}$. From $c_{s2} = \textbf{skip}$ we get by Lemma 7 that $\Lambda_2 \sqsubseteq \Lambda_2'$.
From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.
From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ we get by the rule TSK that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TAH): We get by the rule TAH that $c_1 = x{:=}e$ and $c_{s1} = \textbf{skip}$ and $x \notin pre(\Lambda_1)$ and $lev(x) = \textbf{high}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{skip}$ we get $c_{s2} = \textbf{skip}$. From $c_{s2} = \textbf{skip}$ we get by Lemma 7 that $\Lambda_2 \sqsubseteq \Lambda_2'$.
From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.
From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $x \notin pre(\Lambda_1)$ and $lev(x) = \textbf{high}$ we get by the rule TAH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TFH): We get by the rule TFH that $c_1 = x{:=}e$ and $c_{s1} = \textbf{skip}$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \textbf{high}] \sqsubseteq \Lambda_1'$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{skip}$ we get $c_{s2} = \textbf{skip}$. From $c_{s2} = \textbf{skip}$ we get by Lemma 7 that $\Lambda_2 \sqsubseteq \Lambda_2'$.
From $\Lambda_1[x \mapsto \textbf{high}] \sqsubseteq \Lambda_1'$ we get $\Lambda_1 \sqsubseteq \Lambda_1'$. From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$. From $\Lambda_1[x \mapsto \textbf{high}] \sqsubseteq \Lambda_1'$ we get that $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \textbf{high}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $x \in pre(\Lambda_1)$ we get that $x \in pre((\Lambda_1 \sqcup \Lambda_2))$.
From $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \textbf{high}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $x \in pre((\Lambda_1 \sqcup \Lambda_2))$ we get by the rule TFH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TAL): We get by the rule TAL that $c_1 = x{:=}e$ and $c_{s1} = x{:=}e$ and $\vdash_{lev,\Lambda_1} e : \textbf{low}$ and $x \notin pre(\Lambda_1)$ and $lev(x) = \textbf{low}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.
From $c_{s1} = c_{s2}$ and $c_{s1} = x{:=}e$ we get $c_{s2} = x{:=}e$. From $c_{s2} = x{:=}e$ and $x \notin pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TAL. From this rule we get that $c_2 = x{:=}e$ and $c_{s2} = x{:=}e$ and $\vdash_{lev,\Lambda_2} e : \textbf{low}$ and $\Lambda_2 \sqsubseteq \Lambda_2'$.
From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.
From $\vdash_{lev,\Lambda_1} e : \textbf{low}$ and $\vdash_{lev,\Lambda_2} e : \textbf{low}$ we get that $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \textbf{low}$.
From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \textbf{low}$ and $x \notin pre((\Lambda_1 \sqcup \Lambda_2)$ and $lev(x) = \textbf{low}$ we get by the rule TAL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TFL): We get by the rule TFL that $c_1 = x{:=}e$ and $c_{s1} = x{:=}e$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $x \in pre(\Lambda_1)$ and $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$.

From $c_{s1} = c_{s2}$ and $c_{s1} = x{:=}e$ we get $c_{s2} = x{:=}e$. From $c_{s2} = x{:=}e$ and $x \in pre(\Lambda_1)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TFL. From this rule we get that $c_2 = x{:=}e$ and $c_{s2} = x{:=}e$ and $\vdash_{lev,\Lambda_2} e : \mathbf{low}$ and $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_2'$.

From $\Lambda_1[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_1'$ and $\Lambda_2[x \mapsto \mathbf{low}] \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \mathbf{low}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.

From $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $\vdash_{lev,\Lambda_2} e : \mathbf{low}$ we get that $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \mathbf{low}$.

From $(\Lambda_1 \sqcup \Lambda_2)[x \mapsto \mathbf{low}] \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \mathbf{low}$ and $x \in pre((\Lambda_1 \sqcup \Lambda_2)$ we get by the rule TFL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TLO): We get by the rule TLO that $c_1 = \mathbf{lock}(l)$ and $c_{s1} = \mathbf{lock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.

From $c_{s1} = c_{s2}$ and $c_{s1} = \mathbf{lock}(l)$ we get $c_{s2} = \mathbf{lock}(l)$. From $c_{s2} = \mathbf{lock}(l)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TLO. From this rule we get $c_2 = \mathbf{lock}(l)$ and $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.

From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ we get by the rule TLO that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TUL): We get by the rule TUL that $c_1 = \mathbf{unlock}(l)$ and $c_{s1} = \mathbf{unlock}(l)$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.

From $c_{s1} = c_{s2}$ and $c_{s1} = \mathbf{unlock}(l)$ we get $c_{s2} = \mathbf{unlock}(l)$. From $c_{s2} = \mathbf{unlock}(l)$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TUL. From this rule we get $c_2 = \mathbf{unlock}(l)$ and $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.

From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ we get by the rule TUL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TSP): We get by the rule TSP that $c_1 = \mathbf{spawn}(c_A)$ and $c_{s1} = \mathbf{spawn}(c_{sA})$ and $\vdash_{lev} c_A : c_{sA}$ and $\Lambda_1 \sqsubseteq \Lambda_1'$.

From $c_{s1} = c_{s2}$ and $c_{s1} = \mathbf{spawn}(c_{sA})$ we get $c_{s2} = \mathbf{spawn}(c_{sA})$. From $c_{s2} = \mathbf{spawn}(c_{sA})$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TSP. From this rule we get $c_2 = \mathbf{spawn}(c_B)$ and $\vdash_{lev} c_B : c_{sA}$ and $\Lambda_2 \sqsubseteq \Lambda_2'$.

From $\Lambda_1 \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.

From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $\vdash_{lev} c_A : c_{sA}$ we get by the rule TSP that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TSQ): We get by the rule TSQ that $c_1 = c_A; c_B$ and $c_{s1} = c_{sA}; c_{sB}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$.

From $c_{s2} = c_{s1}$ and $c_{s1} = c_{sA}; c_{sB}$ we get that $c_{s2} = c_{sA}; c_{sB}$. From $c_{s2} = c_{sA}; c_{sB}$ we get that there are only two rules that can be applied last in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$, namely TSQ and TIH. We make a case distinction based on these two possibilities.

- Case (TSQ): In this case, we get by the rule TSQ that $c_2 = c_C; c_D$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sB}$.

  From $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sB}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$ and $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2'')\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$.

  From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$ and $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2'')\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$ we get by the rule TSQ that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.
- Case (TIH): In this case, we get by the rule TIH that $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_C\ \mathbf{else}\ c_D\ \mathbf{fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sB}$ and $c_{sA} = \mathbf{skip}$.

From $\vdash_{lev} \Lambda_1''\{c_B\}\Lambda_1' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sB}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$.
From $c_{sA} = \textbf{skip}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ we get by Lemma 9 that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2) : c_{sA}$.
From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2) : c_{sA}$ and $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$ we get by the rule TSQ that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TIL): We get by the rule TIL that $c_1 = \textbf{if } e \textbf{ then } c_A \textbf{ else } c_B \textbf{ fi}$ and $c_{s1} = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_B\}\Lambda_1' : c_{sB}$ and $\vdash_{lev,\Lambda_1} e : \textbf{low}$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$ we get $c_{s2} = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$.
From $c_{s2} = \textbf{if } e \textbf{ then } c_{sA} \textbf{ else } c_{sB} \textbf{ fi}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TIL. From this rule we get $c_2 = \textbf{if } e \textbf{ then } c_C \textbf{ else } c_D \textbf{ fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sB}$ and $\vdash_{lev,\Lambda_2} e : \textbf{low}$.
From $\vdash_{lev,\Lambda_1} e : \textbf{low}$ and $\vdash_{lev,\Lambda_2} e : \textbf{low}$ we get that $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \textbf{low}$.
From $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_B\}\Lambda_1' : c_{sB}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sB}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$.
From $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \textbf{low}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sB}$ we get by the rule TIL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TIH): We get by the rule TIL that $c_1 = \textbf{if } e \textbf{ then } c_A \textbf{ else } c_B \textbf{ fi}$ and $c_{s1} = \textbf{skip}; c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_B\}\Lambda_1' : c_{sA}$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{skip}; c_{sA}$ we get that $c_{s2} = \textbf{skip}; c_{sA}$. From $c_{s2} = \textbf{skip}; c_{sA}$ we get that there are only two rules that can be applied last in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$, namely TSQ and TIH. We make a case distinction based on these two possibilities.

Case (TIH): In this case, we get by the rule TIH that $c_2 = \textbf{if } e \textbf{ then } c_C \textbf{ else } c_D \textbf{ fi}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sA}$.
From $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_B\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_D\}\Lambda_2' : c_{sA}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$.
From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ we get by the rule TIH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TSQ): In this case, we get by the rule TSQ that $c_2 = c_C; c_D$ and $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : \textbf{skip}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sA}$.
From $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_1\{c_B\}\Lambda_1' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_D\}\Lambda_2' : c_{sA}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2'')\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2'')\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$.
From $\vdash_{lev} \Lambda_2\{c_C\}\Lambda_2'' : \textbf{skip}$ we get by Lemma 7 that $\Lambda_2 \sqsubseteq \Lambda_2''$. From $\Lambda_1 \sqsubseteq \Lambda_1$ and $\Lambda_2 \sqsubseteq \Lambda_2''$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1 \sqcup \Lambda_2'')$. From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1 \sqcup \Lambda_2'')$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2'')\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2'')\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ we get by Lemma 6 that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$.
From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_B\}(\Lambda_1' \sqcup \Lambda_2') : c_{sA}$ we get by the rule TIH that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TWL): We get by the rule TWL that $c_1 = \textbf{while } e \textbf{ do } c_A \textbf{ od}$ and $c_{s1} = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$ and $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_1'' \sqsubseteq \Lambda_1'$ and $\vdash_{lev,\Lambda_1''} e : \textbf{low}$ and $\vdash_{lev} \Lambda_1''\{c_A\}\Lambda_1'' : c_{sA}$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$ we get that $c_{s2} = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$.
From $c_{s2} = \textbf{while } e \textbf{ do } c_{sA} \textbf{ od}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TWL. From this rule we get that $c_2 = \textbf{while } e \textbf{ do } c_B \textbf{ od}$ and $\Lambda_2 \sqsubseteq \Lambda_2''$ and $\Lambda_2'' \sqsubseteq \Lambda_2'$ and $\vdash_{lev,\Lambda_2''} e : \textbf{low}$ and $\vdash_{lev} \Lambda_2''\{c_B\}\Lambda_2'' : c_{sA}$.

From $\vdash_{lev,\Lambda_1''} e : \mathbf{low}$ and $\vdash_{lev,\Lambda_2''} e : \mathbf{low}$ we get that $\vdash_{lev,(\Lambda_1''\sqcup\Lambda_2'')} e : \mathbf{low}$.
From $\vdash_{lev} \Lambda_1''\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_2''\{c_B\}\Lambda_2'' : c_{sA}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2'')\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$.
From $\Lambda_1 \sqsubseteq \Lambda_1''$ and $\Lambda_1'' \sqsubseteq \Lambda_1'$ and $\Lambda_2 \sqsubseteq \Lambda_2''$ and $\Lambda_2'' \sqsubseteq \Lambda_2'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1'' \sqcup \Lambda_2'')$ and $(\Lambda_1'' \sqcup \Lambda_2'') \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$.
From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1'' \sqcup \Lambda_2'')$ and $(\Lambda_1'' \sqcup \Lambda_2'') \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')$ and $\vdash_{lev} (\Lambda_1'' \sqcup \Lambda_2'')\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$ we get by the rule TWL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

Case (TAN): We get by the rule TAN that $c_1 = c_A@\overrightarrow{a}_A$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $c_{s1} = c_{sA}@\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ and $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}_A$ and $\forall x.\Lambda_{1\,lev}''\langle x\rangle \sqsubseteq \Lambda_{1\,lev}'\langle x\rangle$.
From $c_{s2} = c_{s1}$ and $c_{s1} = c_{sA}@\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ we get that $c_{s2} = c_{sA}@\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$.
From $c_{s2} = c_{sA}@\overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TAN. From this rule we get that $c_2 = c_B@\overrightarrow{a}_B$ and $\vdash_{lev} \Lambda_2\{c_B\}\Lambda_2'' : c_{sA}$ and $\overrightarrow{a}_B \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}} = \overrightarrow{a}_A \restriction_{\mathsf{A\text{-}NR,A\text{-}NW}}$ and $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}_B$ and $\forall x.\Lambda_{2\,lev}''\langle x\rangle \sqsubseteq \Lambda_{2\,lev}'\langle x\rangle$.
From $pre(\Lambda_1) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_B\}\Lambda_2'' : c_{sA}$ we get by Lemma 8 $pre(\Lambda_1'') = pre(\Lambda_2'')$. Hence, from $\vdash_{lev} \Lambda_1\{c_A\}\Lambda_1'' : c_{sA}$ and $\vdash_{lev} \Lambda_2\{c_B\}\Lambda_2'' : c_{sA}$ we get by the induction hypothesis that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$.
From $\forall x.\Lambda_{1\,lev}''\langle x\rangle \sqsubseteq \Lambda_{1\,lev}'\langle x\rangle$ and $\forall x.\Lambda_{2\,lev}''\langle x\rangle \sqsubseteq \Lambda_{2\,lev}'\langle x\rangle$ we get that $\forall x.(\Lambda_1'' \sqcup \Lambda_2'')_{lev}\langle x\rangle \sqsubseteq (\Lambda_1' \sqcup \Lambda_2')_{lev}\langle x\rangle$.
From $\Lambda_1' = \Lambda_1'' \oplus_{lev} \overrightarrow{a}_A$ and $\Lambda_2' = \Lambda_2'' \oplus_{lev} \overrightarrow{a}_B$ we get by definition of $\oplus_{lev}$ that $\forall x \in pre(\Lambda_1').\Lambda_1'(x) = \Lambda_{1\,lev}''\langle x\rangle$ and $\forall x \in pre(\Lambda_2').\Lambda_2'(x) = \Lambda_{2\,lev}''\langle x\rangle$. From $pre(\Lambda_2') = pre(\Lambda_1')$ and $\forall x \in pre(\Lambda_1').\Lambda_1'(x) = \Lambda_{1\,lev}''\langle x\rangle$ and $\forall x \in pre(\Lambda_2').\Lambda_2'(x) = \Lambda_{2\,lev}''\langle x\rangle$ we get that $\forall x \in pre((\Lambda_1' \sqcup \Lambda_2')).(\Lambda_1' \sqcup \Lambda_2') = (\Lambda_1'' \sqcup \Lambda_2'')_{lev}\langle x\rangle$. From $\forall x \in pre((\Lambda_1' \sqcup \Lambda_2')).(\Lambda_1' \sqcup \Lambda_2') = (\Lambda_1'' \sqcup \Lambda_2'')_{lev}\langle x\rangle$ we get by definition of $\oplus_{lev}$ that $(\Lambda_1' \sqcup \Lambda_2') = (\Lambda_1'' \sqcup \Lambda_2'') \oplus_{lev} \overrightarrow{a}_A$.
From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_A\}(\Lambda_1'' \sqcup \Lambda_2'') : c_{sA}$ and $\forall x.(\Lambda_1'' \sqcup \Lambda_2'')_{lev}\langle x\rangle \sqsubseteq \Lambda_1' \sqcup \Lambda_2')_{lev}\langle x\rangle$ and $(\Lambda_1' \sqcup \Lambda_2') = (\Lambda_1'' \sqcup \Lambda_2'') \oplus_{lev} \overrightarrow{a}_A$ we get by the rule TAN that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}(\Lambda_1' \sqcup \Lambda_2') : c_{s1}$.

□

We define an equality that relates all mode states that agree on all the assumptions made in these mode states.

Definition 9. *Two mode states $mdst, mdst' \in MdSt$ make equal assumptions (denoted by: $mdst =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst'$), if and only if $mdst(\mathsf{A\text{-}NR}) = mdst'(\mathsf{A\text{-}NR})$ and $mdst(\mathsf{A\text{-}NW}) = mdst'(\mathsf{A\text{-}NW})$*

We now show that, whenever two commands are typeable with the same partial type environments and have identical **low**-slices, then the fact that the first command can do a step in a given memory, then this implies that the second command can also do a step in any memory that is **low** equal with respect to a partial type environment, the resulting commands are again typable with identical partial type environments (type preservation), and the resulting memories are again **low**-equal with respect to a partial type environment.

Lemma 11. *If*

- $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}$ *and* $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ *with* $c_{s1} = c_{s2}$*, and*
- $mdst_1, mdst_2 \in comp(lev_1, \Lambda)$*, and*
- $mdst_1 =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst_2$*, and*

– $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$, *and*
– $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$

then there is $\alpha' \in Eve$, $mdst_2' \in MdSt$, $c_2', c_{s1}', c_{s2}' \in Com$, $mem_2' \in Mem$, *and* Λ'' *such that*

– $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : c_{s1}'$, *and* $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : c_{s2}'$, *with* $c_{s1}' = c_{s2}'$,
– $mdst_1', mdst_2' \in comp(lev, \Lambda'')$, *and*
– $mdst_1' =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst_2'$, *and*
– $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$, *and*
– $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$.

Proof. We proof this lemma by structural induction on the derivations of the two judgments $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}$ and $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$.

Hence, let Λ, Λ', $c_1, c_{s1}\, c_2, c_{s2}, c_1' \in Com$, $mem_1, mem_2, mem_1' \in Mem$, $lkst, lkst' \in LkSt$, and $mdst_1, mdst_2, mdst_1' \in MdSt$ be arbitrary such that

– $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}$ and $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ with $c_{s1} = c_{s2}$, and
– $mdst_1, mdst_2 \in comp(lev_1, \Lambda)$, and
– $mdst_1 =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst_2$, and
– $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$, and
– $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$

We make a case distinction on the last rule applied used in the derivation of $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}$.

Case (TSK): By the rule TSK we get $c_1 = c_{s1} = \mathbf{skip}$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda) = pre(\Lambda')$. From $c_1 = \mathbf{skip}$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is SK and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$. Thus, $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$ holds.
From $\vdash_{lev} \Lambda\{c_2\}\Lambda' : \mathbf{skip}$ we get by Lemma 7 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $c_2' = \mathbf{stop}$, $lkst' = lkst$, $mdst_2' = mdst_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$.
From $\Lambda \sqsubseteq \Lambda'$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$. From $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, and $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda) = pre(\Lambda')$ we get by the definition of $comp$ that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst_2$ that $mdst_1' =_{\{\mathsf{A\text{-}NR,A\text{-}NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TAH): By the rule TAH we get that $c_1 = x{:=}e$, $c_{s1} = \mathbf{skip}$, $x \notin pre(\Lambda)$, $lev(x) = \mathbf{high}$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda) = pre(\Lambda')$ and ${\Lambda'}_{lev}\langle x\rangle = \mathbf{high}$.
From $c_1 = x{:=}e$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is AS and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1[x \mapsto v]$ for some v. From $mem_1' = mem_1[x \mapsto v]$ and ${\Lambda'}_{lev}\langle x\rangle = \mathbf{high}$ we get by definition of $=^{lev,\Lambda'}_{\mathbf{low}}$ that $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$.
From $\vdash_{lev} \Lambda\{c_2\}\Lambda' : \mathbf{skip}$ we get by Lemma 7 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $c_2' = \mathbf{stop}$, $lkst' = lkst$, $mdst_2' = mdst_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$.

From $\Lambda \sqsubseteq \Lambda'$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$. From $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, and $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of $comp$ that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TFH): By the rule TFH we get that $c_1 = x{:=}e$, $c_{s1} = \mathbf{skip}$, $x \in pre(\Lambda)$, $\Lambda[x \mapsto \mathbf{high}] \sqsubseteq \Lambda'$, and thus also $pre(\Lambda) = pre(\Lambda')$. From $c_1 = x{:=}e$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is AS and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1[x \mapsto v]$ for some v.
From $mem_1' = mem_1[x \mapsto v]$ and $\Lambda[x \mapsto \mathbf{high}] \sqsubseteq \Lambda'$ we get by definition of $=^{lev,\Lambda'}_{\mathbf{low}}$ that $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$.
From $\Lambda \sqsubseteq \Lambda'$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$. From $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, and $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of $comp$ that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TAL): By the rule TAL we get that $c_1 = x{:=}e$, $c_{s1} = c_{s2} = x{:=}e$, $x \notin pre(\Lambda)$, $lev(x) = \mathbf{low}$, $\vdash_{lev,\Lambda} e : \mathbf{low}$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda') = pre(\Lambda)$. From $c_1 = x{:=}e$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is AS and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1[x \mapsto eval(e, mem_1)]$.
From $c_{s2} = x{:=}e$, $lev(x) = \mathbf{low}$, and $x \notin pre(\Lambda)$ we know that the only rule to derive the judgment $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ can be TAL. Hence, $c_2 = x{:=}e$. Thus we get by the rule AS that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $lkst' = lkst$, $mdst_2' = mdst_2$, and $mem_2' = mem_2[x \mapsto eval(e, mem_2)]$.
From $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $\vdash_{lev,\Lambda} e : \mathbf{low}$ we get that $eval(e, mem_1) = eval(e, mem_2)$. Hence, $mem_1' = mem_1[x \mapsto v]$ and $mem_2' = mem_2[x \mapsto v]$ for $v = eval(e, mem_1)$. Thus, we have $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$. From $\Lambda \sqsubseteq \Lambda'$ and $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of $comp$ that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TFL): By the rule TFL we get that $c_1 = x{:=}e$, $c_{s1} = c_{s2} = x{:=}e$, $x \in pre(\Lambda)$, $\vdash_{lev,\Lambda} e : \mathbf{low}$, $\Lambda[x \mapsto \mathbf{low}] \sqsubseteq \Lambda'$, and thus also $pre(\Lambda') = pre(\Lambda)$. From $c_1 = x{:=}e$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is AS and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1[x \mapsto eval(e, mem_1)]$.
From $c_{s2} = x{:=}e$, $x \in pre(\Lambda)$, we know that the only rule to derive the judgment $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ can be TFL. Hence, $c_2 = x{:=}e$. Thus we get by the rule AS that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $lkst' = lkst$, $mdst_2' = mdst_2$, and $mem_2' = mem_2[x \mapsto eval(e, mem_1)]$.

From $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $\vdash_{lev,\Lambda} e : \mathbf{low}$ we get that $eval(e, mem_1) = eval(e, mem_2)$. Hence, $mem_1' = mem_1[x \mapsto v]$ and $mem_2' = mem_2[x \mapsto v]$ for $v = eval(e, mem_1)$. Thus, we have $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$. From $\Lambda \sqsubseteq \Lambda'$ and $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of *comp* that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TLO): By the rule TLO we get that $c_1 = c_{s1} = c_{s2} = \mathbf{lock}(l)$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda') = pre(\Lambda)$. From $c_1 = \mathbf{lock}(l)$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is LK and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst \cup \{l\}$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$.
From $c_{s2} = \mathbf{lock}(l)$, we know that the only rule to derive the judgment $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ can be TLO. Hence, $c_2 = \mathbf{lock}(l)$. Thus we get by the rule LK that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ with $lkst'' = lkst \cup \{l\}$, $mdst_2' = mdst_2$, and $mem_2' = mem_2$. Since $lkst'' = lkst \cup \{l\}$ and $lkst' = lkst \cup \{l\}$ we have $lkst'' = lkst'$.
From $mem_1' = mem_1$, $mem_2' = mem_2$, and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$. From $\Lambda \sqsubseteq \Lambda'$ and $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of *comp* that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TUL): From the assumption of this case we get by the rule TUL that $c_1 = c_{s1} = c_{s2} = \mathbf{unlock}(l)$, $l \in lkst$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda') = pre(\Lambda)$. From $c_1 = \mathbf{unlock}(l)$ we get that the only rule to derive $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ is ULK and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst \setminus \{l\}$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$.
From $c_{s2} = \mathbf{unlock}(l)$, we know that the only rule to derive the judgment $\vdash_{lev} \Lambda_2'\{c_2\}\Lambda_2 : c_{s2}$ can be TUL. Hence, $c_2 = \mathbf{unlock}(l)$. Since $l \in lkst$ we get by the rule ULK that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ with $lkst'' = lkst\setminus\{l\}$, $mdst_2' = mdst_2$, and $mem_2' = mem_2$. Since $lkst'' = lkst \setminus \{l\}$ and $lkst' = lkst \setminus \{l\}$ we have $lkst'' = lkst'$.
From $mem_1' = mem_1$, $mem_2' = mem_2$, and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$. From $\Lambda \sqsubseteq \Lambda'$ and $mem_1' =^{lev,\Lambda}_{\mathbf{low}} mem_2'$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.
From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of *comp* that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.
Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TSP): From the assumption of this case we get by the rule TSP that $c_1 = \mathbf{spawn}(c_3)$, $c_{s1} = c_{s2} = \mathbf{spawn}(c_{s3})$, $\Lambda \sqsubseteq \Lambda'$, and thus also $pre(\Lambda') = pre(\Lambda)$.
From $c_1 = \mathbf{spawn}(c_3)$ we get that the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ must be SP and, hence, $c_1' = \mathbf{stop}$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$. Thus, $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_{1'}$.
From $c_{s2} = \mathbf{spawn}(c_{s3})$, we know that the last rule in the derivation of $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ must be TSP. Hence, $c_2 = \mathbf{spawn}(c_4)$. Thus we get by the rule SP

that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $lkst' = lkst$, $mdst_2' = mdst_2$, and $mem_2' = mem_2$. Thus, $mem_2 =^{lev,\Lambda'}_{\mathbf{low}} mem_{2'}$.

From $\Lambda \sqsubseteq \Lambda'$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$. From $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, $mem_2' =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, and $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_1$ we get $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.

From $mdst_1' = mdst_1$, $mdst_2' = mdst_2$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, and $pre(\Lambda') = pre(\Lambda)$ we get by the definition of $comp$ that $mdst_1', mdst_2' \in comp(lev, \Lambda')$, and from $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.

Since $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$, we get from the rule TST that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : \mathbf{stop}$ with $\Lambda'' = \Lambda'$.

Case (TSQ): By the rule TSQ that $c_1 = c_3; c_4$, $c_{s1} = c_{s3}; c_{s4}$, $\vdash_{lev} \Lambda\{c_3\}\Lambda_1 : c_{s3}$, and $\vdash_{lev} \Lambda_1\{c_4\}\Lambda' : c_{s3}$.

From $c_1 = c_3; c_4$ and $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ we get by the rule SQ1 and SQ2 that $\langle c_3, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_3', lkst', mdst_1', mem_1'\rangle$.

From $c_{s1} = c_{s3}; c_{s4}$ and $c_{s1} = c_{s2}$ we get that $c_{s2} = c_{s3}; c_{s4}$. Hence, the last rule applied in the derivation of $\vdash_{lev} \Lambda_1\{c_2\}\Lambda_2 : c_{s2}$ must be either TSQ or TIH. We distinguish these two cases.

Case (TSQ): From the assumption of this case, we get by the rule TSQ that $c_2 = c_5; c_6$, $\vdash_{lev} \Lambda\{c_5\}\Lambda_2 : c_{s5}$, and $\vdash_{lev} \Lambda_2\{c_6\}\Lambda' : c_{s6}$ with $c_{s5} = c_{s3}$ and $c_{s6} = c_{s4}$.

From $\vdash_{lev} \Lambda\{c_3\}\Lambda_1 : c_{s3}$ and $\vdash_{lev} \Lambda\{c_5\}\Lambda_2 : c_{s5}$ and $c_{s5} = c_{s3}$ we get by Lemma 10 that $\vdash_{lev} \Lambda\{c_3\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $\vdash_{lev} \Lambda\{c_5\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}$. From $\vdash_{lev} \Lambda_1\{c_4\}\Lambda' : c_{s4}$ and $\vdash_{lev} \Lambda_2\{c_6\}\Lambda' : c_{s6}$ and $c_{s6} = c_{s4}$ we get by Lemma 10 that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}\Lambda' : c_{s4}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_6\}\Lambda' : c_{s6}$.

From $\vdash_{lev} \Lambda\{c_3\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $\vdash_{lev} \Lambda\{c_5\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}$ and $c_{s5} = c_{s3}$ and $mdst_1, mdst_2 \in comp(lev, \Lambda,)$ and $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $\langle c_3, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_3', lkst', mdst_1', mem_1'\rangle$ we get by the induction hypothesis that there is $\alpha' \in Eve$, $mdst_2' \in MdSt$, c_5', c_{s3}', $c_{s5}' \in Com$, $mem_2' \in Mem$, and Λ'' such that

- $\vdash_{lev} \Lambda''\{c_3'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}'$, and $\vdash_{lev} \Lambda''\{c_5'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}'$, with $c_{s3}' = c_{s5}'$,
- $mdst_1', mdst_2' \in comp(lev, \Lambda'')$, and
- $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$, and
- $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$, and
- $\langle c_5, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_5', lkst', mdst_2', mem_2'\rangle$.

We now distinguish two cases based on whether $c_3' = \mathbf{stop}$.

Case ($c_3' = \mathbf{stop}$): In this case, we get from $\vdash_{lev} \Lambda''\{c_3'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}'$ by the rule TST that $c_{s3}' = \mathbf{stop}$ and $\Lambda'' = (\Lambda_1 \sqcup \Lambda_2)$. From $c_{s3}' = \mathbf{stop}$ and $c_{s3}' = c_{s5}'$ we get $c_{s5}' = \mathbf{stop}$. From $c_{s5}' = \mathbf{stop}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda''\{c_5'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}'$ must be TST. From this rule we get that $c_5' = \mathbf{stop}$.

From $c_1 = c_3; c_4$ and $c_3' = \mathbf{stop}$ and $\langle c_3, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_3', lkst', mdst_1', mem_1'\rangle$ we get that the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ must be SQ2. From this rule we get that $c_1' = c_4$.

From $c_2 = c_5; c_6$ and $c_5' = \mathbf{stop}$ and $\langle c_5, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_5', lkst', mdst_1', mem_1'\rangle$ we get that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst', mdst_2', mem_2'\rangle$ with $c_2' = c_6$ is derivable with the rule SQ2.

Since $\Lambda'' = (\Lambda_1 \sqcup \Lambda_2)$ and

- $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}\Lambda' : c_{s4}$, and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_6\}\Lambda' : c_{s6}$, with $c_{s4} = c_{s6}$,

· $mdst_1', mdst_2' \in comp(lev, \Lambda'')$, and
· $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$, and
· $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$, and
· $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst', mdst_2', mem_2'\rangle$.

we can conclude this case.

Case ($c_3' \neq \mathbf{stop}$): In this case, we get from $\vdash_{lev} \Lambda''\{c_3'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}'$ by the typing rules that $c_{s3}' \neq \mathbf{stop}$. From $c_{s3}' \neq \mathbf{stop}$ and $c_{s3}' = c_{s5}'$ we get $c_{s5}' \neq \mathbf{stop}$. From $c_{s5}' \neq \mathbf{stop}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda''\{c_5'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}'$ cannot be TST and, hence, we get from the typing rules that $c_5' \neq \mathbf{stop}$

From $c_3' \neq \mathbf{stop}$ and $\langle c_3, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_3', lkst', mdst_1', mem_1'\rangle$ we get that the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ must be SQ1. From this rule we get that $c_1' = c_3'; c_4$.

From $c_2 = c_5; c_6$ and $c_5' \neq \mathbf{stop}$ and $\langle c_5, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_5', lkst', mdst_2', mem_2'\rangle$ we get by SQ1 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ is derivable with $c_2' = c_5'; c_6$.

From $c_1' = c_3'; c_4$ and $\vdash_{lev} \Lambda''\{c_3'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}'$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}\Lambda' : c_{s4}$ we get by the rule TSQ that $\vdash_{lev} \Lambda''\{c_3'; c_4\}\Lambda' : c_{s3}'; c_{s4}$ is derivable.

From $c_2' = c_5'; c_6$ and $\vdash_{lev} \Lambda''\{c_5'\}(\Lambda_1 \sqcup \Lambda_2) : c_{s5}'$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_6\}\Lambda' : c_{s6}$ we get by the rule TSQ that $\vdash_{lev} \Lambda''\{c_5'; c_6\}\Lambda' : c_{s5}'; c_{s6}$ is derivable.

From $c_{s3}' = c_{s5}'$ and $c_{s4} = c_{s6}$ we get that $c_{s3}'; c_{s4} = c_{s5}'; c_{s6}$. Hence, we can conclude this case.

Case (TIH): From the assumption of this case, we get by the rule TIH that $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_5\ \mathbf{else}\ c_6\ \mathbf{fi}$ and $c_{s2} = \mathbf{skip}; c_5$ and $\vdash_{lev} \Lambda\{c_5\}\Lambda' : c_{s5}$, and $\vdash_{lev} \Lambda\{c_6\}\Lambda' : c_{s6}$ with $c_{s5} = c_{s6}$. From $c_{s1} = c_{s2}$ and $c_{s1} = c_{s3}; c_{s4}$ and $c_{s2} = \mathbf{skip}; c_5$ we get that $c_{s3} = \mathbf{skip}$ and $c_{s4} = c_{s5}$.

From $c_{s3} = \mathbf{skip}$ and $\vdash_{lev} \Lambda\{c_3\}\Lambda_1 : c_{s3}$ we get by Lemma 7 and the fact that commands evaluate deterministically in our language that $c_3' = \mathbf{stop}$ and $lkst' = lkst$ and $mdst_1' = mdst_1$ and $mem_1' =^{lev,\Lambda_1}_{\mathbf{low}} mem_1$ and $\Lambda \sqsubseteq \Lambda_1$. From $\Lambda \sqsubseteq \Lambda_1$ we get that $pre(\Lambda) = pre(\Lambda_1)$. From $c_1 = c_3; c_4$ and $c_3' = \mathbf{stop}$ and $\langle c_3, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_3', lkst', mdst_1', mem_1'\rangle$ we get that the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ must be SQ2. From this rule we get that $c_1' = c_4$.

From $\vdash_{lev} \Lambda_1\{c_4\}\Lambda' : c_{s4}$ and $\vdash_{lev} \Lambda\{c_5\}\Lambda' : c_{s5}$ and $\vdash_{lev} \Lambda\{c_6\}\Lambda' : c_{s6}$ and $c_{s5} = c_{s6}$ and $c_{s4} = c_{s5}$ we get by Lemma 10 that $\vdash_{lev} (\Lambda \sqcup \Lambda_1)\{c_4\}\Lambda' : c_{s4}$ and $\vdash_{lev} (\Lambda \sqcup \Lambda_1)\{c_5\}\Lambda' : c_{s5}$ and $\vdash_{lev} (\Lambda \sqcup \Lambda_1)\{c_6\}\Lambda' : c_{s6}$.

From $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_5\ \mathbf{else}\ c_6\ \mathbf{fi}$ we get by the rule IFT and IFF that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ is derivable with $c_2' = c_i$ for some $i \in \{5, 6\}$ and $lkst'' = lkst$ and $mdst_2' = mdst_2$ and $mem_2' = mem_2$.

It remains to show that
* $mdst_1', mdst_2' \in comp(lev, (\Lambda \sqcup \Lambda_1))$, and
* $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$, and
* $mem_1' =^{lev,(\Lambda \sqcup \Lambda_1)}_{\mathbf{low}} mem_2'$.

From $mdst_1 = mdst_1'$ and $mdst_2 = mdst_2'$ and $pre(\Lambda) = pre(\Lambda_1) = pre((\Lambda \sqcup \Lambda_1))$ and $mdst_1, mdst_2 \in comp(lev, \Lambda)$ we get that $mdst_1', mdst_2' \in comp(lev, (\Lambda \sqcup \Lambda_1))$.

From $mdst_1 = mdst_1'$ and $mdst_2 = mdst_2'$ and $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ we get that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.

From $mem_1' =^{lev,\Lambda_1}_{\mathbf{low}} mem_1$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $mem_2' = mem_2$ and $\Lambda \sqsubseteq (\Lambda \sqcup \Lambda_1)$ and $\Lambda_1 \sqsubseteq (\Lambda \sqcup \Lambda_1)$ we get that $mem_1' =^{lev,(\Lambda \sqcup \Lambda_1)}_{\mathbf{low}} mem_2'$.

Case (TIL): By the rule TIL we get that $c_1 = \mathbf{if}\ e\ \mathbf{then}\ c_3\ \mathbf{else}\ c_4\ \mathbf{fi}$, $c_{s1} = c_{s2} = \mathbf{if}\ e\ \mathbf{then}\ c_{s3}\ \mathbf{else}\ c_{s4}\ \mathbf{fi}$, $\vdash_{lev,\Lambda} e : \mathbf{low}$, $\vdash_{lev} \Lambda\{c_3\}\Lambda' : c_{s3}$, and $\vdash_{lev} \Lambda\{c_4\}\Lambda' : c_{s4}$.
From $c_{s2} = \mathbf{if}\ e\ \mathbf{then}\ c_{s3}\ \mathbf{else}\ c_{s4}\ \mathbf{fi}$ we get that the last rule applied in the derivation of $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ must be TIL. From $c_{s2} = \mathbf{if}\ e\ \mathbf{then}\ c_{s3}\ \mathbf{else}\ c_{s4}\ \mathbf{fi}$ we get by this rule that $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_5\ \mathbf{else}\ c_6\ \mathbf{fi}$, $\vdash_{lev} \Lambda\{c_5\}\Lambda' : c_{s3}$, and $\vdash_{lev} \Lambda\{c_6\}\Lambda' : c_{s3}$.
From $\vdash_{lev,\Lambda} e : \mathbf{low}$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$, we get that $eval(e, mem_1) = eval(e, mem_2)$. We now distinguish two cases based on whether $eval(e, mem_1) = \mathbf{true}$ or $eval(e, mem_1) = \mathbf{false}$.

Case ($eval(e, mem_1) = \mathbf{true}$): From the assumption of this case we get by the rule IFT that $c_1' = c_3$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$.
From $eval(e, mem_1) = eval(e, mem_2)$ and the assumption of this case we also get by the rule IFT that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ is derivable with $c_2' = c_5$, $lkst'' = lkst$, $mdst_2' = mdst_2$, and $mem_2' = mem_2$.
Since $\vdash_{lev} \Lambda\{c_3\}\Lambda' : c_{s3}$, $\vdash_{lev} \Lambda\{c_5\}\Lambda' : c_{s3}$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$, and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$, we can conclude this case.

Case ($eval(e, mem_1) = \mathbf{false}$): From the assumption of this case we get by the rule IFF that $c_1' = c_4$, $lkst' = lkst$, $mdst_1' = mdst_1$, and $mem_1' = mem_1$.
From $eval(e, mem_1) = eval(e, mem_2)$ and the assumption of this case we also get by the rule IFF that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ is derivable with $c_2' = c_6$, $lkst'' = lkst$, $mdst_2' = mdst_2$, and $mem_2' = mem_2$.
Since $\vdash_{lev} \Lambda\{c_4\}\Lambda' : c_{s4}$, $\vdash_{lev} \Lambda\{c_6\}\Lambda' : c_{s4}$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$, and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$, we can conclude this case.

Case (TIH): From the assumption of this case we get by the rule TIH that $c_1 = \mathbf{if}\ e\ \mathbf{then}\ c_3\ \mathbf{else}\ c_4\ \mathbf{fi}$, $c_{s1} = \mathbf{skip}; c_{s3}$, $\vdash_{lev} \Lambda\{c_3\}\Lambda' : c_{s3}$, $\vdash_{lev} \Lambda\{c_4\}\Lambda' : c_{s4}$, and $c_{s3} = c_{s4}$.
From $c_1 = \mathbf{if}\ e\ \mathbf{then}\ c_3\ \mathbf{else}\ c_4\ \mathbf{fi}$ and $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ we get by IFT and IFF that $c_1' = c_i$ for some $i \in \{3, 4\}$ and $lkst' = lkst$ and $mdst_1' = mdst_1$ and $mem_1' = mem_1$.
From $c_{s1} = c_{s2}$ and $c_{s1} = \mathbf{skip}; c_{s3}$ we get that $c_{s2} = \mathbf{skip}; c_{s3}$. From $c_{s2} = \mathbf{skip}; c_{s3}$ we get that that last rule in the derivation of $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ must be either TIH or TSQ. We distinguish these two cases.

Case (TIH): In this case, we get by the rule TIH that $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_5\ \mathbf{else}\ c_6\ \mathbf{fi}$ and $\vdash_{lev} \Lambda\{c_5\}\Lambda' : c_{s5}$ and $\vdash_{lev} \Lambda\{c_6\}\Lambda' : c_{s6}$ and $c_{s3} = c_{s5} = c_{s6}$.
From $c_2 = \mathbf{if}\ e\ \mathbf{then}\ c_5\ \mathbf{else}\ c_6\ \mathbf{fi}$ we get that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ is derivable with the rule IFT or IFF, and $c_2' = c_i$ for some $i \in \{5, 6\}$ and $lkst'' = lkst$ and $mdst_2' = mdst_2$ and $mem_2' = mem_2$.
Hence, all conditions that we need to show hold directly due to the assumptions of this case.

Case (TSQ): In this case, we get by the rule TSQ that $c_2 = c_5; c_6$ and $\vdash_{lev} \Lambda\{c_5\}\Lambda_2 : \mathbf{skip}$, and $\vdash_{lev} \Lambda_2\{c_6\}\Lambda' : c_{s6}$ with $c_{s6} = c_{s3}$.
From $\vdash_{lev} \Lambda\{c_5\}\Lambda_2 : \mathbf{skip}$ we get by Lemma 7 that $\langle c_5, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_5', lkst'', mdst_2', mem_2'\rangle$ is derivable with $c_5' = \mathbf{stop}$ and $lkst'' = lkst$ and $mdst_2' = mdst_2$ and $mem_2' =^{lev,\Lambda_2}_{\mathbf{low}} mem_2$ and $\Lambda \sqsubseteq \Lambda_2$. From $\Lambda \sqsubseteq \Lambda_2$ we get that $pre(\Lambda) = pre(\Lambda_2)$. From $\langle c_5, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_5', lkst'', mdst_2', mem_2'\rangle$ and $c_5' = \mathbf{stop}$ we get by the rule SQ2 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst'', mdst_2', mem_2'\rangle$ with $c_2' = c_6$.

From $c_{s6} = c_{s3}$ and $c_{s3} = c_{s4}$ and $pre(\Lambda) = pre(\Lambda_2)$ and $\vdash_{lev} \Lambda_2\{c_6\}\Lambda' : c_{s6}$ and $\vdash_{lev} \Lambda\{c_3\}\Lambda' : c_{s3}$ and $\vdash_{lev} \Lambda\{c_4\}\Lambda' : c_{s4}$ we get by Lemma 10 that $\vdash_{lev} (\Lambda \sqcup \Lambda_2)\{c_6\}\Lambda' : c_{s6}$ and $\vdash_{lev} (\Lambda \sqcup \Lambda_2)\{c_3\}\Lambda' : c_{s3}$ and $\vdash_{lev} (\Lambda \sqcup \Lambda_2)\{c_4\}\Lambda' : c_{s4}$.
It remains to show that

- $mdst'_1, mdst'_2 \in comp(lev, (\Lambda \sqcup \Lambda_2))$, and
- $mdst'_1 =_{\{\text{A-NR,A-NW}\}} mdst'_2$, and
- $mem'_1 =^{lev,(\Lambda \sqcup \Lambda_2)}_{\mathbf{low}} mem'_2$.

From $mdst_1 = mdst'_1$ and $mdst_2 = mdst'_2$ and $pre(\Lambda) = pre(\Lambda_1) = pre((\Lambda \sqcup \Lambda_2))$ and $mdst_1, mdst_2 \in comp(lev, \Lambda)$ we get that $mdst'_1, mdst'_2 \in comp(lev, (\Lambda \sqcup \Lambda_2))$.
From $mdst_1 = mdst'_1$ and $mdst_2 = mdst'_2$ and $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ we get that $mdst'_1 =_{\{\text{A-NR,A-NW}\}} mdst'_2$.
From $mem'_2 =^{lev,\Lambda_2}_{\mathbf{low}} mem_2$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $mem'_1 = mem_1$ and $\Lambda \sqsubseteq (\Lambda \sqcup \Lambda_2)$ and $\Lambda_2 \sqsubseteq (\Lambda \sqcup \Lambda_2)$ we get that $mem'_1 =^{lev,(\Lambda \sqcup \Lambda_2)}_{\mathbf{low}} mem'_2$.

Case (TWL): We get by the rule TWL that $c_1 = \mathbf{while}\ e\ \mathbf{do}\ c_3\ \mathbf{od}$ and $c_{s1} = \mathbf{while}\ e\ \mathbf{do}\ c_{s3}\ \mathbf{od}$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $\Lambda \sqsubseteq \Lambda_1$ and $\Lambda_1 \sqsubseteq \Lambda'$ and $\vdash_{lev} \Lambda_1\{c_3\}\Lambda_1 : c_{s3}$.
From $c_{s1} = \mathbf{while}\ e\ \mathbf{do}\ c_{s3}\ \mathbf{od}$ and $c_{s1} = c_{s2}$ we get that $c_{s2} = \mathbf{while}\ e\ \mathbf{do}\ c_{s3}\ \mathbf{od}$.
Hence, the last rule applied in the derivation of $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$ must be TWL.
From this rule we get that $c_2 = \mathbf{while}\ e\ \mathbf{do}\ c_4\ \mathbf{od}$ and $\Lambda \sqsubseteq \Lambda_2$ and $\Lambda_2 \sqsubseteq \Lambda'$ and $\vdash_{lev} \Lambda_2\{c_4\}\Lambda_2 : c_{s3}$ and $\vdash_{lev,\Lambda_2} e : \mathbf{low}$.
From $\Lambda \sqsubseteq \Lambda_1$ and $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $eval(e, mem_1) = eval(e, mem_2)$.
We now distinguish two cases based on whether $eval(e, mem_1) = \mathbf{false}$ or $eval(e, mem_1) = \mathbf{true}$.

Case ($eval(e, mem_1) = \mathbf{false}$): From the assumption of this case we get by the rule WHF that $c'_1 = \mathbf{stop}$, $lkst' = lkst$, $mdst'_1 = mdst_1$, and $mem'_1 = mem_1$.
From $c_2 = \mathbf{while}\ e\ \mathbf{do}\ c_4\ \mathbf{od}$ and $eval(e, mem_1) = eval(e, mem_2)$ and the assumption of this case we also get by the rule WHF that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c'_2, lkst'', mdst'_2, mem'_2\rangle$ is derivable with $c'_2 = \mathbf{stop}$, $lkst'' = lkst$, $mdst'_2 = mdst_2$, and $mem'_2 = mem_2$.
From $\Lambda \sqsubseteq \Lambda_1$ and $\Lambda_1 \sqsubseteq \Lambda'$ and $mem'_1 = mem_1$ and $mem'_2 = mem_2$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get that $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$.
From $\Lambda \sqsubseteq \Lambda_1$ and $\Lambda_1 \sqsubseteq \Lambda$ we get that $pre(\Lambda) = pre(\Lambda')$. Hence, we get from $mdst_1, mdst_2 \in comp(lev, \Lambda)$, by definition of $comp$ that $mdst_1, mdst_2 \in comp(lev, \Lambda')$.
Since $c'_1 = \mathbf{stop}$ and $c'_2 = \mathbf{stop}$ and $\vdash_{lev} \Lambda'\{\mathbf{stop}\}\Lambda' : \mathbf{stop}$ and $mdst_1, mdst_2 \in comp(lev, \Lambda')$ and $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$ and $mem_1 =^{lev,\Lambda'}_{\mathbf{low}} mem_2$, we can conclude this case.

Case ($eval(e, mem_1) = \mathbf{true}$): From the assumption of this case we get by the rule WHT that $c'_1 = c_3; c_1$ and $lkst' = lkst$ and $mdst'_1 = mdst_1$ and $mem'_1 = mem_1$.
From $c_2 = \mathbf{while}\ e\ \mathbf{do}\ c_4\ \mathbf{od}$ and $eval(e, mem_1) = eval(e, mem_2)$ and the assumption of this case we also get by the rule WHF that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c'_2, lkst'', mdst'_2, mem'_2\rangle$ is derivable with $c'_2 = c_4; c_2$, $lkst'' = lkst$, $mdst'_2 = mdst_2$, and $mem'_2 = mem_2$.
From $\vdash_{lev,\Lambda_1} e : \mathbf{low}$ and $\vdash_{lev,\Lambda_2} e : \mathbf{low}$ we get that $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \mathbf{low}$.
From $\vdash_{lev} \Lambda_1\{c_3\}\Lambda_1 : c_{s3}$ and $\vdash_{lev} \Lambda_2\{c_4\}\Lambda_2 : c_{s3}$ we get by Lemma 10 that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_3\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$.
From $\Lambda_1 \sqsubseteq \Lambda'$ and $\Lambda_2 \sqsubseteq \Lambda'$ we get that $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq \Lambda'$.

From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1 \sqcup \Lambda_2)$ and $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq \Lambda'$ and $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \mathbf{low}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_3\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $c_1 = \mathbf{while}\ e\ \mathbf{do}\ c_3\ \mathbf{od}$ we get by the rule TWL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}\Lambda' : c_{s1}$. From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_3\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1\}\Lambda' : c_{s1}$ and $c_1' = c_3; c_1$ we get by the rule TSQ that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_1'\}\Lambda' : c_{s3}; c_{s1}$.

From $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq (\Lambda_1 \sqcup \Lambda_2)$ and $(\Lambda_1 \sqcup \Lambda_2) \sqsubseteq \Lambda'$ and $\vdash_{lev,(\Lambda_1 \sqcup \Lambda_2)} e : \mathbf{low}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $c_2 = \mathbf{while}\ e\ \mathbf{do}\ c_4\ \mathbf{od}$ we get by the rule TWL that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_2\}\Lambda' : c_{s1}$. From $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_4\}(\Lambda_1 \sqcup \Lambda_2) : c_{s3}$ and $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_2\}\Lambda' : c_{s1}$ and $c_2' = c_4; c_2$ we get by the rule TSQ that $\vdash_{lev} (\Lambda_1 \sqcup \Lambda_2)\{c_2'\}\Lambda' : c_{s3}; c_{s1}$.

It remains to show that

* $mdst_1', mdst_2' \in comp(lev, (\Lambda_1 \sqcup \Lambda_2))$, and
* $mdst_1' =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2'$, and
* $mem_1' =^{lev,(\Lambda_1 \sqcup \Lambda_2)}_{\mathbf{low}} mem_2'$.

From $mdst_1 = mdst_1'$ and $mdst_2 = mdst_2'$ and $pre(\Lambda) = pre(\Lambda_1) = pre(\Lambda_2) = pre((\Lambda_1 \sqcup \Lambda_2))$ and $mdst_1, mdst_2 \in comp(lev, \Lambda)$ we get that $mdst_1', mdst_2' \in comp(lev, (\Lambda_1 \sqcup \Lambda_2))$.

From $mdst_1 = mdst_1'$ and $mdst_2 = mdst_2'$ and $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$ we get that $mdst_1' =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2'$.

From $mem_2' = mem_2$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ and $mem_1' = mem_1$ and $\Lambda \sqsubseteq (\Lambda_1 \sqcup \Lambda_2)$ we get that $mem_1' =^{lev,(\Lambda \sqcup \Lambda_2)}_{\mathbf{low}} mem_2'$. Hence, we can conclude this case.

Case (TAN): From the assumption of this case we get by the rule TAN that $c_1 = c_1''@\overrightarrow{a}_1$, $\vdash_{lev} \Lambda\{c_1''\}\Lambda_1 : c_{s1}''$, $\Lambda' = (\Lambda_1 \oplus_{lev} \overrightarrow{a}_1)$, $\forall x.{\Lambda_1}_{lev}\langle x\rangle \sqsubseteq {\Lambda'}_{lev}\langle x\rangle$, and $c_{s1} = c_{s1}''@\overrightarrow{a}_1 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$.

We first show, that the last rule in the type derivation for c_2 must TAN and how the variables in this last step must be instantiated. From $c_{s1} = c_{s2}$ and $c_{s1} = c_{s1}''@\overrightarrow{a}_1 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$ we get that the last rule to derive $\vdash_{lev} \Lambda_1'\{c_2\}\Lambda_2 : c_{s2}$ must be TAN. Thus, we get from this rule that $c_2 = c_2''@\overrightarrow{a}_2$, $\vdash_{lev} \Lambda\{c_2''\}\Lambda_2 : c_{s2}''$, $\Lambda' = (\Lambda_2 \oplus_{lev} \overrightarrow{a}_2)$, $\forall x.{\Lambda_2}_{lev}\langle x\rangle \sqsubseteq {\Lambda'}_{lev}\langle x\rangle$, and $c_{s2} = c_{s2}''@\overrightarrow{a}_2 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$. From $c_{s1} = c_{s2}$ and $c_{s2} = c_{s2}''@\overrightarrow{a}_2 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$, we get that $c_{s2} = c_{s1}''@\overrightarrow{a}_1 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$, $c_{s2}'' = c_{s1}''$, and $\overrightarrow{a}_2 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}} = \overrightarrow{a}_1 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$.

Now we show that c_1'' and c_2'' can be typed with the same resulting partial type environment and this type environment can still fulfill the premises for TAN. From $\vdash_{lev} \Lambda\{c_1''\}\Lambda_1 : c_{s1}''$ and $\vdash_{lev} \Lambda\{c_2''\}\Lambda_2 : c_{s2}''$ and $c_{s1}'' = c_{s2}''$ we get by Lemma 8 that $pre(\Lambda_1) = pre(\Lambda_2)$. Hence, we get from Lemma 10 that $\vdash_{lev} \Lambda\{c_1''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}''$ and $\vdash_{lev} \Lambda\{c_2''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s2}''$. Since $\forall x.{\Lambda_1}_{lev}\langle x\rangle \sqsubseteq {\Lambda'}_{lev}\langle x\rangle$ and $\forall x.{\Lambda_2}_{lev}\langle x\rangle \sqsubseteq {\Lambda'}_{lev}\langle x\rangle$ we also have $\forall x.(\Lambda_1 \sqcup \Lambda_2)_{lev}\langle x\rangle \sqsubseteq {\Lambda'}_{lev}\langle x\rangle$. From $\Lambda' = (\Lambda_1 \oplus_{lev} \overrightarrow{a}_1)$ and $\Lambda' = (\Lambda_2 \oplus_{lev} \overrightarrow{a}_2)$ and $pre(\Lambda_1) = pre(\Lambda_2)$ and $\overrightarrow{a}_2 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}} = \overrightarrow{a}_1 \restriction_{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}}$ we get by definition of $\oplus_{lev}$ that $\Lambda' = (\Lambda_1 \sqcup \Lambda_2) \oplus_{lev} \overrightarrow{a}_1$.

From $c_1 = c_1''@\overrightarrow{a}_1$ we get that the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$ must be either AN1 or AN2. From these rules we get that $\langle c_1'', lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1''', lkst', mdst_1'', mem_1'\rangle$ is derivable. From $\vdash_{lev} \Lambda\{c_1''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}''$ and $\vdash_{lev} \Lambda\{c_2''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s2}''$ and $c_{s1}'' = c_{s2}''$ and $mdst_1, mdst_2 \in comp(lev, \Lambda)$ and $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$ we get by the induction hypothesis that there is $\alpha' \in Eve$, $mdst_2'' \in MdSt$, $c_2''', c_{s1}''', c_{s2}''' \in Com$, $mem_2' \in Mem$, and Λ'' such that $\vdash_{lev} \Lambda''\{c_1'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}'''$, and $\vdash_{lev} \Lambda''\{c_2'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s2}'''$, with $c_{s1}''' = c_{s2}'''$, and $mdst_1'', mdst_2'' \in comp(lev, \Lambda'')$, and

$mdst_1'' =_{\{\text{A-NR,A-NW}\}} mdst_2''$, and $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$, and $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$.

We now distinguish two cases depending on the last rule in the derivation of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$.

Case (AN1): In this case, we get by the rule AN1 that $c_1''' = \mathbf{stop}$ and $mdst_1' = updMds(mdst_1'', \overrightarrow{a}_1)$ and $c_1' = \mathbf{stop}$.

From $c_1''' = \mathbf{stop}$ we get that the last rule in the derivation of $\vdash_{lev} \Lambda''\{c_1'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}'''$ must be TST. Hence, $c_{s1}''' = \mathbf{stop}$ and $\Lambda'' = (\Lambda_1 \sqcup \Lambda_2)$. From $c_{s1}''' = \mathbf{stop}$ and $c_{s1}''' = c_{s2}'''$ we get $c_{s2}''' = \mathbf{stop}$. Hence, the last rule in the derivation of $\vdash_{lev} \Lambda''\{c_2'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s2}'''$ must also be TST and, thus, $c_2''' = \mathbf{stop}$.

From $c_2''' = \mathbf{stop}$ and $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$ and $c_2 = c_2''@\overrightarrow{a}_2$ we get by the rule AN1 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ is derivable with $mdst_2' = updMds(mdst_2'', \overrightarrow{a}_2)$ and $c_2' = \mathbf{stop}$.

From $c_1' = \mathbf{stop}$ and $c_2' = \mathbf{stop}$ we get by the rule TST that $\vdash_{lev} \Lambda'\{c_1'\}\Lambda' : \mathbf{stop}$ and $\vdash_{lev} \Lambda'\{c_2'\}\Lambda' : \mathbf{stop}$.

It remains to show that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$ and $mdst_1', mdst_2' \in comp(lev, \Lambda')$ and $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.

From $mdst_1' = updMds(mdst_1'', \overrightarrow{a}_1)$ and $mdst_2' = updMds(mdst_2'', \overrightarrow{a}_2)$ and $\overrightarrow{a}_2 \restriction_{\text{A-NR,A-NW}} = \overrightarrow{a}_1 \restriction_{\text{A-NR,A-NW}}$ we get by the definition of $updMds$ that $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$.

From $\Lambda'' = (\Lambda_1 \sqcup \Lambda_2)$ and $\Lambda' = (\Lambda_1 \sqcup \Lambda_2) \oplus_{lev} \overrightarrow{a}_1$ we get that $\Lambda' = (\Lambda'' \oplus_{lev} \overrightarrow{a}_1)$. From $mdst_1'' \in comp(lev, \Lambda'')$ and $mdst_1' = updMds(mdst_1'', \overrightarrow{a}_1)$ and $\Lambda' = (\Lambda'' \oplus_{lev} \overrightarrow{a}_1)$ we get that $mdst_1' \in comp(lev, \Lambda')$. From $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$ and $mdst_1' \in comp(lev, \Lambda')$ we get that $mdst_2' \in comp(lev, \Lambda')$.

From $\forall x.(\Lambda_1 \sqcup \Lambda_2)_{lev}\langle x\rangle \sqsubseteq \Lambda'_{lev}\langle x\rangle$ and $\Lambda'' = (\Lambda_1 \sqcup \Lambda_2)$ and $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$ we get that $mem_1' =^{lev,\Lambda'}_{\mathbf{low}} mem_2'$.

Case (AN2): In this case, we get by the rule AN2 that $c_1''' \neq \mathbf{stop}$ and $mdst_1' = mdst_1''$ and $c_1' = c_1'''@\overrightarrow{a}_1$. From $c_1''' \neq \mathbf{stop}$ and $\vdash_{lev} \Lambda''\{c_1'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}'''$ we get that $c_{s1}''' \neq \mathbf{stop}$. Hence, we get from $c_{s1}''' = c_{s2}'''$ that $c_{s2}''' \neq \mathbf{stop}$.

From $c_2''' \neq \mathbf{stop}$ and $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$ and $c_2 = c_2''@\overrightarrow{a}_2$ we get by the rule AN2 that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$ is derivable with $mdst_2' = mdst_2''$ and $c_2' = c_2'''@\overrightarrow{a}_2$.

From $c_1' = c_1'''@\overrightarrow{a}_1$ and $c_2' = c_2'''@\overrightarrow{a}_2$ and $\vdash_{lev} \Lambda''\{c_1'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s1}'''$, and $\vdash_{lev} \Lambda''\{c_2'''\}(\Lambda_1 \sqcup \Lambda_2) : c_{s2}'''$ and $\forall x.(\Lambda_1 \sqcup \Lambda_2)_{lev}\langle x\rangle \sqsubseteq \Lambda'_{lev}\langle x\rangle$ and $\Lambda' = (\Lambda_1 \sqcup \Lambda_2) \oplus_{lev} \overrightarrow{a}_1$ we get by the rule TAN that $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : c_{s1}'$ and $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : c_{s2}'$ with $c_{s1}' = c_{s1}'''@\overrightarrow{a}_1 \restriction_{\text{A-NR,A-NW}}$ and $c_{s2}' = c_{s2}'''@\overrightarrow{a}_2 \restriction_{\text{A-NR,A-NW}}$. Hence, from $c_{s1}''' = c_{s2}'''$ and $\overrightarrow{a}_1 \restriction_{\text{A-NR,A-NW}} = \overrightarrow{a}_2 \restriction_{\text{A-NR,A-NW}}$ we get $c_{s1}' = c_{s2}'$.

From $mdst_1'', mdst_2'' \in comp(lev, \Lambda'')$ and $mdst_1'' =_{\{\text{A-NR,A-NW}\}} mdst_2''$ and $mdst_1' = mdst_1''$ and $mdst_2' = mdst_2''$ we get that $mdst_1', mdst_2'' \in comp(lev, \Lambda'')$ and $mdst_1'' =_{\{\text{A-NR,A-NW}\}} mdst_2''$.

Since we already obtained $mem_1' =^{lev,\Lambda''}_{\mathbf{low}} mem_2'$ from the induction hypothesis, we can conclude this case.

□

We now show that whenever two commands have identical **low**-slices and the first command spawns a new thread, then the second command can also spawn a new thread in its next step and the commands of the spawned threads have identical **low**-slices.

Lemma 12. *If* $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}, \vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$, $c_{s1} = c_{s2}$, *and* $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1', lkst', mdst_1', mem_1'\rangle$,
then there is $c_2', c_4 \in Com$, $mdst_2, mdst_2' \in MdSt$, $mem_2' \in Mem$, *and* Λ_3 *such that* $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_2', lkst'', mdst_2', mem_2'\rangle$, $\vdash_{lev} \Lambda_3\{c_3\}\Lambda_3 : c_{s3}$, $\vdash_{lev} \Lambda_3\{c_4\}\Lambda_3 : c_{s4}$, $c_{s3} = c_{s4}$, *and* $pre(\Lambda_3) = \emptyset$.

Proof. We prove this by structural induction on the derivation height of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1', lkst', mdst_1', mem_1'\rangle$.

The induction base is the tuple a derivation height of 1. From $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1', lkst', mdst_1', mem_1'\rangle$, we know that derivations of height 1 are only possible with the rule SP and, hence, $c_1 = \mathbf{spawn}(c_3)$. Thus, the only rule to derive $\vdash_{lev} \Lambda_1\{c_1\}\Lambda_1' : c_{s1}$ is TSP. From this rule we get $c_1 = \mathbf{spawn}(c_3)$, $c_{s1} = \mathbf{spawn}(c_{s3})$, $\vdash_{lev} \Lambda_3\{c_3\}\Lambda_3 : c_{s3}$, and $pre(\Lambda_3) = \emptyset$.

From $c_{s1} = c_{s2}$ we get that $c_{s2} = \mathbf{spawn}(c_{s4})$ with $c_{s3} = c_{s4}$. Hence, we know that the last rule used in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must have been TSP. From this rule we get $c_2 = \mathbf{spawn}(c_4)$, $\vdash_{lev} \Lambda_4\{c_4\}\Lambda_4 : c_{s4}$, and $pre(\Lambda_4) = \emptyset$. Since $pre(\Lambda_4) = \emptyset$ and $pre(\Lambda_3) = \emptyset$. From $c_2 = \mathbf{spawn}(c_4)$, we get by semantics rule SP that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle \mathbf{stop}, lkst, mdst_2', mem_2'\rangle$ is derivable.

For the induction step, let $n > 1$ be the height of the derivation. Derivations of $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1', lkst', mdst_1', mem_1'\rangle$ with a height of $n > 1$ are only possible with the rules AN1, AN2, SQ1, and SQ2. Hence, $c_1 = c_1''@\overrightarrow{a}_1$, or $c_1 = c_1''; c_1'''$. We distinguish these two cases.

Case ($c_1 = c_1''@\overrightarrow{a}_1$): From the assumption of this case, we get by the rule TAN that $c_{s1} = c_{s1}''@\overrightarrow{a}_1 \restriction_{\text{A-NR,A-NW}}$ and $\vdash_{lev} \Lambda_1\{c_1''\}\Lambda_1'' : c_{s1}''$. Thus, we get from $c_{s1} = c_{s2}$ that $c_{s2} = c_{s1}''@\overrightarrow{a}_1 \restriction_{\text{A-NR,A-NW}}$. Hence, the last typing rule in the derivation of $\vdash_{lev} \Lambda_1'\{c_2\}\Lambda_2' : c_{s2}$ must be TAN. From this rule we get that $c_2 = c_2''@\overrightarrow{a}_2$ and $\vdash_{lev} \Lambda_2\{c_2''\}\Lambda_2'' : c_{s2}''$ with $c_{s1}'' = c_{s2}''$.
From $c_1 = c_1''@\overrightarrow{a}_1$ we get by the rules AN1 and AN2 that $\langle c_1'', lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1''', lkst', mdst_1'', mem_1'\rangle$. Since $c_{s1}'' = c_{s2}''$, we get from the induction hypothesis that there is $c_2''', c_4 \in Com$, $mdst_2, mdst_2'' \in MdSt$, $mem_2' \in Mem$, and Λ_3 such that $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$, $\vdash_{lev} \Lambda_3\{c_3\}\Lambda_3 : c_{s3}$, $\vdash_{lev} \Lambda_3\{c_4\}\Lambda_3 : c_{s4}$, $c_{s3} = c_{s4}$, and $pre(\Lambda_3) = \emptyset$.
It remains to show that there is $c_2' \in Com$, $mdst_2' \in MdSt$, such that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_2', lkst', mdst_2', mem_2'\rangle$. This follows directly from $c_2 = c_2''@\overrightarrow{a}_2$ and $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$ by the rules AN1 and AN2.

Case ($c_1 = c_1''; c_1'''$): From the assumption of this case, we get by the rule TSQ that $c_{s1} = c_{s1}''; c_{s1}'''$ and $\vdash_{lev} \Lambda_1\{c_1''\}\Lambda_1'' : c_{s1}''$. Hence, the last typing rule in the derivation of $\vdash_{lev} \Lambda_2\{c_2\}\Lambda_2' : c_{s2}$ must be TSQ. From this rule we get that $c_2 = c_2''; c_2'''$ and $\vdash_{lev} \Lambda_2\{c_2''\}\Lambda_2'' : c_{s2}''$ with $c_{s1}'' = c_{s2}''$.
From $c_1 = c_1''; c_1'''$ we get by the rules SQ1 and SQ2 that $\langle c_1'', lkst, mdst_1, mem_1\rangle \xrightarrow{\nearrow\langle c_3,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_1''', lkst', mdst_1'', mem_1'\rangle$. Thus, we get from $c_{s1} = c_{s2}$ that $c_{s2} = c_{s1}'', c_{s1}'''$. Since $c_{s1}'' = c_{s2}''$, we get from the induction hypothesis that there is $c_2''', c_4 \in Com$, $mdst_2, mdst_2'' \in MdSt$, $mem_2' \in Mem$, and Λ_3 such that $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4,\emptyset,mdst_\perp\rangle\rightarrow} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$, $\vdash_{lev} \Lambda_3\{c_3\}\Lambda_3 : c_{s3}$, $\vdash_{lev} \Lambda_3\{c_4\}\Lambda_3 : c_{s4}$, $c_{s3} = c_{s4}$, and $pre(\Lambda_3) = \emptyset$.

It remains to show that there is $c_2' \in Com$, $mdst_2' \in MdSt$, such that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4, \emptyset, mdst_\perp\rangle} \langle c_2', lkst', mdst_2', mem_2'\rangle$. This follows directly from $c_2 = c_2''; c_2'''$ and $\langle c_2'', lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow\langle c_4, \emptyset, mdst_\perp\rangle} \langle c_2''', lkst', mdst_2'', mem_2'\rangle$ by the rules SQ1 and SQ2.

□

We define an equivalence on memories that requires equivalence on **low**-variables and variables for which we make a noread assumption.

Definition 10. *Let $lev : Var \to Lev$ be a domain assignment and $mdst \in MdSt$ be a mode state. Two memories $mem, mem' \in Mem$ are* **low***-equal modulo modes (denoted by: $mem =^{lev,mdst}_{\mathbf{low}} mem'$), if and only if the following condition is satisfied:*

- $\forall x \in Var.lev(x) = \mathbf{low} \wedge x \notin mdst(\mathsf{A\text{-}NR}) \implies mem(x) = mem'(x)$.

Two memories are related by $=^{lev,mdst}_{\mathbf{low}}$ if they agree on all variables of the security level **low** for which no no-read assumption is currently made.

Similar to [MSS11,Sud13] we define a compositional security property in two steps. First, we define a closure condition for binary relations that captures updates of an environment, i.e. other threads, that respect assumptions of a given thread. Second, we define a bisimulation on local configurations that defines our notion of security for individual threads in arbitrary environments that respect the assumptions of this thread.

Definition 11. *A binary relation $\mathcal{R}_{lev} \subseteq LCnf \times LCnf$ with $lev : Var \to Lev$ is* closed under globally consistent changes *if for all $c_1, c_2 \in Com$, $lkst_1, lkst_2 \in LkSt$, $mdst_1, mdst_2 \in MdSt$, and $mem_1, mem_2 \in Mem$ with*

$$\langle c_1, lkst_1, mdst_1, mem_1\rangle \mathcal{R}_{lev} \langle c_2, lkst_2, mdst_2, mem_2\rangle)$$

the following three conditions are satisfied

1. $\forall x \in Var.(lev(x) = \mathbf{high} \wedge x \notin mdst_1(\mathsf{A\text{-}NW})$
   $\implies \forall v_1, v_2 \in Val.$
   $\langle c_1, lkst_1, mdst_1, mem_1[x \mapsto v_1]\rangle \mathcal{R}_{lev} \langle c_2, lkst_2, mdst_2, mem_2[x \mapsto v_2]\rangle)$,
2. $\forall x \in Var.(lev(x) = \mathbf{low} \wedge x \notin mdst_1(\mathsf{A\text{-}NW})$
   $\implies \forall v, \in Val.$
   $\langle c_1, lkst_1, mdst_1, mem_1[x \mapsto v]\rangle \mathcal{R}_{lev} \langle c_2, lkst_2, mdst_2, mem_2[x \mapsto v]\rangle)$.

The definition of the closure condition captures updates of **high** variables (first item) and **low** variables (second item) by other threads similar to the closure conditions in [MSS11,Sud13]. Note that our definition of the closure condition only considers the mode state in the first local configuration. In the context in which we will use the closure condition we will explicitly ensure that the mode states of both local configurations are compatible (i.e. mode states agree on the assumptions).

We now define a bisimulation relation that characterizes secure information flow modulo modes.

Definition 12. *A symmetric binary relation $\mathcal{R}_{lev} \subseteq LCnf \times LCnf$ with $lev : Var \to Lev$ is a* strong low bisimulation modulo modes *if it is closed under globally consistent*

changes, and if for all $c_1, c_2 \in Com$, $lkst_1, lkst_2 \in LkSt$, $mdst_1, mdst_2 \in MdSt$, *and* $mem_1, mem_2 \in Mem$ *with*

$$\langle c_1, lkst_1, mdst_1, mem_1\rangle \mathcal{R}_{lev} \langle c_2, lkst_2, mdst_2, mem_2\rangle$$

the following conditions are satisfied

1. $lkst_1 = lkst_2$, $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$, $mem_1 =^{lev,mdst_1}_{\mathbf{low}} mem_2$,
2. *for all* $c_1' \in Com$, $lkst_1' \in LkSt$, $mdst_1' \in MdSt$, $mem_1' \in Mem$, *and* $\alpha_1 \in Eve$ *with* $\langle c_1, lkst_1, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst_1', mdst_1', mem_1'\rangle$ *there are* $c_2' \in Com$, $lkst_2' \in LkSt$, $mdst_2' \in MdSt$, $mem_2' \in Mem$, *and* $\alpha_2 \in Eve$ *such that the following conditions are satisfied:*
   (a) $\langle c_2, lkst_2, mdst_2, mem_2\rangle \xrightarrow{\alpha} \langle c_2', lkst_2', mdst_2', mem_2'\rangle$,
   (b) $\langle c_1', lkst_1', mdst_1', mem_1'\rangle \mathcal{R}_{lev} \langle c_2', lkst_2', mdst_2', mem_2'\rangle$, *and*
   (c) *if there is* $c_3 \in Com$ *such that* $\alpha_1 = \nearrow_{\langle c_3, \emptyset, mdst_\perp\rangle}$, *then there is* $c_4 \in Com$ *such that* $\alpha_2 = \nearrow_{\langle c_4, \emptyset, mdst_\perp\rangle}$ *and*

$$\langle c_3, \emptyset, mdst_\perp, mem_1'\rangle \mathcal{R}_{lev} \langle c_4, \emptyset, mdst_\perp, mem_1'\rangle \ .$$

The relation $\sim_{lev}$ *is the union of all strong low bisimulations modulo modes.*

We now show that our type system is sound with respect to our bisimulation-based security property.

Theorem 5. *If* $\vdash_{lev} \Lambda\{c\}\Lambda' : c'$ *is derivable, then*

$$\langle c, lkst, mdst_1, mem_1\rangle \sim_{lev} \langle c, lkst, mdst_2, mem_2\rangle$$

holds for all $lkst \in LkSt$, $mdst_1, mdst_2 \in MdSt$, *and* $mem_1, mem_2 \in Mem$ *with* $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$ *and* $mem_1(x) =^{lev,\Lambda}_{\mathbf{low}} mem_2(x)$.

Proof. We prove Theorem 5 in three steps. In the first step, we construct a family of binary relations on local configurations $\mathcal{R}^{\Lambda'}_{lev}$ that is parameterized by a partial type environments. In the second step, we show that

$$\langle c, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} lev \langle c, lkst, mdst_2, mem_2\rangle$$

holds. In the third step, we show that the union of all relations $\mathcal{R}^{\Lambda'}_{lev}$ in the family is a strong low bisimulation modulo modes. Since $\sim_{lev}$ is the union of all strong low bisimulation modulo modes, this suffices to show that

$$\langle c, lkst, mdst, mem_1\rangle \sim_{lev} \langle c, lkst, mdst, mem_2\rangle$$

holds.

Before we start, note that whenever $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$ holds, then

- $mdst_1 \in comp(lev, \Lambda)$ holds if and only if $mdst_2 \in comp(lev, \Lambda)$ holds, and
- $x \in mdst_1(md)$ holds iff $x \in mdst_2(md)$ holds for $md \in \{\mathsf{A\text{-}NR}, \mathsf{A\text{-}NW}\}$.

Hence, we only need to check for these properties in one of two mode states when the two mode states fulfill $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$.

Step 1: Constructing the family of relations. We define

$$\mathcal{R}^{\Lambda'}_{lev} = \left\{ (\langle c_1, lkst, mdst_1, mem_1\rangle, \langle c_2, lkst, mdst_2, mem_2\rangle) \;\middle|\; \begin{array}{l} \exists \Lambda. \\ \vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1} \wedge \vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2} \\ \wedge c_{s1} = c_{s2} \wedge mdst_1, mdst_2 \in comp(lev, \Lambda) \\ \wedge mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2 \\ \wedge mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2 \end{array} \right\}$$

Step 2: Showing that $\langle c, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} lev \langle c, lkst, mdst_2, mem_2\rangle$ *holds.* By the assumption $\vdash_{lev} \Lambda\{c\}\Lambda' : c'$ of the theorem we get directly that

$$\langle c, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} \langle c, lkst, mdst_2, mem_2\rangle$$

holds for all $lkst \in LkSt$, $mdst_1, mdst_2 \in MdSt$, and $mem_1, mem_2 \in Mem$ with $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$, and $mem_1(x) =^{lev,\Lambda}_{\mathbf{low}} mem_2(x)$.

Step 3: Showing that $\mathcal{R}^{\Lambda'}_{lev}$ *is a strong low bisimulation modulo modes.* It is clear from the definition that the family of relations $\mathcal{R}^{\Lambda'}_{lev}$ is symmetric, because all conditions are symmetric.

We show that $\mathcal{R}^{\Lambda'}_{lev}$ is closed under globally consistent changes. Hence, let $\langle c_1, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} \langle c_2, lkst, mdst_2, m$ According to the definition of "closed under globally consistent changes" (Definition 11), we must show that for all $x \in Var$ with $x \notin mdst_1(\mathsf{A\text{-}NW})$ the following two conditions hold:

1. If $lev(x) = \mathbf{low}$,
   then $\langle c_1, lkst, mdst_1, mem_1[x \mapsto v]\rangle \mathcal{R}^{\Lambda_1,\Lambda_2}_{lev} \langle c_2, lkst, mdst_2, mem_2[x \mapsto v]\rangle$ holds for all $v \in Val$, and
2. If $lev(x) = \mathbf{high}$,
   then $\langle c_1, lkst, mdst_1, mem_1[x \mapsto v_1]\rangle \mathcal{R}^{\Lambda_1}_{lev} \langle c_2, lkst, mdst_2, mem_2[x \mapsto v_2]\rangle$ holds for all $v_1, v_2 \in Val$.

Let Λ be a partial type environment that has the properties required in the definition of $\mathcal{R}^{\Lambda'}_{lev}$, i.e. $mdst_1, mdst_2 \in comp(lev, \Lambda)$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$.

We must show that the memories are still related by $=^{lev,\Lambda}_{\mathbf{low}}$ after the modification of the variables. For condition (1) this is immediate, because the variables are set to equal values on both sides of the relation.

For condition (2), we know that $lev(x) = \mathbf{high}$. Hence, $\Lambda_{lev}\langle x\rangle = \mathbf{low}$ only if $x \in pre(\Lambda)$.

Since $mdst_1 \in comp(lev, \Lambda)$, this would mean that $x \in mdst(\mathsf{A\text{-}NW})$. This contradicts the assumption that $x \notin mdst(\mathsf{A\text{-}NW})$. Hence, $mem_1[x \mapsto v_1] =^{lev,\Lambda'_1}_{\mathbf{low}} mem_2[x \mapsto v_2]$ also holds for condition (2).

Now we show that whenever $\langle c_1, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} \langle c_2, lkst, mdst_2, mem_2\rangle$, then $mem_1 =^{lev,mdst_1}_{\mathbf{low}} mem_2$. Let Λ be a partial type environment with the properties stated in the definition of $\mathcal{R}^{\Lambda'}_{lev}$, i.e. $mdst_1, mdst_2 \in comp(lev, \Lambda)$ and $mem_1 =^{lev,\Lambda}_{\mathbf{low}} mem_2$. To show that $mem_1 =^{lev,mdst_1}_{\mathbf{low}} mem_2$ holds, assume $lev(x) = \mathbf{low}$ and $x \notin mdst_1(\mathsf{A\text{-}NR})$. Since $mdst_1 \in comp(lev, \Lambda)$ according to the definition of $\mathcal{R}^{\Lambda'}_{lev}$, we have $x \notin pre(\Lambda)$. Hence, $\Lambda'_{1\,lev}\langle x\rangle = \mathbf{low}$ and, thus, $mem_1(x) = mem_2(x)$ follows directly from $mem_1 =^{lev_1,\Lambda}_{\mathbf{low}} mem_2$.

We finally show that whenever $\langle c_1, lkst, mdst_1, mem_1\rangle \mathcal{R}^{\Lambda'}_{lev} \langle c_2, lkst, mdst_2, mem_2\rangle$ and $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c'_1, lkst', mdst'_1, mem'_1\rangle$, then there is $c'_2 \in Com$, $mdst'_2$, $\alpha' \in Eve$, and $mem'_2 \in Mem$ such that the following three conditions hold:

1. $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$,
2. $\langle c_1', lkst', mdst_1', mem_1'\rangle \mathcal{R}_{lev}^{\Lambda'} \langle c_2', lkst', mdst_2', mem_2'\rangle$, and
3. if there is $c_3 \in Com$ such that $\alpha = \nearrow_{\langle c_3, \emptyset, mdst_\perp\rangle}$, then there is Λ'' and $c_4 \in Com$ such that $\alpha' = \nearrow_{\langle c_4, \emptyset, mdst_\perp\rangle}$ and

$$\langle c_3, \emptyset, mdst_\perp, mem_1'\rangle \mathcal{R}_{lev}^{\Lambda''} \langle c_4, \emptyset, mdst_\perp, mem_1'\rangle \ .$$

From $\langle c_1, lkst, mdst_1, mem_1\rangle \mathcal{R}_{lev}^{\Lambda'} \langle c_2, lkst, mdst_2, mem_2\rangle$, we know by the definition of $\mathcal{R}_{lev}^{\Lambda'}$ that $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}$, $\vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}$, $c_{s1} = c_{s2}$, $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2$, and $mem_1 =_{\mathbf{low}}^{lev,\Lambda} mem_2$.

Assume that $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_1', mem_1'\rangle$.

From $\vdash_{lev} \Lambda\{c_1\}\Lambda' : c_{s1}, \vdash_{lev} \Lambda\{c_2\}\Lambda' : c_{s2}, mdst_1, mdst_2 \in comp(lev, \Lambda), mdst_1 =_{\{\text{A-NR,A-NW}\}} mdst_2, mem_1 =_{\mathbf{low}}^{lev,\Lambda} mem_2, c_{s1} = c_{s2}$, and $\langle c_1, lkst, mdst_1, mem_1\rangle \xrightarrow{\alpha} \langle c_1', lkst', mdst_2', mem_1'\rangle$ we get by Lemma 11 that there is $mdst_2' \in MdSt$, $\alpha' \in Eve$, $c_2', c_{s1}', c_{s2}' \in Com$, $mem_2' \in Mem$, and Λ'' such that $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\alpha'} \langle c_2', lkst', mdst_2', mem_2'\rangle$, $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : c_{s1}'$, $\vdash_{lev} \Lambda''\{c_2'\}\Lambda' : c_{s2}'$, $c_{s1}' = c_{s2}'$, $mdst_1', mdst_2' \in comp(lev, \Lambda'')$, $mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$, and $mem_1' =_{\mathbf{low}}^{lev,\Lambda''} mem_2'$.

From $\vdash_{lev} \Lambda''\{c_1'\}\Lambda' : c_{s1}', \vdash_{lev} \Lambda''\{c_2'\}\Lambda' : c_{s2}', c_{s1}' = c_{s2}', mdst_1', mdst_2' \in comp(lev, \Lambda''), mdst_1' =_{\{\text{A-NR,A-NW}\}} mdst_2'$, and $mem_1' =_{\mathbf{low}}^{lev,\Lambda''} mem_2'$, we get by the definition of $\mathcal{R}_{lev}^{\Lambda'}$ that $\langle c_1', lkst', mdst_1', mem_1'\rangle \mathcal{R}_{lev}^{\Lambda'} \langle c_2', lkst', mdst_2', mem_2'\rangle$.

Hence, the first and the second condition are fulfilled. It remains to show that the third condition, i.e. if there is $c_3 \in Com$ such that $\alpha = \nearrow_{\langle c_3, \emptyset, mdst_\perp\rangle}$, then there is $c_4 \in Com$ and Λ''' such that $\alpha' = \nearrow_{\langle c_4, \emptyset, mdst_\perp\rangle}$ and

$$\langle c_3, \emptyset, mdst_\perp, mem_1'\rangle \mathcal{R}_{lev}^{\Lambda'''} \langle c_4, \emptyset, mdst_\perp, mem_1'\rangle \ .$$

Hence, assume that there is $c_3 \in Com$ such that $\alpha = \nearrow_{\langle c_3, \emptyset, mdst_\perp\rangle}$. By Lemma 12 we get that there is $c_2', c_4 \in Com$, $lkst'' \in LkSt$, $mdst_2' \in MdSt$, $mem_2' \in Mem$, and Λ''' such that $\alpha' = \nearrow_{\langle c_4, \emptyset, mdst_\perp\rangle}$, $\langle c_2, lkst, mdst_2, mem_2\rangle \xrightarrow{\nearrow_{\langle c_4, \emptyset, mdst_\perp\rangle}} \langle c_2', lkst'', mdst_2', mem_2'\rangle$. $\vdash_{lev} \Lambda'''\{c_3'\}\Lambda''' : c_{s3}$, $\vdash_{lev} \Lambda'''\{c_4'\}\Lambda''' : c_{s4}$, $c_{s3} = c_{s4}$, and $pre(\Lambda''') = \emptyset$. From $pre(\Lambda''') = \emptyset$ we get that $mdst_\perp \in comp(lev, \Lambda''')$. From $mem_1' = mem_1'$ we get that $mem_1' =_{\mathbf{low}}^{lev,\Lambda'''} mem_1'$. Thus,

$$\langle c_3, \emptyset, mdst_\perp, mem_1'\rangle \mathcal{R}_{lev}^{\Lambda'''} \langle c_4, \emptyset, mdst_\perp, mem_1'\rangle$$

holds. □

A.4 Soundness of the Combined Analyses

In this subsection, we prove the soundess of our combined analysis. This includes the proof for the security type system with respect to termination-sensitive noninterference (Theorem 3), as well as the concrete combination of our analyses (Corollary 1). The following table lists the dependencies between lemmas and theorems in this subsection.

Lemma/Theorem	Depends on lemmas/theorems
Lemma 13	none
Lemma 14	Lemma 13
Theorem 3	Lemma 14, Lemma 11, Theorem 5
Corollary 1	Theorems 1, 2, 3

Definition 13. *A command c does not read variable x, if for all c', $lkst$, $lkst'$, $mdst$, $mdst'$, mem, mem', and α with $\langle c, lkst, mdst, mem\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem'\rangle$ one of the following two conditions is satisfied:*

- $\forall Val.\langle c, lkst, mdst, mem[x \mapsto v]\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem'[x \mapsto v]\rangle$, *or*
- $\forall Val.\langle c, lkst, mdst, mem[x \mapsto v]\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem'\rangle$.

Note that in a local configuration $\langle c, lkst, mdst, mem\rangle$ the command c does not read any variable for which it provides a no-read guarantee, because the conditions in the definition of "does not read" and "provide its no-read guarantees" coincide.

First we prove that a command that does not read some variables is not influenced by said variables.

Lemma 13. *Let $\langle c, lkst, mdst, mem_1\rangle, \langle c', lkst', mdst', mem_1'\rangle \in LCnf$ be local configurations, $\alpha \in Eve$ be an event, $x_1, \ldots, x_k \in Var$ be variables, and $mem_2 \in Mem$ be a memory.*

If c does not read x_i for all $i \in \{1, \ldots, k\}$,
$\langle c, lkst, mdst, mem_1\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem_1'\rangle$, $mem_1(x) = mem_2(x)$ for all $x \in Var \setminus \{x_1, \ldots, x_k\}$, then there is a memory $mem_2' \in Mem$ such that
$\langle c, lkst, mdst, mem_2\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem_2'\rangle$ and $mem_1'(x) = mem_2'(x)$ for all $x \in Var \setminus \{x_1, \ldots, x_k\}$.

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

Proof. We prove this lemma by induction on the number of variables k. Let firstly $k = 0$. In this case, we have $mem_1 = mem_2$ and we conclude by setting $mem_2' = mem_1'$.

Now let $k > 0$. Let $mem_3 = mem_2[x_k \mapsto mem_1(x_k)]$. This means, mem_3 and mem_1 differ only in the variables in $\{x_1, \ldots, x_{k-1}\}$. Hence, we get from the induction hypothesis that there is mem_3' with

$$\langle c, lkst, mdst, mem_3\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem_3'\rangle$$

and mem_1' and mem_3' only differ in the variables 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 is v such that $mem_3 = mem_2[x_k \mapsto v)]$. Since c does not read x_k one of the following conditions holds:

- $\langle c, lkst, mdst, mem_2\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem_3'[x_k \mapsto v]\rangle$
- $\langle c, lkst, mdst, mem_2\rangle \xrightarrow{\alpha} \langle c', lkst', mdst', mem_3'\rangle$

In the first case, we define $mem_2' = mem_3'[x_k \mapsto v]$, and in the second case, we define $mem_2' = mem_3'$. Hence, mem_2' and mem_3' differ at most in x_k. Moreover, if $mem_2(x_k) \neq mem_2'(x_k)$, then $mem_2'(x_k) \neq v$. Hence, it must be the second case, thus, $mem_2'(x_k) = mem_3'(x_k)$. Finally, if $mem_3(x_k) \neq mem_3'(x_k)$, we can turn around the reasoning and consider $mem_3 = mem_2[x_k \mapsto v']$ to conclude.

Hence, mem_1' and mem_2' differ only on the variables in $\{x_1, \ldots, x_k\}$ and they do not differ on those variables in $\{x_1, \ldots, x_k\}$ that have been written in one of the execution steps.

We now show that for two global configurations in which each pair of configurations is related by our bisimulation, all steps of the first configuration can be matched by the second thread, and in the resulting global configurations each pair of configurations is again related by our bisimulation relation.

Lemma 14. *Let* $gcnf_1 = \langle[(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, (c_{1,n}, lkst_n, mdst_{1,n})], mem_1\rangle$ *and* $gcnf_2 = \langle[(c_{2,1}, lkst_1, mdst_{2,1}), \ldots, (c_{2,n}, lkst_n, mdst_{2,n})], mem_2\rangle$ *be two global configurations that use modes globally sound, provide sound guarantees, and that satisfy* $\forall i, j.i \neq j \implies lkst_i \cap lkst_j = \emptyset$.

If there is a global configuration

$$gcnf'_1 = \langle[(c'_{1,1}, lkst'_1, mdst'_{1,1}), \ldots, (c'_{1,m}, lkst'_m, mdst_{1,m})'], mem'_1\rangle$$

such that $gcnf_1 \twoheadrightarrow gcnf'_1$ *and there exist* $mem_{1,i}, mem_{2,i} \in Mem$ *for all* $i \in \{1, \ldots, n\}$ *with*

- $\langle c_{1,i}, lkst_i, mdst_{1,i}, mem_{1,i}\rangle \sim_{lev} \langle c_{2,i}, lkst_i, mdst_{2,i}, mem_{2,i}\rangle$, *and*
- $mem_{1,i}(x) = mem_1(x)$ *and* $mem_{2,i}(x) = mem_2(x)$ *hold for all* x *with* $(lev(x) = \textbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{j\in\{1,\ldots,n\}} mdst_{1,j}(\mathsf{A\text{-}NR})$,

then there exist $gcnf'_2$, $c'_{2,1}, \ldots, c'_{2,m}$, $mdst'_{2,1}, \ldots, mdst'_{2,m}$, *and* mem'_2 *with* $gcnf'_2 = \langle[(c'_{2,1}, lkst'_1, mdst'_{2,1}), \ldots, (c'_{2,m}, lkst'_m, mdst'_{2,m})], mem'_2\rangle$ *such that*

1. $gcnf_2 \twoheadrightarrow gcnf'_2$, *and*
2. *for all* $i \in \{1, \ldots, m\}$ *there are* $mem'_{1,i}, mem'_{2,i} \in Mem$ *with*
   - $\langle c'_{1,i}, lkst'_i, mdst'_{1,i}, mem'_{1,i}\rangle \sim_{lev} \langle c'_{2,i}, lkst'_i, mdst'_{2,i}, mem'_{2,i}\rangle$, *and*
   - $mem'_{1,i}(x) = mem_1(x)$ *and* $mem'_{2,i}(x) = mem_2(x)$ *hold for all* x *with* $(lev(x) = \textbf{high}) \vee mem'_1(x) = mem'_2(x) \vee x \notin \bigcup_{j\in\{1,\ldots,m\}} mdst'_{1,j}(\mathsf{A\text{-}NR})$, *and*
3. $\forall i, j.i \neq j \implies lkst'_i \cap lkst'_j = \emptyset$.

Proof. We prove this lemma in two steps. In the first step, we construct a global configuration $gcnf'_2$ such that conclusion (1) and (3) from the lemma is satisfied, i.e. $gcnf_2 \twoheadrightarrow gcnf'_2$. In the second step, we prove that the global configuration $gcnf'_2$ also satisfies conclusion (2) of this lemma.

Step 1 (Constructing $gcnf'_2$*).* We show that the execution step $gcnf_1 \twoheadrightarrow gcnf'_1$ can be matched by an execution step in $gcnf_2$. From $gcnf_1 \twoheadrightarrow gcnf'_1$,

$$gcnf_1 = \langle[(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, (c_{1,n}, lkst_n, mdst_{1,n})], mem_1\rangle$$

and $gcnf'_1 = \langle[(c'_{1,1}, lkst'_1, mdst'_{1,1}), \ldots, (c'_{1,m}, lkst'_m, mdst_{1,m})'], mem'_1\rangle$ we get by the definition of the global transition system that either $m = n$ or $m = n + 1$ and there is some $j, j' \in \{1, \ldots, n\}$ and $\alpha \in Eve$ with $n - j' = j$

(A) $\langle c_{1,j}, lkst_j, mdst_{1,j}, mem_1\rangle \xrightarrow{\alpha} \langle c'_{1,m-j'}, lkst'_{m-j'}, mdst'_{1,m-j'}, mem'_1\rangle$
(B) $(c_{1,i}, lkst_i, mdst_{1,i}) = (c'_{1,i}, lkst'_i, mdst'_{1,i})$ for all $i < j$
(C) $(c_{1,n-i}, lkst_{n-i}, mdst_{1,n-i}) = (c'_{1,m-i}, lkst'_{m-i}, mdst'_{1,m-i})$ for all $i < j'$
(D) if $m = n + 1$, then $(c'_{1,j}, lkst'_j, mdst'_{1,j}) = (c_3, \emptyset, mdst_\perp)$ and $\alpha = \nearrow_{\langle c_3,\emptyset,mdst_\perp\rangle}$

By assumption of this lemma there are $mem_{1,j}, mem_{2,j} \in Mem$ such that for all $x \in Var$ we have

(E) $\langle c_{1,j}, lkst_j, mdst_{1,j}, mem_{1,j}\rangle \sim_{lev} \langle c_{2,j}, lkst_j, mdst_{2,j}, mem_{2,j}\rangle$, and
(F) $[(lev(x) = \textbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,n\}\setminus\{j\}} mdst_{1,k}(\mathsf{A\text{-}NR})]$
$\implies mem_{1,j}(x) = mem_1(x)$, and
(G) $[(lev(x) = \textbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,n\}\setminus\{j\}} mdst_{1,k}(\mathsf{A\text{-}NR})]$
$\implies mem_{2,j}(x) = mem_2(x)$.

From (E) we get by definition of strong low bisimulation modulo modes that

(H) $mem_{1,j} =^{lev,mdst_{1,j}}_{\mathbf{low}} mem_{2,j}$.

Due to (F), $mem_{1,j}$ and mem_1 differ only in variables x with $lev(x) = \mathbf{low}$, $mem_1(x) \neq mem_2(x)$, and $x \in mdst_{1,k}(\mathsf{A\text{-}NR})$ for some $k \neq j$. As by the assumption of this lemma, $gcnf_1$ uses modes globally sound, we have $x \in mdst_{1,j}(\mathsf{G\text{-}NR})$ for these variables. Moreover, by assumption of this lemma, $gcnf_1$ provides its guarantees and, hence, $c_{1,j}$ does not read the variables whose values differ in mem_1 and $mem_{1,j}$. We may hence apply Lemma 13 for the transition in (A) and the memory $mem_{1,j}$. This yields a memory $mem'_{1,m-j'}$ with

(I) $\langle c_{1,j}, lkst_j, mdst_{1,j}, mem_{1,j}\rangle \xrightarrow{\alpha} \langle c'_{1,m-j'}, lkst'_{m-j'}, mdst'_{1,m-j'}, mem'_{1,m-j'}\rangle$
(J) $[(lev(x) = \mathbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,n\}\setminus\{j\}} mdst_{1,k}(\mathsf{A\text{-}NR})]$
$\implies mem'_{1,m-j'}(x) = mem'_1(x)$ for all $x \in Var$.

From (E) we get due to (I) that there is $mdst_{2,m-j'}$, $c'_{2,m-j'}$, $\alpha' \in Eve$, $mem'_{2,m-j'} \in Mem$ with

(K) $\langle c_{2,j}, lkst_j, mdst_{2,j}, mem_{2,j}\rangle \xrightarrow{\alpha'} \langle c'_{2,m-j'}, lkst'_{m-j'}, mdst'_{2,m-j'}, mem'_{2,m-j'}\rangle$
(L)
$$\langle c'_{1,m-j'}, lkst'_{m-j'}, mdst'_{1,m-j'}, mem'_{1,m-j'}\rangle$$
$$\sim_{lev}$$
$$\langle c'_{2,m-j'}, lkst'_{m-j'}, mdst'_{2,m-j'}, mem'_{2,m-j'}\rangle$$
(M) if $\alpha = \nearrow_{\langle c_3,\emptyset,mdst_\perp\rangle}$, then there is c_4 such that $\alpha = \nearrow_{\langle c_4,\emptyset,mdst_\perp\rangle}$ and $\langle c'_{1,j}, lkst'_j, mdst'_{1,j}, mem'_{1,j}\rangle \sim_{lev} \langle c'_{2,j}, lkst'_j, mdst'_{2,j}, mem'_{2,j}\rangle$ with $c'_{1,j} = c_3$, $c'_{2,j} = c_4$, $lkst'_j = \emptyset$, $mdst'_{1,j} = mdst'_{2,j} = mdst_\perp$, and $mem'_{2,j} = mem'_{1,j} = mem'_{1,m-j'}$.

From (L) we get by the definition of strong low bisimulation modulo modes that

(N) $mdst_{1,m-j'} =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_{2,m-j'}$, and
(O) $mem'_{1,m-j'} =^{lev,mdst_{1,m-j'}}_{\mathbf{low}} mem'_{2,m-j'}$.

Due to (G), we may exploit globally sound use of modes and providing sound guarantees as before to apply Lemma 13 (as before) for the transition in (K) and the memory mem_2. This yields a memory mem'_2 such that

(P) $\langle c_{2,j}, lkst_j, mdst_{2,j}, mem_2\rangle \xrightarrow{\alpha'} \langle c'_{2,m-j'}, lkst'_{m-j'}, mdst'_{2,m-j'}, mem'_2\rangle$
(Q) $[(lev(x) = \mathbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,n\}\setminus\{j\}} mdst_{1,k}(\mathsf{A\text{-}NR})]$
$\implies mem'_{2,j}(x) = mem'_2(x)$ for all $x \in Var$.

That means, we have now constructed $c_{2,m-j'}$, $mdst_{2,m-j'}$ and mem'_2. It remains to construct $c'_{2,i}$, $lkst'_i$ and $mdst'_{2,i}$ for $i \neq (m - j')$. To this end, we define

(R) $c'_{2,i} = c_{2,i}$, $lkst'_i = lkst_i$, and $mdst'_{2,i} = mdst_{2,i}$ for $i < j$
(S) $c'_{2,m-i} = c_{2,m-i}$, $lkst'_{m-i} = lkst_{m-i}$, and $mdst'_{2,m-i} = mdst_{2,m-i}$ for $i < j'$.

Now assume that $m = n$ and $\alpha \neq l$, then we get by the definition of the global transition system that $gcnf_2 \twoheadrightarrow gcnf'_2$ with
$gcnf'_2 = \langle[(c'_{2,1}, lkst'_1, mdst'_{2,1}), \ldots, (c'_{2,m}, lkst'_m, mdst'_{2,m})], mem'_2\rangle$. Since $\alpha \neq l$, we get from the definition of the local transition system (Figure 2) that

$lkst_{m-j'} \subseteq lkst_j$. Hence, $\forall i,k.i \neq k \implies lkst'_i \cap lkst'_k$ follows directly from $\forall i,k.i \neq k \implies lkst_i \cap lkst_k$.

Now assume that $m = n$ and $\alpha = l$, then we get from $gcnf_1 \twoheadrightarrow gcnf'_1$ by the definition of the global transition system that
$l \notin locks([(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, c_{1,n}, lkst_n, mdst_{1,n}])$. From this we get by the definition of the global transition system that $gcnf_2 \twoheadrightarrow gcnf'_2$ with $gcnf'_2 = \langle[(c'_{2,1}, lkst'_1, mdst'_{2,1}), \ldots, (c'_{2,m}, lkst'_m, mdst'_{2,m})], m$
Since $\alpha = l$, we get from the definition of the local transition system (Figure 2) that $lkst_{m-j'} = lkst_j \cup \{l\}$. Since $l \notin locks([(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, c_{1,n}, lkst_n, mdst_{1,n}])$ we get from $lkst_{m-j'} = lkst_j \cup \{l\}$ and $lkst'_i = lkst_i$ for all $i < j$ and $lkst'_{m-i} = lkst_{n-i}$ for all $i < j'$ and $\forall i,k.i \neq k \implies lkst_i \cap lkst_k$ that $\forall i,k.i \neq k \implies lkst'_i \cap lkst'_k$.

Now assume that $m = n + 1$. We further define $c'_{2,j} = c_4$, $lkst'_j = \emptyset$, and $mdst'_{2,j} = mdst_\perp$ and get from the definition of the global transition system that $gcnf_2 \twoheadrightarrow gcnf'_2$ with
$gcnf'_2 = \langle[(c'_{2,1}, lkst'_1, mdst'_{2,1}), \ldots, (c'_{2,m}, lkst'_m, mdst'_{2,m})], mem'_2\rangle$. In this case, we have $\alpha = \nearrow \langle c_3, \emptyset, mdst_\perp\rangle$ and $\alpha = \nearrow \langle c_4, \emptyset, mdst_\perp\rangle$. Hence, we get by the definition of the local transition system (Figure 2) that $lkst_j = lkst'_{m-j'}$. From $lkst_i = lkst'_i$ for all $i < j$ and $lkst_{n-i} = lkst_{m-i}$ for all $i < j'$ and $lkst_j = lkst'_{m-j'}$ and $lkst'_j = \emptyset$ and $\forall i,k.i \neq k \implies lkst_i \cap lkst_k$ that $\forall i,k.i \neq k \implies lkst'_i \cap lkst'_k$ holds.

Step 2 (Showing that $gcnf'_2$ satisfies conclusion (2)). In this step, we show that for all $i \in \{1, \ldots, m\}$ there are memories $mem'_{1,i}, mem'_{2,i} \in Mem$ with $mem'_1(x) = mem'_{1,i}(x)$ and $mem'_2(x) = mem'_{2,i}(x)$ for all x with $(lev(x) = \mathbf{high}) \vee mem'_1(x) = mem'_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,m\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, and
$\langle c'_{1,i}, lkst'_i, mdst'_{1,i}, mem'_{1,i}\rangle \sim_{lev} \langle c'_{2,i}, lkst'_i, mdst'_{1,i}, mem'_{2,i}\rangle$.

We distinguish four cases $i < j$, $i = m - j'$, $i > m - j'$, and $i = j$ where (j is the index of of the command performing the execution step, and $j' = n - j$ is the index counted from the end of the command performing the execution step as exhibited in Step 1).

Case (i=$m - j'$): The memories $mem'_{1,m-j'}$ and $mem'_{2,m-j'}$ have already been constructed in Step 1. In (L), we have already established that

$$\langle c'_{1,m-j'}, lkst'_{m-j'}, mdst'_{1,m-j'}, mem'_{1,m-j'}\rangle \sim_{lev} \langle c'_{2,m-j'}, lkst'_{m-j'}, mdst'_{2,m-j'}, mem'_{2,m-j'}\rangle.$$

It remains to show that for all x with $(lev(x) = \mathbf{high}) \vee mem'_1(x) = mem'_2(x) \vee x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{m-j'\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, we have $mem'_1(x) = mem'_{1,m-j'}(x)$ and $mem'_2(x) = mem'_{2,m-j'}(x)$.
Assume first that $lev(x) = \mathbf{high}$. Then $mem'_1(x) = mem'_{1,i}(x)$ and $mem'_2(x) = mem'_{2,i}(x)$ follow directly from (J) and (Q).
Assume now that $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{m-j'\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. For all $k \neq (m - j')$ and $k \neq j$ we have $mdst_{1,k} = mdst'_{1,k}$ according to (B) and (C). If $m = n + 1$ (and hence $m - j' \neq j$), we have $mdst'_j = mdst_\perp$ according to (D). Hence, $\bigcup_{k\in\{1,\ldots,m\}\setminus\{m-j'\}} mdst'_{1,k}(\mathsf{A\text{-}NR}) = \bigcup_{k\in\{1,\ldots,n\}\setminus\{j\}} mdst_{1,k}(\mathsf{A\text{-}NR})$. Thus, $mem'_1(x) = mem'_{1,i}(x)$ and $mem'_2(x) = mem'_{2,i}(x)$ follow directly from (J) and (Q).
Assume finally that $mem'_1(x) = mem'_2(x)$. If $mem_1(x) = mem_2(x)$ also holds, then $mem'_1(x) = mem'_{1,i}(x)$ and $mem'_2(x) = mem'_{2,i}(x)$ follow directly from (J) and (Q). Hence, assume that $mem_1(x) \neq mem_2(x)$. This means, that the execution step from (A) or (I) has modified x. Since both execution steps have been obtained

with Lemma 13, we get that x is modified to the same value in $mem_{1,m-j'}$ and mem_1 respectively $mem_{2,m-j'}$ and mem_2. This concludes this case.

Case ($i < j$): We define the memories $mem'_{1,i}$ and $mem'_{2,i}$ as follows for all $x \in Var$:

(T) If $lev(x) = \textbf{high}$ or $mem'_1(x) = mem'_2(x)$ or
$x \notin \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, then
$mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$.

(U) Otherwise, i.e. $lev(x) = \textbf{low}$, $mem'_1(x) \neq mem'_2(x)$, and
$x \in \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$,
$mem'_{1,i}(x) = mem_{1,i}(x)$ and $mem'_{2,i}(x) = mem_{2,i}(x)$.

We first show that $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$ hold for all all x with $lev(x) = \textbf{high}$ or $mem'_1(x) = mem'_2(x)$ or $x \notin \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. From the properties assumed for x we get by (T) that $mem'_{1,i}(x) = mem'_1$ and $mem'_{2,i}(x) = mem'_2$ hold directly.

We now show that $\langle c'_{1,i}, lkst'_i, mdst'_{1,i}, mem'_{1,i}\rangle \sim_{lev} \langle c'_{2,i}, lkst'_i, mdst'_{2,i}, mem'_{2,i}\rangle$. Since $i < j$, we have $c'_{1,i} = c_{1,i}$, $c'_{2,i} = c_{2,i}$, $lkst'_i = lkst_i$, $mdst'_{1,i} = mdst_{1,i}$, and $mdst'_{2,i} = mdst_{2,i}$. Hence, we need to show that
$\langle c_{1,i}, lkst_i, mdst_{1,i}, mem'_{1,i}\rangle \sim_{lev} \langle c_{2,i}, lkst_i, mdst_{2,i}, mem'_{2,i}\rangle$. According to the assumptions of this Lemma, we have
$\langle c_{1,i}, lkst_i, mdst_{1,i}, mem_{1,i}\rangle \sim_{lev} \langle c_{2,i}, lkst_i, mdst_{2,i}, mem_{2,i}\rangle$. Since $\sim_{lev}$ is closed under globally consistent changes, it suffices to show that $mem'_{1,i}$ and $mem'_{2,i}$ can be obtained from $mem_{1,i}$ and $mem_{2,i}$, respectively, using the closure condition for globally consistent changes.

Note that due to the definition in (T) and (U), $mem'_{1,i}(x) \neq mem_{1,i}(x)$ or $mem'_{2,i}(x) \neq mem_{2,i}(x)$ only hold if $lev(x) = \textbf{high}$ or $mem'_1(x) = mem'_2(x)$ or $x \notin \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. We further only need to consider cases in which one of the three conditions holds. Moreover, due to (T) we have $mem'_{1,i}(x) = mem_1(x)$ and $mem'_{2,i}(x) = mem_2(x)$ in these cases.

First, consider a variable x with $x \notin mdst_{1,i}(\mathsf{A\text{-}NW})$. If $lev(x) = \textbf{high}$, then the new values of x can be set by global consistent changes, because global consistent changes allow modifying **high** variables to arbitrary values. If $lev(x) = \textbf{low}$, we get by the assumptions from the previous paragraph that $mem'_1(x) = mem'_2(x)$ or $x \notin \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. If $x \notin \bigcup_{k\in\{1,\dots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, this means in particular that $x \notin mdst'_{1,j}(\mathsf{A\text{-}NR})$. Hence, by (O) we get that $mem'_{1,m-j'}(x) = mem'_{2,m-j'}(x)$, and hence by (J) and (Q) we get $mem'_1(x) = mem'_2(x)$. Since global consistent changes allows modifying these variables to identical values, we can conclude this case.

Now consider a variable x with $x \in mdst_{1,i}(\mathsf{A\text{-}NW})$. This means in particular, that $x \in mdst_{2,i}(\mathsf{A\text{-}NW})$ also holds due to $mdst_{1,i} =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_{2,i}$. From the assumption that the global configurations $gcnf_1$ and $gcnf_2$ use modes globally sound, we get that $x \in mdst_{1,j}(\mathsf{G\text{-}NW})$ and $x \in mdst_{2,j}(\mathsf{G\text{-}NW})$. Since $gcnf_1$ and $gcnf_2$ also provide sound guarantees, $c_{1,j}$ and $c_{2,j}$ do not write x. Hence, due to the definition of "does not write", we get $mem_1(x) = mem'_1(x)$ and $mem_2(x) = mem'_2(x)$.

Assume now that $mem_1(x) = mem_{1,j}(x)$, and $mem_2(x) = mem_{2,j}(x)$. Then $mem'_{1,m-j'}(x) = mem_{1,j}(x)$ and $mem_{2,m-j'}(x) = mem_{2,j}(x)$, due to the fact that x is not written (established in the previous paragraph).

Assume now contrarily that $mem_1(x) \neq mem_{1,j}(x)$, or $mem_2(x) \neq mem_{2,j}(x)$. Then we have from the assumptions of this Lemma that $lev(x) = \textbf{low}$, $mem_1(x) \neq mem_2(x)$, and $x \in \bigcup_{k\in\{1,\dots,n\}\setminus\{i\}} mdst_{1,k}(\mathsf{A\text{-}NR})$. If $mem'_1(x) = mem'_2(x)$, this

would contradict that $c_{1,j}$ and $c_{2,j}$ do not write x (established two paragraphs before). Hence, assume $mem'_1(x) \neq mem'_2(x)$. Due to the assumption above this implies that $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. However, since $x \in \bigcup_{k\in\{1,\ldots,n\}\setminus\{i\}} mdst_{1,k}(\mathsf{A\text{-}NR})$ and all mode states except for the mode state $mdst_j$ and possibly the new mode state $mdst'_j$ do not change during the execution step, and if there is a new mode state $mdst'_j$, then $mdst'_j = mdst_\perp$, we get that $x \in mdst_{1,j}(\mathsf{A\text{-}NR})$ and $x \notin mdst_{1,m-j'}(\mathsf{A\text{-}NR})$. This contradicts that $mem_{1,j} =^{lev,mdst_{1,j}}_{\mathbf{low}} mem_{2,j}$ and $mem'_{1,m-j'} =^{lev,mdst'_{1,m-j'}}_{\mathbf{low}} mem'_{2,m-j'}$ while $c_{1,j}$ and $c_{2,j}$ do not write x. Hence, we know that $mem_{1,i}(x) = mem'_1(x)$ and $mem_{2,i}(x) = mem'_2(x)$ for all variables with $x \in mdst'_{1,i}(\mathsf{A\text{-}NW})$ and, hence, we must not apply any changes.

Case ($i > m - j'$): This case is analogous to the case $i < j$, but with different indexing, i.e. whenever one encounters an i for a symbol without a prime one uses $n - j'$ and whenever one encounters an i for a symbol with a prime one uses $m - j'$.

Case ($i = j$): If $m = n$, then this case ($i = j$) coincides with the case $i = m - j'$. Hence assume that $m = n + 1$.

We define the memories $mem'_{1,j}$ and $mem'_{2,j}$ as follows for all $x \in Var$:

(V) If $lev(x) = \mathbf{high}$ or $mem'_1(x) = mem'_2(x)$ or
$x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{j\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, then
$mem'_{1,j}(x) = mem'_1(x)$ and $mem'_{2,j}(x) = mem'_2(x)$.

(W) Otherwise, i.e. $lev(x) = \mathbf{low}$, $mem'_1(x) \neq mem'_2(x)$, and
$x \in \bigcup_{k\in\{1,\ldots,m\}\setminus\{j\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$,
$mem'_{1,j}(x) = mem'_{2,j}(x) = mem'_{1,m-j'}(x)$.

We first show that $mem'_{1,j}(x) = mem'_1(x)$ and $mem'_{2,j}(x) = mem'_2(x)$ hold for all all x with $lev(x) = \mathbf{high}$ or $mem'_1(x) = mem'_2(x)$ or $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{j\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$.

From the properties assumed for x we get by (V) that $mem'_{1,j}(x) = mem'_1(x)$ and $mem'_{2,j}(x) = mem'_2(x)$ hold directly.

We now show that $\langle c'_{1,j}, lkst'_j, mdst'_{1,j}, mem'_{1,j}\rangle \sim_{lev} \langle c'_{2,j}, lkst'_j, mdst'_{2,j}, mem'_{2,j}\rangle$. Since $i = j$ and $m = n + 1$, we have $c'_{1,j} = c_3$, $c'_{2,j} = c_4$, $lkst'_j = \emptyset$, and $mdst'_{1,j} = mdst'_{2,j} = mdst_\perp$. Hence, we need to show that $\langle c_3, \emptyset, mdst_\perp, mem'_{1,j}\rangle \sim_{lev} \langle c_4, \emptyset, mdst_\perp, mem'_{2,j}\rangle$. According to (M) we have
$\langle c_3, \emptyset, mdst_\perp, mem'_{1,m-j'}\rangle \sim_{lev} \langle c_4, \emptyset, mdst_\perp, mem'_{2,m-j'}\rangle$ Since $\sim_{lev}$ is closed under globally consistent changes, it suffices to show that $mem'_{1,j}$ and $mem'_{2,j}$ can be obtained from $mem'_{1,m-j'}$.

Note that due to the definition in (V) and (W), $mem'_{1,j}(x) \neq mem'_{1,m-j'}(x)$ or $mem'_{2,j}(x) \neq mem'_{1,m-j'}(x)$ can only hold if $lev(x) = \mathbf{high}$ or $mem'_1(x) = x'_2(x)$ or $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. We further only need to consider cases in which one of the three conditions holds. Moreover, due to (V) we have $mem'_{1,j}(x) = mem'_1(x)$ and $mem'_{2,j}(x) = mem'_2(x)$ in these cases.

Since $mdst'_{1,j} = mdst_\perp$ there are no variables x with $x \in mdst'_{1,j}(\mathsf{A\text{-}NW})$, and we only need to consider variables x with $x \notin mdst'_{1,j}(\mathsf{A\text{-}NW})$. Hence, assume $x \notin mdst'_{1,j}(\mathsf{A\text{-}NW})$ holds for x. If $lev(x) = \mathbf{high}$, then the new values of x can be set by global consistent changes, because global consistent changes allow modifying **high** variables to arbitrary values. If $lev(x) = \mathbf{low}$, we get by the assumptions from the previous paragraph that $mem'_1(x) = mem'_2(x)$ or $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{j\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$. If $x \notin \bigcup_{k\in\{1,\ldots,m\}\setminus\{i\}} mdst'_{1,k}(\mathsf{A\text{-}NR})$, this means in particular that $x \notin mdst'_{1,j}(\mathsf{A\text{-}NR})$. Hence, by (O) we get that $mem'_{1,m-j'}(x) = mem'_{2,m-j'}(x)$, and hence by (J) and (Q) we get $mem'_1(x) = mem'_2(x)$. Since global consistent changes allows modifying these variables to identical values, we can conclude this case.

□

We are now ready to prove the soundness of our security type system with respect to termination-sensitive noninterference (Theorem 3).

Proof. Let lev and $c, c' \in Com$ be arbitrary such that c ensures a sound use of mdoes and $\vdash_{lev} c : c'$.

We now must show

$$\begin{array}{l}\forall \overrightarrow{ccnf}_1 \in CCnf^*.\forall mem_1, mem_1', mem_2 \in Mem.\\ \langle[(c, \emptyset, mdst_\perp)], mem_1\rangle \twoheadrightarrow^* \langle \overrightarrow{ccnf}_1, mem_1'\rangle \wedge trm(\overrightarrow{ccnf}_1) \wedge mem_1 =^{lev}_{\mathbf{low}} mem_2\\ \implies \exists \overrightarrow{ccnf}_2 \in CCnf^*.\exists mem_2' \in Mem.\\ \langle[(c, \emptyset, mdst_\perp)], mem_2\rangle \twoheadrightarrow^* \langle \overrightarrow{ccnf}_2, mem_2'\rangle \wedge trm(\overrightarrow{ccnf}_2) \wedge mem_1' =^{lev}_{\mathbf{low}} mem_2'\end{array}$$

Hence, let $\overrightarrow{ccnf}_1 \in CCnf^*$ and $mem_1, mem_2, mem_1' \in Mem$ be arbitrary such that $trm(\overrightarrow{ccnf}_1)$, $mem_1 =^{lev}_{\mathbf{low}} mem_2$, and $\langle[(c, \emptyset, mdst_\perp)], mem_1\rangle \twoheadrightarrow^* \langle \overrightarrow{ccnf}, mem_1'\rangle$. Hence, we know that there is a k such that $\langle[(c, \emptyset, mdst_\perp)], mem_1\rangle \twoheadrightarrow_k \langle \overrightarrow{ccnf}, mem_1'\rangle$. We now show that k inductive applications of Lemma 14 establish the desired result.

From $\vdash_{lev} c : c'$ we get by the rule TTH that $\vdash_{lev} \Lambda\{c\}\Lambda : c'$ with $pre(\Lambda) = \emptyset$. From $\vdash_{lev} \Lambda\{c\}\Lambda : c'$ we get by Theorem 5 that $\langle c, lkst, mdst_1, mem_1\rangle \sim_{lev} \langle c, lkst, mdst_2, mem_2\rangle$ holds for all $lkst \in LkSt$, $mdst_1, mdst_2 \in MdSt$, and $mem_1, mem_2 \in Mem$ with $mdst_1, mdst_2 \in comp(lev, \Lambda)$, $mdst_1 =_{\{\mathsf{A\text{-}NR},\mathsf{A\text{-}NW}\}} mdst_2$, and $mem_1(x) = mem_2(x)$ for all x with $lev_\Lambda\langle x\rangle = \mathbf{low}$.

From $pre(\Lambda) = \emptyset$ we get by the definition of $mdst_\perp$ and $comp(lev, \Lambda)$ that $mdst_\perp \in comp(lev, \Lambda)$.

From $pre(\Lambda) = \emptyset$ we get that $mem_1 =^{lev}_{\mathbf{low}} mem_2$ implies that $mem_1(x) = mem_2(x)$ for all x with $lev_\Lambda\langle x\rangle = \mathbf{low}$.

Hence, we have $\langle c, \emptyset, mdst_\perp, mem_1\rangle \sim_{lev} \langle c, \emptyset, mdst_\perp, mem_2\rangle$ for all $mem_1 =^{lev}_{\mathbf{low}} mem_2$.

Furthermore, since the lock state is $\emptyset$, the global configurations $\langle[c, \emptyset, mdst_\perp], mem_1\rangle$ and $\langle[c, \emptyset, mdst_\perp], mem_2\rangle$ satisfies $\forall i, j.i \neq j \implies lkst_i \cap lkst_j = \emptyset$.

Since sound use of modes is invariant under execution steps in our semantics, and the postconditions of Lemma 14 again establish the preconditions of Lemma 14 for the subsequent global configurations, we can apply Lemma 14 k times inductively to obtain that there is $\overrightarrow{ccnf}_2 \in CCnf^*$ and mem_2' such that $\langle[(c, \emptyset, mdst_\perp)], mem_2\rangle \twoheadrightarrow_k \langle \overrightarrow{ccnf}_2, mem_2'\rangle$.

It remains to show that $trm(\overrightarrow{ccnf}_2)$, and $mem_1' =^{lev}_{\mathbf{low}} mem_2'$.

From the inductive application of Lemma 14, we also get that there is $c_{1,1}, \ldots, c_{1,m}, c_{2,1}, \ldots, c_{2,m} \in Com$ and $lkst_1, \ldots, lkst_m \in LkSt$ and
$mdst_{1,1}, \ldots, mdst_{1,m}, mdst_{2,1}, \ldots, mdst_{2,m} \in MdSt$ such that
$\overrightarrow{ccnf}_1 = [(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, (c_{1,m}, lkst_m, mdst_{1,m})]$ and
$\overrightarrow{ccnf}_2 = [(c_{2,1}, lkst_1, mdst_{2,1}), \ldots, (c_{2,m}, lkst_m, mdst_{2,m})]$ and
for all $i \in \{1, \ldots, m\}$ there are $mem_{1,i}', mem_{2,i}' \in Mem$ with

- $\langle c_{1,i}, lkst_i, mdst_{1,i}, mem_{1,i}'\rangle \sim_{lev} \langle c_{2,i}, lkst_i, mdst_{2,i}, mem_{2,i}'\rangle$, and
- $mem_{1,i}'(x) = mem_1'(x)$ and $mem_{2,i}'(x) = mem_2'(x)$ hold for all x with $(lev(x) = \mathbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{j\in\{1,\ldots,n\}} mdst_{1,j}(\mathsf{A\text{-}NR})$

From $trm(\overrightarrow{ccnf}_1)$ and $\overrightarrow{ccnf}_1 = [(c_{1,1}, lkst_1, mdst_{1,1}), \ldots, (c_{1,m}, lkst_m, mdst_{1,m})]$ we get that $c_{1,i} = \mathbf{stop}$ for all $i \in \{1, \ldots, m\}$. From $\langle c_{1,i}, lkst_i, mdst_{1,i}, mem_{1,i}'\rangle \sim_{lev}$

$\langle c_{2,i}, lkst_i, mdst_{2,i}, mem'_{2,i}\rangle$ for all $i \in \{1,\ldots,m\}$, we get that $c_{2,i} = \textbf{stop}$ for all $i \in \{1,\ldots,m\}$. Hence, we get from $\overrightarrow{ccnf}_2 = [(c_{2,1}, lkst_1, mdst_{2,1}), \ldots, (c_{2,m}, lkst_{2,m}, mdst_m)]$ by the definition of trm that $trm(\overrightarrow{ccnf}_2)$.

From the typing rules TSP and TTH, we know that the command of each thread is typeable with partial type environments that have an empty preimage. As we have seen in Lemma 11, this means that all resulting mode state must be compatible with the partial type environment with empty preimage. Thus, $x \notin mdst_{1,i}(\mathsf{A\text{-}NR})$ holds for all x with $lev(x) = \textbf{low}$. Hence, we get from $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$ hold for all x with $(lev(x) = \textbf{high}) \vee mem_1(x) = mem_2(x) \vee x \notin \bigcup_{j\in\{1,\ldots,n\}} mdst_{1,j}(\mathsf{A\text{-}NR})$, that $mem'_{1,i}(x) = mem'_1(x)$ and $mem'_{2,i}(x) = mem'_2(x)$ hold for all x with $lev(x) = \textbf{low}$. From this combined with $\langle c_{1,i}, lkst_i, mdst_{1,i}, mem'_{1,i}\rangle \sim_{lev} \langle c_{2,i}, lkst_i, mdst_{2,i}, mem'_{2,i}\rangle$ for all $i \in \{1,\ldots,m\}$ we get by the definition of $\sim_{lev}$, $=^{lev}_{\textbf{low}}$, and $=^{lev,mdst}_{\textbf{low}}$ that $mem'_1 =^{lev}_{\textbf{low}} mem'_2$. □

The following is the proof of Corollary 1.

Proof. Corollary 1 follows directly from the soundness result for the security type system (Theorem 3), and the soundness result for the guarantee inference (Theorem 2), and the soundness result for the DPN analysis (Theorem 1). □

A.5 Proof for example analysis

The following is the proof sketch for our example analysis 4.

Proof (sketch). The judgment $\emptyset \vdash \emptyset, \emptyset\{\textbf{skip}; c_{s2}\}\emptyset, \emptyset : c'_{s2}$ with

$c'_{s2} =$
skip@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, ∅), rel(G-NW, ∅)];
spawn(
  skip@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, ∅), rel(G-NW, ∅)];
  lock(l)@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, {$o1$}), rel(G-NW, {$o2$})];
  $o2$:=$o1$@[acq(G-NR, {$o1$}), acq(G-NW, {$o2$}), rel(G-NR, {$o2$}), rel(G-NW, {$o1$})];
  $o1$:=$o2$@[acq(G-NR, {$o2$}), acq(G-NW, {$o1$}), rel(G-NR, ∅), rel(G-NW, ∅)];
  unlock(l)@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, ∅), rel(G-NW, ∅)]
)@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, ∅), rel(G-NW, ∅)];
lock(l)@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, {$s1$}), rel(G-NW, {$o2$})]@[acq(A-NR, {$o1$})];
$o1$:=$s1$@[acq(G-NR, {$s1$}), acq(G-NW, {$o1$}), rel(G-NR, {$s2$}), rel(G-NW, {$s1$})];
$s1$:=$s2$@[acq(G-NR, {$s2$}), acq(G-NW, {$s1$}), rel(G-NR, {$o1$}), rel(G-NW, {$s2$})];
$s2$:=$o1$@[acq(G-NR, {$o1$}), acq(G-NW, {$s2$}), rel(G-NR, ∅), rel(G-NW, {$o1$})];
$o1$:=0@[acq(G-NR, ∅), acq(G-NW, {$o1$}), rel(G-NR, ∅), rel(G-NW, ∅)];
unlock(l)@[acq(G-NR, ∅), acq(G-NW, ∅), rel(G-NR, ∅), rel(G-NW, ∅)]@[rel(A-NR, {$o1$})]

is derivable with the rules ISQ, ISK, ISP, ILO, IAS, IUL, and IAN.

The judgment $lev \vdash c'_{s2} : c''_{s2}$ is derivable with the rules TTH, TSQ, TAN, TSK, TSP, TLO, TEX, TAL, TUL, TFH, TAH, and TFL.

For $ccnf = (c'_{s2}, \emptyset, mdst_\perp)$ and the DPN $\mathcal{M}_{ccnf}$, we observe that the set of reachable DPN configurations starting from $ccnf\#$ has at most two concurrent control states, one for each thread. Furthermore, we observe that the spawned thread has only one control state that might be in conflict (as defined by the automaton $\mathcal{A}_{ccnf}$) with a

control configuration of the spawning thread, namely, $(c_{conflict}, lkst_{conflict}, mdst_{conflict})$ with

$c_{conflict} =$
$o2{:=}o1@[\mathsf{acq}(\text{G-NR}, \{o1\}), \mathsf{acq}(\text{G-NW}, \{o2\}), \mathsf{rel}(\text{G-NR}, \{o2\}), \mathsf{rel}(\text{G-NW}, \{o1\})];$
$o1{:=}o2@[\mathsf{acq}(\text{G-NR}, \{o2\}), \mathsf{acq}(\text{G-NW}, \{o1\}), \mathsf{rel}(\text{G-NR}, \emptyset), \mathsf{rel}(\text{G-NW}, \emptyset)];$
$\mathbf{unlock}(l)@[\mathsf{acq}(\text{G-NR}, \emptyset), \mathsf{acq}(\text{G-NW}, \emptyset), \mathsf{rel}(\text{G-NR}, \emptyset), \mathsf{rel}(\text{G-NW}, \emptyset)]$

and $lkst_{conflict} = \{l\}$ and $mdst_{conflict}(\text{G-NR}) = x \setminus \{(o1)\}$. However, all control states with a mode state $mdst'_{conflict}$ such that $o1 \in mdst'_{conflict}(\text{A-NR})$ also have a lock state $lkst'_{conflict}$ with $l \in lkst'_{conflict}$. Since $l \in lkst_{conflict}$ and $l \in lkst'_{conflict}$ no DPN configuration is reachable from $ccnf\#$ that contains the two control states that are in conflict due to consistent use of locks in the DPN. □