Proofs

Proof of Theorem 1. It is straightforward because NTNI is a generalization of TNI where the policy defines all possible flows explicitly. Hence by considering the transitive and reflexive closure ($\sqsubseteq^*$) of the transitive relation ($\sqsubseteq$) as the nontransitive one, the theorem holds.

1) Let $L_{\mathcal{N}} = L_{\mathcal{T}}, \trianglerighteq = \sqsubseteq^*$, and $\Gamma_{\mathcal{N}} = \Gamma_{\mathcal{T}}$. Then, $C(\ell) = \{\ell' | \ell' \trianglerighteq \ell\} = \{\ell' | \ell' \sqsubseteq^* \ell\}$, and according to the definitions 1 and 3, $\forall \ell . \left(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \iff M_1 \overset{\ell}{=}_{\mathcal{T}} M_2\right)$.
2) Considering Definitions 3 and 4,
$\Big(\forall \ell \in L_{\mathcal{N}}. \forall M_1, M_2. \left(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \wedge \langle c, M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle c, M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle\right) \Rightarrow M_1' \overset{\ell}{=}_{\mathcal{N}} M_2'\Big) \iff$
$\Big(\forall \ell \in L_{\mathcal{N}}. \forall M_1, M_2. \left(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \wedge \langle c, M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle c, M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle\right) \Rightarrow M_1' \overset{C(\ell)}{=}_{\mathcal{N}} M_2'\Big)$, thus the theorem holds.

□

Proof of Lemma 1. The transformed program c' is partitioned into three sections such that $c' = init_{c'}; orig_{c'}; final_{c'}$: (1) initial assignments for *temp* variables ($init_{c'}$), (2) the original program that variables are renamed to *temp* variables ($orig_{c'}$), and (3) final assignments for *sink* variables ($final_{c'}$).

1) The init section only sets the values of x_{temp} variables and each assignment is in the form of $x_{temp} := x$ for all $x \in Var_c$. We also know that $\forall x \in Var_c. x_{sink} \notin FV(init_{c'})$. Using the rule (WRITE) of the semantics by the number of elements in Var_c, we can conclude that the init section always terminates and $\forall M. \exists! M'. \langle init_{c'}, M\rangle \rightarrow^{|Var_c|} \langle \text{stop}, M'\rangle \wedge \forall x \in Var_c. M'(x_{temp}) = M'(x) = M(x) \wedge M'(x_{sink}) = M(x_{sink})$.
2) The program c and the $orig_{c'}$ section are identical up to α-renaming of variables $x \in Var_c$ with x_{temp}, and $\forall x \in Var_c. x \notin FV(orig_{c'}) \wedge x_{sink} \notin FV(orig_{c'})$. Thus, we write $\forall M_1, M_2. \forall x \in Var_c. M_1(x) = M_2(x) = M_2(x_{temp}) \Rightarrow \forall n \in \mathbb{N}. \langle c, M_1\rangle \rightarrow^n \langle c_1, M_1'\rangle \wedge \langle orig_{c'}, M_2\rangle \rightarrow^n \langle c_2, M_2'\rangle \wedge M_1'(x) = M_2'(x_{temp}) \wedge M_2'(x) = M_2(x) = M_1(x) \wedge M_2(x_{sink}) = M_2'(x_{sink})$.
3) The final section includes assignments from the value of x_{temp} variables to x_{sink} variables where assignments are in the form of $x_{sink} := x_{temp}$ for all $x \in Var_c$. We also know that $\forall x \in Var_c. x \notin FV(final_{c'})$. Similar to the init section, by applying the rule (WRITE) by the number of elements in Var_c, we can write $\forall M. \exists! M'. \langle final_{c'}, M\rangle \rightarrow^{|Var_c|} \langle \text{stop}, M'\rangle \wedge \forall x \in Var_c. M'(x_{sink}) = M'(x_{temp}) = M(x_{temp}) \wedge M'(x) = M(x)$.
4) If we use the semantic rule (SEQ-I) for the sequence of these three sections and follow the aforementioned statements, we can conclude that Lemma 1 holds.

□

Proof of Lemma 2. Using Lemma 1, we can establish a correspondence between the two security definitions. We have $\langle c, M\rangle \rightarrow^* \langle \text{stop}, M'\rangle \iff \langle Canonical(c), M\rangle \rightarrow^* \langle \text{stop}, M''\rangle$, which means the termination behavior stays the same. Then given that $\forall x \in Var_c. M'(x) = M''(x_{temp}) = M''(x_{sink}) \wedge M(x) = M''(x)$, the lemma is proven. □

Proof of Lemma 3. For simplicity, we write $c' = Canonical(c)$. We know that $\forall x. \big(P(x) \iff Q(x)\big) \Rightarrow \big(\forall x. P(x) \iff \forall x. Q(x)\big)$. So to prove the lemma, we show the correctness of the following statement:
$\forall M_1, M_2. \langle c', M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle c', M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle \Rightarrow \Big(\forall \ell \in L_{\mathcal{N}}. \big(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \Rightarrow M_1' \overset{\ell}{=}_{\mathcal{N}} M_2'\big) \iff \forall \ell' \in L_{\mathcal{T}}. \big(M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2 \Rightarrow M_1' \overset{\ell'}{=}_{\mathcal{T}} M_2'\big)\Big)$.
If the execution of the program c' for (at least) one of the two arbitrary memories M_1 and M_2 does not terminate, then the premise in both security definitions does not hold, thus the lemma holds. Assuming the program is terminating for both memories, we prove the statement as follows:

1) Left to right:
   a) Let $I_{\mathcal{N}} = \{\ell \in L_{\mathcal{N}} | M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2\}$ be the set of levels in $L_{\mathcal{N}}$ that the two memories are indistinguishable for the set of labels can flow to them. Then, we have $I_{\mathcal{N}} \in L_{\mathcal{T}} \wedge I_{\mathcal{T}} = \{\ell' \in L_{\mathcal{T}} | M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2\} = \{\ell' \in L_{\mathcal{T}} | \ell \in I_{\mathcal{N}} \wedge \ell' \in \wp(C(\ell))\} = \wp(I_{\mathcal{N}})$ based on Definition 1.
   b) Using Lemma 1, we can conclude that $\forall \ell \in I_{\mathcal{N}}. \forall x \in Var_c. \Gamma_{\mathcal{N}}(x) = \ell \Rightarrow \big(\exists x_{sink} \in Var_{c'}. \Gamma_{\mathcal{T}}(x_{sink}) = C(\ell) \wedge M_1'(x_{sink}) = M_2'(x_{sink})\big) \wedge \big(\exists x \in Var_{c'}. \Gamma_{\mathcal{T}}(x) = \{\ell\} \wedge M_1'(x) = M_2'(x)\big) \wedge \big(\exists x_{temp} \in Var_{c'}. \Gamma_{\mathcal{T}}(x_{temp}) = L_{\mathcal{N}} \wedge M_1'(x_{temp}) = M_2'(x_{temp})\big) \wedge \ell \in I_{\mathcal{T}} \wedge C(\ell) \in I_{\mathcal{T}}$.
   c) Therefore, $\forall \ell \in I_{\mathcal{N}}. M_1' \overset{\ell}{=}_{\mathcal{N}} M_2' \iff \forall \ell' \in I_{\mathcal{T}}. M_1' \overset{\ell'}{=}_{\mathcal{T}} M_2'$. Hence, $\forall \ell \in L_{\mathcal{N}}. \big(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \Rightarrow M_1' \overset{\ell}{=}_{\mathcal{N}} M_2'\big) \Rightarrow \forall \ell' \in L_{\mathcal{T}}. \big(M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2 \Rightarrow M_1' \overset{\ell'}{=}_{\mathcal{T}} M_2'\big)$.
2) Right to left:
   a) Let $I_{\mathcal{T}} = \{\ell' \in L_{\mathcal{T}} | M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2\}$ and $I_{\mathcal{N}} = \{\ell \in L_{\mathcal{N}} | M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2\} = \{\ell \in L_{\mathcal{N}} | C(\ell) \in I_{\mathcal{T}}\}$.
   b) According to Lemma 1, we have $\forall \ell' \in I_{\mathcal{T}}. \exists \ell \in L_{\mathcal{N}}. \Big(\ell' = \{\ell\} \Rightarrow \wp(C(\ell)) \subseteq I_{\mathcal{T}} \wedge \forall x \in Var_c. \Gamma_{\mathcal{N}}(x) = \ell \Rightarrow \big(\exists x_{sink} \in Var_{c'}. \Gamma_{\mathcal{T}}(x_{sink}) = C(\ell) \wedge M_1'(x_{sink}) = M_2'(x_{sink})\big) \wedge \big(\exists x \in Var_{c'}. \Gamma_{\mathcal{T}}(x) = \ell' \wedge M_1'(x) = M_2'(x)\big) \wedge \big(\exists x_{temp} \in Var_{c'}. \Gamma_{\mathcal{T}}(x_{temp}) = L_{\mathcal{N}} \wedge M_1'(x_{temp}) = M_2'(x_{temp})\big)\Big)$.
   c) Thus, $\forall \ell' \in I_{\mathcal{T}}. M_1' \overset{\ell'}{=}_{\mathcal{T}} M_2' \iff \forall \ell \in I_{\mathcal{N}}. M_1' \overset{\ell}{=}_{\mathcal{N}} M_2'$. Hence, $\forall \ell' \in L_{\mathcal{T}}. \big(M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2 \Rightarrow M_1' \overset{\ell'}{=}_{\mathcal{T}} M_2'\big) \Rightarrow \forall \ell \in L_{\mathcal{N}}. \big(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \Rightarrow M_1' \overset{\ell}{=}_{\mathcal{N}} M_2'\big)$.

□

Proof of Theorem 2. By using Lemma 2 and Lemma 3. □

Proof of Theorem 3. To show soundness of the type system, we prove the following statement: $pc \vdash \Gamma\{c'\}\Gamma' \Rightarrow \Big(\forall \ell \in L_{\mathcal{T}}. \forall M_1, M_2. \big(M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2 \wedge \langle c', M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle c', M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle\big) \Rightarrow M_1' \overset{\ell}{=}_{\Gamma',\mathcal{T}} M_2'\Big) \wedge \forall x \in Var_{sink}. \Gamma'(x) = \Gamma(x)$, where $c' = Canonical(c)$ and $M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2 \iff \forall x \in Var_{c'}. \Gamma(x) \sqsubseteq \ell \Rightarrow M_1(x) = M_2(x)$. The first part of the statement denotes the definition of security in the flow-sensitive style and the second part of the statement ensures the flow-insensitivity of *sink* variables.

The first three rules determine the security level of expression e, which is the join of security levels associated with free variables of the expression.

By induction on the typing derivation and the structure of c', we have $\forall M. \forall x \in Var_{c'}. \big(\langle c', M\rangle \rightarrow^* \langle \text{stop}, M'\rangle \wedge pc \vdash \Gamma\{c'\}\Gamma' \wedge pc \not\sqsubseteq \Gamma'(x)\big) \Rightarrow M(x) = M'(x)$, where $pc \not\sqsubseteq \Gamma'(x)$ implies that no assignment to x occurs in c'. Note that in the assignment to sink variables (rule TT-WRITE-II), the memory gets updated in a secure way since $pc \sqcup \Gamma(x') \sqsubseteq \Gamma(x) \Rightarrow pc \sqsubseteq \Gamma(x)$.

It can also be easily proven by induction on the typing derivation that $pc \vdash \Gamma\{c'\}\Gamma' \wedge pc' \sqsubseteq pc \Rightarrow pc' \vdash \Gamma\{c'\}\Gamma'$.

By induction on the typing derivation and the structure of c', we show that $pc \vdash \Gamma\{c'\}\Gamma' \Rightarrow \forall \ell \in L_{\mathcal{T}}. \forall M_1, M_2. \big(M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2 \wedge \langle c', M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle c', M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle\big) \Rightarrow M_1' \overset{\ell}{=}_{\Gamma',\mathcal{T}} M_2'$. We discuss the cases as follows:

- Case (TT-SKIP): We directly can write $pc \vdash \Gamma\{skip\}\Gamma \Rightarrow \forall \ell \in L_{\mathcal{T}}. \forall M_1, M_2. \big(M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2 \wedge \langle skip, M_1\rangle \rightarrow^* \langle \text{stop}, M_1\rangle \wedge \langle skip, M_2\rangle \rightarrow^* \langle \text{stop}, M_2\rangle\big) \Rightarrow M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2$.
- Case (TT-WRITE-I): The conclusion part is $pc \vdash \Gamma\{x := e\}\Gamma[x \mapsto pc \sqcup t]$, thus Γ and Γ' only might differ in x; and similarly for M_1 and M_2. The statement holds for this case because $pc \sqsubseteq \Gamma'(x) = pc \sqcup t$.
- Case (TT-WRITE-II): The condition $pc \sqcup \Gamma(x') \sqsubseteq \Gamma(x)$ checks if the assignment is permitted with regard to the transitive policy; it captures implicit (pc) and explicit ($\Gamma(x')$) flows to the variable x. Thus, we have $pc \vdash \Gamma\{x := x'\}\Gamma \Rightarrow \forall \ell \in L_{\mathcal{T}}. \forall M_1, M_2. \big(M_1 \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2 \wedge \langle x := x', M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle x := x', M_2\rangle \rightarrow^* \langle \text{stop}, M_2'\rangle\big) \Rightarrow M_1' \overset{\ell}{=}_{\Gamma,\mathcal{T}} M_2'$.
- Case (TT-IF): Based on the induction hypothesis, $pc \sqcup t \vdash \Gamma\{c_b\}\Gamma' \Rightarrow TNI_{TI}(\mathcal{T}, c_b)$ for $b = true, false$. Since $pc \sqsubseteq pc \sqcup t$, the statement holds for this case.
- Case (TT-WHILE): Based on the induction hypothesis, we have $pc \sqcup t \vdash \Gamma\{c_{body}\}\Gamma \Rightarrow TNI_{TI}(\mathcal{T}, c_{body})$, and $pc \sqsubseteq pc \sqcup t$, thus $pc \vdash \Gamma\{c\}\Gamma \Rightarrow TNI_{TI}(\mathcal{T}, c)$ for $c = while\ e\ do\ c_{body}$.
- Case (TT-Seq): Using the induction hypothesis, we have $pc \vdash \Gamma\{c_1\}\Gamma' \Rightarrow TNI_{TI}(\mathcal{T}, c_1) \wedge pc \vdash \Gamma'\{c_2\}\Gamma'' \Rightarrow TNI_{TI}(\mathcal{T}, c_2)$. Therefore, $pc \vdash \Gamma\{c_1; c_2\}\Gamma'' \Rightarrow TNI_{TI}(\mathcal{T}, c_1; c_2)$.
- Case (TT-SUB): Based on the induction hypothesis, $pc_1 \vdash \Gamma_1\{c\}\Gamma_1' \Rightarrow TNI_{TI}(\mathcal{T}, c)$. Considering the conditions $pc_2 \sqsubseteq pc_1 \wedge \Gamma_2 \sqsubseteq \Gamma_1 \wedge \Gamma_1' \sqsubseteq \Gamma_2'$, we can conclude $pc_2 \vdash \Gamma_2\{c\}\Gamma_2' \Rightarrow TNI_{TI}(\mathcal{T}, c)$.

We also prove the second part which requires the levels of *sink* variables remain unmodified through the program. There is no typing rule that updates the level of *sink* variables of the program, and the subsumption rule (rule TT-SUB) obviously guarantees the property. Therefore, by induction on the typing derivation, we have $\forall x \in Var_{sink}. \Gamma'(x) = \Gamma(x)$.

□

Proof of Theorem 4. By induction on the derivation of expressions, we prove the type for expression e is the union of the security levels (i.e., the collected information flows) of free variables of the expression, formally $\Gamma \vdash e : t \Rightarrow t = \bigcup_{x \in FV(e)} \Gamma(x)$:

- Case (VALUE): We label values as empty set since they are visible for all levels and no free variable exists.
- Case (NT-READ): The type of variable x (i.e., $\Gamma(x)$) is the set of labels that might affect the value of the variable x in the program. It must capture all the possible flows to the variable, including the label of itself.
- Case (NT-OPERATION): Based on the induction hypothesis, it is easy to conclude that $t_1 \cup t_2 = \bigcup_{x \in FV(e_1 \oplus e_2)} \Gamma(x)$.
- Case (NT-SUB-I): The subtyping rule for expressions shows adding more security labels to the type of e keeps the expression well-typed.

By induction on the typing derivation and the structure of c, we prove the theorem as follows:

- Case (NT-SKIP): It is easy to see that $\mathcal{P}, \Gamma, pc \vdash skip : t \Rightarrow NTNI_{TI}(\mathcal{N}, skip)$ for any $\mathcal{N}$.
- Case (NT-WRITE): This rule checks the explicit and implicit flows to the variable x have been collected in $\Gamma(x)$ and permitted by $\trianglerighteq$ relation. The type t is the union set of $\Gamma(x)$ (all collected information flows) and the type of e (the explicit flows). Considering pc (implicit flows) of the assignment, the premise investigates the presence of all labels in $t \cup pc$ in the collected flows to the variable x ($\Gamma(x)$), and guarantees that those are permitted according to the nontransitive flow. Hence, $\forall M_1, M_2. M_1 \overset{C(t \cup pc)}{=}_{\mathcal{N}} M_2 \wedge \langle x := e, M_1\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \wedge \langle x := e, M_2\rangle \rightarrow^* \langle \text{stop}, M_1'\rangle \Rightarrow M_1' \overset{t \cup pc}{=}_{\mathcal{N}} M_2'$. Thus, $NTNI_{TI}(\mathcal{N}, x := e)$ holds.
- Case (NT-IF): Based on the subtyping rule, we write $\mathcal{P}, \Gamma, pc \cup t_1 \vdash c_{true} : t_2 \Rightarrow \mathcal{P}, \Gamma, pc \vdash c_{true} : t_2$, and similarly for c_{false}. Aggregating the labels in t_1 and t_2 and using the induction hypothesis prove the theorem statement for *if* commands.
- Case (NT-WHILE) Similar to the previous case, if c_{body} is well-typed under $pc \cup t_1$, according to the induction hypothesis, this case is also proved.
- Case (NT-SEQ): Using the induction hypothesis, $c_1; c_2$ has type $t_1 \cup t_2$ and $NTNI(\mathcal{N}, c_1; c_2)$ holds.

- Case (NT-SUB-II): The induction hypothesis shows $NTNI(\mathcal{N}, c)$ holds if c is well-typed, for example, has type t_1 under pc_1. If we extend the type with more security labels under a smaller pc, the command c remains well-typed and satisfies $NTNI(\mathcal{N}, c)$.

□

Proof of Theorem 5. First, we start with demonstrating that $\mathcal{P}, \Gamma_1, pc \vdash c : t \Rightarrow \mathcal{P}', \Gamma'_1, pc \vdash c' : t$, where $c' = Canonical(c)$ and we extend the typing context Γ_1 to Γ'_1 and the labeling function $\mathcal{P}$ to $\mathcal{P}'$ by adding temp and sink variables with the same mappings for any variable x of the program, i.e., $\forall x \in Var_c. \mathcal{P}'(x) = \mathcal{P}'(x_{temp}) = \mathcal{P}'(x_{sink}) = \mathcal{P}(x) \wedge \Gamma'_1(x) = \Gamma'_1(x_{temp}) = \Gamma'_1(x_{sink}) = \Gamma_1(x)$.

As discussed in Lemma 1, the program is partitioned in three parts: $c' = init_{c'}; orig_{c'}; final_{c'}$. By induction on the derivation of $init_{c'}$ and using the two rules (NT-WRITE) and (NT-SEQ), we have $\mathcal{P}', \Gamma'_1, pc \vdash init_{c'} : t$ because statements are assignments of the form $x_{temp} := x$ and $\Gamma'_1(x) = \Gamma'_1(x_{temp})$. Also, since $\mathcal{P}, \Gamma_1, pc \vdash c : t$ holds, then $\forall \ell \in \Gamma_1(x) \trianglerighteq \mathcal{P}(x)$, and thus $\forall \ell \in \Gamma'_1(x_{temp}). \ell \trianglerighteq \mathcal{P}'(x_{temp})$.

We know that c and $orig(c')$ are identical up to α-renaming of variables $x \in Var_c$ with x_{temp}. Therefore, $\mathcal{P}, \Gamma_1, pc \vdash c : t \Rightarrow \mathcal{P}', \Gamma'_1, pc \vdash c' : t$ because $\Gamma_1(x) = \Gamma'_1(x_{temp})$, $\mathcal{P}(x) = \mathcal{P}(x_{temp})$, and $x, x_{sink} \notin FV(c')$.

At the final section, statements are the form of $x_{sink} := x_{temp}$. Similar to the init section, because $\Gamma'_1(x_{temp}) = \Gamma'_1(x_{sink})$ and $\forall \ell \in \Gamma'_1(x_{sink}). \ell \trianglerighteq \mathcal{P}'(x_{sink})$, we can write $\mathcal{P}', \Gamma'_1, pc \vdash final_{c'} : t$. Applying the rule (NT-SEQ) two times, we conclude $\mathcal{P}', \Gamma'_1, pc \vdash init_{c'}; orig_{c'}; final_{c'} : t$.

Then, we prove $\mathcal{P}', \Gamma'_1, pc \vdash c' : t \Rightarrow pc \vdash \Gamma_2\{c'\}\Gamma_3$ where $c' = Canonical(c)$. Remember that in the transitive type system $L_{\mathcal{T}} = \wp(L_{\mathcal{N}})$, $\sqsubseteq = \subseteq$, and $\sqcup = \cup$. To connect the typing contexts together meaningfully, the following constraints must be considered $\forall x \in Var_c$:

- $\Gamma_3(x_{temp}) \sqsubseteq \Gamma'_1(x_{temp})$: The final type of x_{temp} contains the set of labels in the last assignment that flow to the variable in the program c', due to flow-sensitivity of the transitive type system, while $\Gamma'_1(x_{temp})$ is the predicted set of *all* information flows to the variable x_{temp}.
- $\Gamma_2(x) = \{\mathcal{P}(x)\}, \Gamma_2(x_{temp}) = L_{\mathcal{N}}, \Gamma_2(x_{sink}) = C(\mathcal{P}(x))$: The conditions are based on the labeling function presented in Definition 5 to adjust the nontransitive mapping to the transitive one.
- $\Gamma_3(x) = \Gamma_2(x), \Gamma_3(x_{sink}) = \Gamma_2(x_{sink})$: As shown in Figure 9a, if the program is well-typed, the types for variables remain untouched except for Var_{temp}.

There is a one-to-one correspondence between typing rules for expressions, which yields the union set of $\Gamma(x)$ for free variables $FV(e)$ as the type of the expression e. Thus, $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$.

By induction on the nontransitive typing derivation $\mathcal{P}', \Gamma'_1, pc \vdash c' : t$ and the structure of c':

- Case (NT-SKIP): Based on the rule (TT-SKIP), $pc \vdash \Gamma_2\{c'\}\Gamma_2$ holds.
- Case (NT-WRITE): We separate this case for two subcases according to the variable on the left side of the assignment:
  - If $x \in Var_{temp}$, since $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$, based on the rule (TT-WRITE-I), we write $pc \vdash \Gamma_2\{c'\}\Gamma_2[x \mapsto pc \sqcup t']$.
  - If $x \in Var_{sink}$, we know that $e = x_{temp}$ is the only case in program c' at the $final_{c'}$ section. Because $\Gamma_3(x_{temp}) \subseteq \Gamma'_1(x_{temp})$ and $\forall \ell \in \Gamma'_1(x_{temp}) \cup pc. \ell \in \Gamma'_1(x_{sink}) \wedge \ell \trianglerighteq \mathcal{P}'(x_{sink}) \Rightarrow pc \sqcup \Gamma_3(x_{temp}) \sqsubseteq C(\mathcal{P}'(x_{sink})) \Rightarrow pc \sqcup \Gamma_3(x_{temp}) \sqsubseteq \Gamma_3(x_{sink})$. Hence, based on the rule (TT-WRITE-II), $pc \vdash \Gamma_3\{x := e\}\Gamma_3$.
- Case (NT-IF): Using the induction hypothesis and $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$, the statement $pc \vdash \Gamma_2\{c'\}\Gamma_3$ holds for this case with respect to the rule (TT-IF).
- Case (NT-WHILE): Similar to the case (NT-IF), and according to the rule (TT-WHILE).
- Case (NT-SEQ): Using the induction hypothesis, $pc \vdash \Gamma_2\{c_1\}\Gamma_3$ and $pc \vdash \Gamma_3\{c_2\}\Gamma_4$, then $pc \vdash \Gamma_2\{c_1; c_2\}\Gamma_4$ by using the rule (TT-SEQ).
- Case (NT-SUB-II): Using the induction hypothesis, we write $pc_1 \vdash \Gamma_2\{c\}\Gamma'_2$. Since $pc_2 \sqsubseteq pc_1$, $\Gamma_3 \sqsubseteq \Gamma_2$, $\Gamma'_2 \sqsubseteq \Gamma'_3$ and in combination with the rule (NT-SUB-I), $pc_2 \vdash \Gamma_3\{c\}\Gamma'_3$ holds.

□

Proof of Theorem 6. Simliar to the proof of Theorem 1, by considering the transitive and reflexive closure ($\sqsubseteq^*$) of the transitive relation ($\sqsubseteq$) as the nontransitive one, the theorem holds.

1) Let $L_{\mathcal{N}} = L_{\mathcal{T}}, \trianglerighteq = \sqsubseteq^*$, and $\Gamma_{\mathcal{N}} = \Gamma_{\mathcal{T}}$. Then, $C(\ell) = \{\ell' | \ell' \trianglerighteq \ell\} = \{\ell' | \ell' \sqsubseteq^* \ell\}$, and according to the definitions 8 and 12, $\forall \ell. \left(I_1 \overset{C(\ell)}{=}_{\mathcal{N}} I_2 \iff I_1 \overset{\ell}{=}_{\mathcal{T}} I_2\right)$, and based on the defintions 7 and 11, $\forall \ell. \left(O_1 \overset{C(\ell)}{=}_{\mathcal{N}} O_2 \iff O_1 \overset{\ell}{=}_{\mathcal{T}} O_2\right)$.
2) Considering Definitions 11 and 14, $\Big(\forall \ell \in L_{\mathcal{N}}. \forall M. \forall I_1, I_2. \left(I_1 \overset{C(\ell)}{=}_{\mathcal{N}} I_2 \wedge \langle c, M, I_1, \emptyset\rangle \rightsquigarrow O_1\right) \Rightarrow \exists O_2. \langle c, M, I_2, \emptyset\rangle \rightsquigarrow O_2 \wedge O_1 \overset{\ell}{=}_{\mathcal{N}} O_2\Big) \iff \Big(\forall \ell \in L_{\mathcal{N}}. \forall M. \forall I_1, I_2. \left(I_1 \overset{C(\ell)}{=}_{\mathcal{N}} I_2 \wedge \langle c, M, I_1, \emptyset\rangle \rightsquigarrow O_1\right) \Rightarrow \exists O_2. \langle c, M, I_2, \emptyset\rangle \rightsquigarrow O_2 \wedge O_1 \overset{C(\ell)}{=}_{\mathcal{N}} O_2\Big)$, thus the theorem holds.

□

Proof of Lemma 4. It is clear that the transformation only modifies the labels in the input and output commands of the given program, thus the behavior of the rest of the program stays unaffected. The changes in the labels of the input commands can be formulated as $\forall \ell. I(\ell) = I'(\{\ell\})$, where I is the input for the program c and I' is the input for the program c'.

By induction on the semantic rules shown in Figure 23, it is proven that c' progresses the same as c with the difference that outputs are sent to the channel $C(\ell)$ in lieu of ℓ. Therefore,

we formulate it for the two outputs O and O' of programs c and c' respectively as $O' = O[v_\ell \mapsto v_{C(\ell)}]$, which means the only difference between the output sequences O and O' are the labels of output values; ones with the label ℓ in O are recorded at the same index in O' with the label $C(\ell)$.

□

Proof of Theorem 7. Let $c' = Transform(c)$, $L_\mathcal{T} = \wp(L_\mathcal{N})$, $\sqsubseteq = \subseteq$ and $\forall x \in Var_c.\, \Gamma_\mathcal{T}(x) = \{\Gamma_\mathcal{N}(x)\}$.

1) We have $\forall \ell \in L_\mathcal{N}.\forall I_1, I_2.\, I_1 \overset{C(\ell)}{=}_\mathcal{N} I_2 \iff \forall \ell' \in L_\mathcal{T}.\forall I'_1, I'_2.\, I'_1 \overset{\ell'}{=}_\mathcal{T} I'_2$ because of Definitions 8 and 12. Based on Lemma 4, we also know $\forall \ell \in L_\mathcal{N}.I(\ell) = I'(\{\ell\})$.
2) According to Definitions 10 and 14, and the semantic relation presented in Lemma 4, the statement $\forall M.\Big(\forall \ell \in L_\mathcal{N}.\forall I_1, I_2.\big(I_1 \overset{C(\ell)}{=}_\mathcal{N} I_2 \wedge \langle c, M, I_1, \varnothing\rangle \rightsquigarrow O_1\big) \Rightarrow \exists O_2.\langle c, M, I_2, \varnothing\rangle \rightsquigarrow O_2 \wedge O_1 \overset{\ell}{=}_\mathcal{N} O_2\Big) \iff \Big(\forall \ell' \in L_\mathcal{T}.\forall I'_1, I'_2.\big(I'_1 \overset{\ell'}{=}_\mathcal{T} I'_2 \wedge \langle c', M, I'_1, \varnothing\rangle \rightsquigarrow O'_1\big) \Rightarrow \exists O'_2.\langle c', M, I'_2, \varnothing\rangle \rightsquigarrow O'_2 \wedge O'_1 \overset{\ell'}{=}_\mathcal{T} O'_2\Big)$ holds if $O_1 \overset{\ell}{=}_\mathcal{N} O_2 \iff O'_1 \overset{\ell'}{=}_\mathcal{T} O'_2$.
3) As stated in Lemma 4, we conclude $O'_1 = O_1[v_\ell \mapsto v_{C(\ell)}] \wedge O'_2 = O_2[v_\ell \mapsto v_{C(\ell)}]$. Hence, $O_1 \overset{\ell}{=}_\mathcal{N} O_2 \iff O'_1 \overset{\ell'}{=}_\mathcal{T} O'_2$ and consequently, the theorem holds.

□

Proof of Theorem 8. We prove the following statement by induction on the typing derivation and the structure of $c' = Transform(c)$: $pc \vdash \Gamma\{c'\}\Gamma' \Rightarrow \forall \ell \in L_\mathcal{T}.\forall M.\forall I_1, I_2.\, I_1 \overset{\ell}{=}_\mathcal{T} I_2 \wedge \langle c, M, I_1, \varnothing\rangle \rightsquigarrow O_1 \Rightarrow \exists O_2.\langle c, M, I_2, \varnothing\rangle \rightsquigarrow O_2 \wedge O_1 \overset{\ell}{=}_\mathcal{T} O_2$.

The first three rules calculate the security level for expression e, by joining the security levels of its free variables.

The commands that update the security level of a variable are assignment (rules IO-TT-WRITE) and input (rule IO-TT-INPUT). Therefore, by induction on the typing derivation and the structure of c', we can write $\forall I.\forall M.\forall x \in Var_{c'}.\big(\langle c', M, I, \varnothing\rangle \rightarrow^* \langle c'', M', I', O\rangle \wedge pc \vdash \Gamma\{c'\}\Gamma' \wedge pc \not\sqsubseteq \Gamma'(x)\big) \Rightarrow M(x) = M'(x)$, where $pc \not\sqsubseteq \Gamma'(x)$ implies that no input or assignment to x occurs in c'. Note that for input commands (rule TT-WRITE-II), the memory gets updated in a secure way since $pc \sqsubseteq \Gamma'(x)$.

It can be easily proven by induction on the typing derivation that $pc \vdash \Gamma\{c\}\Gamma' \wedge pc' \sqsubseteq pc \Rightarrow pc' \vdash \Gamma\{c\}\Gamma'$.

Next, we investigate each case as follows:

- Case (IO-TT-SKIP): It is easy to see that $pc \vdash \Gamma\{skip\}\Gamma \Rightarrow \forall \ell \in L_\mathcal{T}.\forall M.\forall I_1, I_2.\, I_1 \overset{\ell}{=}_\mathcal{T} I_2 \wedge \langle skip, M, I_1, O\rangle \rightsquigarrow O \Rightarrow \langle skip, M, I_2, O\rangle \rightsquigarrow O \wedge O \overset{\ell}{=}_\mathcal{T} O$.
- Case (IO-TT-WRITE): For this case, we can write $pc \vdash \Gamma\{x := e\}\Gamma' \Rightarrow \forall \ell \in L_\mathcal{T}.\forall M.\forall I_1, I_2.\, I_1 \overset{\ell}{=}_\mathcal{T} I_2 \wedge \langle x := e, M, I_1, O\rangle \rightsquigarrow O \Rightarrow \langle x := e, M, I_2, O\rangle \rightsquigarrow O \wedge O \overset{\ell}{=}_\mathcal{T} O$. Note that the security label of the variable after the execution of the command carries both implicit (pc) and explicit (t) dependencies.
- Case (IO-TT-IF): Based on the induction hypothesis, $pc \sqcup t \vdash \Gamma\{c_b\}\Gamma' \Rightarrow TNI_{PI}(\mathcal{T}, c_b)$ for $b = true, false$. Since $pc \sqsubseteq pc \sqcup t$, the statement holds for this case.
- Case (IO-TT-WHILE): Based on the induction hypothesis, we have $pc \sqcup t \vdash \Gamma\{c_{body}\}\Gamma \Rightarrow TNI_{PI}(\mathcal{T}, c_{body})$, and $pc \sqsubseteq pc \sqcup t$, thus $pc \vdash \Gamma\{c\}\Gamma \Rightarrow TNI_{PI}(\mathcal{T}, c)$ for $c = while\ e\ do\ c_{body}$.
- Case (IO-TT-SEQ): Using the induction hypothesis, we have $pc \vdash \Gamma\{c_1\}\Gamma' \Rightarrow TNI_{PI}(\mathcal{T}, c_1) \wedge pc \vdash \Gamma'\{c_2\}\Gamma'' \Rightarrow TNI_{PI}(\mathcal{T}, c_2)$. Therefore, $pc \vdash \Gamma\{c_1; c_2\}\Gamma'' \Rightarrow TNI_{PI}(\mathcal{T}, c_1; c_2)$.
- Case (IO-TT-INPUT): Taking the condition $pc \sqsubseteq \ell$ into account, the type system only accepts input commands in the same context as the label ℓ or lower. Leaving the premise empty makes the type system unsound, due to not considering implicit flow (pc) to inputs from the level ℓ. Hence, $pc \vdash \Gamma\{input(x, \ell')\}\Gamma' \Rightarrow \forall \ell \in L_\mathcal{T}.\forall M.\forall I_1, I_2.\, I_1 \overset{\ell}{=}_\mathcal{T} I_2 \wedge \langle input(x, \ell'), M, I_1, O\rangle \rightsquigarrow O \Rightarrow \langle input(x, \ell'), M, I_2, O\rangle \rightsquigarrow O \wedge O \overset{\ell}{=}_\mathcal{T} O$.
- Case (IO-TT-OUTPUT): The condition $pc \sqcup \Gamma(x) \sqsubseteq \ell$ controls if the output is permitted with regard to the transitive policy; the premise monitors implicit flow (pc) and explicit flow ($\Gamma(x)$) to the output channel at the level ℓ. Thus, we have $pc \vdash \Gamma\{output(x, \ell')\}\Gamma \Rightarrow \forall \ell \in L_\mathcal{T}.\forall M.\forall I_1, I_2.\, I_1 \overset{\ell}{=}_\mathcal{T} I_2 \wedge \langle output(x, \ell'), M, I_1, O\rangle \rightsquigarrow O_1 \Rightarrow \langle output(x, \ell'), M, I_2, O\rangle \rightsquigarrow O_2 \wedge O_1 \overset{\ell}{=}_\mathcal{T} O_2$ since $O_1 = O_2 = O.M(x)_{\ell'}$.
- Case (IO-TT-SUB): $pc_1 \vdash \Gamma_1\{c\}\Gamma'_1 \Rightarrow TNI_{PI}(\mathcal{T}, c)$. Considering the conditions $pc_2 \sqsubseteq pc_1 \wedge \Gamma_2 \sqsubseteq \Gamma_1 \wedge \Gamma'_1 \sqsubseteq \Gamma'_2$, we can conclude $pc_2 \vdash \Gamma_2\{c\}\Gamma'_2 \Rightarrow TNI_{PI}(\mathcal{T}, c)$.

□

Proof of Lemma 5. For simplicity, we write $c' = Canonical(c)$. We know that $\forall x.\big(P(x) \iff Q(x)\big) \Rightarrow \big(\forall x.P(x) \iff \forall x.Q(x)\big)$. So to prove the lemma, we show the correctness of the following statement:
$\forall M_1, M_2.\langle c', M_1\rangle \rightarrow^* \langle \text{stop}, M'_1\rangle \wedge \langle c', M_2\rangle \rightarrow^* \langle \text{stop}, M'_2\rangle \Rightarrow \Big(\forall \ell \in L_\mathcal{N}.\big(M_1 \overset{C(\ell)}{=}_\mathcal{N} M_2 \Rightarrow M'_1 \overset{\ell}{=}_\mathcal{N} M'_2\big) \iff \forall \ell' \in L_\mathcal{T}.\big(M_1 \overset{\ell'}{=}_\mathcal{T} M_2 \Rightarrow M'_1 \overset{\ell'}{=}_\mathcal{T} M'_2\big)\Big)$.
If the execution of the program c' for (at least) one of the two arbitrary memories M_1 and M_2 does not terminate, then the premise in both security definitions does not hold, thus the lemma holds. Assuming the program is terminating for both memories, we prove the statement as follows:

1) Left to right:
   a) Let $I_\mathcal{N} = \{\ell \in L_\mathcal{N} | M_1 \overset{C(\ell)}{=}_\mathcal{N} M_2\}$ be the set of levels in $L_\mathcal{N}$ that the two memories are indistinguishable for the set of labels can flow to them. Then, we have $I_\mathcal{T} =$

$\{\ell' \in L_{\mathcal{T}} | M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2\} = \{\ell_{snk}, \ell_{src} \in L_{\mathcal{T}} | \ell \in I_{\mathcal{N}}\}$. Based on Definition 1, $M_1 \overset{\ell_{snk}}{=}_{\mathcal{T}} M_2 \Rightarrow M_1 \overset{\ell_{src}}{=}_{\mathcal{T}} M_2$.

b) Using Lemma 1, we can conclude that $\forall \ell \in I_{\mathcal{N}}.\forall x \in Var_c.\Gamma_{\mathcal{N}}(x) = \ell \Rightarrow \big(\exists x_{sink} \in Var_{c'}.\Gamma_{\mathcal{T}}(x_{sink}) = \ell_{snk} \wedge M'_1(x_{sink}) = M'_2(x_{sink})\big) \wedge \big(\exists x \in Var_{c'}.\Gamma_{\mathcal{T}}(x) = \ell_{src} \wedge M'_1(x) = M'_2(x)\big) \wedge \big(\exists x_{temp} \in Var_{c'}.\Gamma_{\mathcal{T}}(x_{temp}) = \top \wedge M'_1(x_{temp}) = M'_2(x_{temp})\big) \wedge \ell_{src}, \ell_{snk} \in I_{\mathcal{T}}$.

c) Therefore, $\forall \ell \in I_{\mathcal{N}}.M'_1 \overset{\ell}{=}_{\mathcal{N}} M'_2 \iff \forall \ell' \in I_{\mathcal{T}}.M'_1 \overset{\ell'}{=}_{\mathcal{T}} M'_2$. Hence, $\forall \ell \in L_{\mathcal{N}}.\big(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \Rightarrow M'_1 \overset{\ell}{=}_{\mathcal{N}} M'_2\big) \Rightarrow \forall \ell' \in L_{\mathcal{T}}.\big(M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2 \Rightarrow M'_1 \overset{\ell'}{=}_{\mathcal{T}} M'_2\big)$.

2) Right to left:

a) Let $I_{\mathcal{T}} = \{\ell' \in L_{\mathcal{T}} | M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2\}$ and $I_{\mathcal{N}} = \{\ell \in L_{\mathcal{N}} | M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2\} = \{\ell \in L_{\mathcal{N}} | \ell_{snk} \in I_{\mathcal{T}}\}$.

b) According to Lemma 1, we have $\forall \ell' \in I_{\mathcal{T}}.\exists \ell \in L_{\mathcal{N}}.\Big(\ell_{snk}, \ell_{src} \in I_{\mathcal{T}} \wedge \forall x \in Var_c.\Gamma_{\mathcal{N}}(x) = \ell \Rightarrow \big(\exists x_{sink} \in Var_{c'}.\Gamma_{\mathcal{T}}(x_{sink}) = \ell_{snk} \wedge M'_1(x_{sink}) = M'_2(x_{sink})\big) \wedge \big(\exists x \in Var_{c'}.\Gamma_{\mathcal{T}}(x) = \ell_{src} \wedge M'_1(x) = M'_2(x)\big) \wedge \big(\exists x_{temp} \in Var_{c'}.\Gamma_{\mathcal{T}}(x_{temp}) = \top \wedge M'_1(x_{temp}) = M'_2(x_{temp})\big)\Big)$.

c) Thus, $\forall \ell' \in I_{\mathcal{T}}.M'_1 \overset{\ell'}{=}_{\mathcal{T}} M'_2 \iff \forall \ell \in I_{\mathcal{N}}.M'_1 \overset{\ell}{=}_{\mathcal{N}} M'_2$. Hence, $\forall \ell' \in L_{\mathcal{T}}.\big(M_1 \overset{\ell'}{=}_{\mathcal{T}} M_2 \Rightarrow M'_1 \overset{\ell'}{=}_{\mathcal{T}} M'_2\big) \Rightarrow \forall \ell \in L_{\mathcal{N}}.\big(M_1 \overset{C(\ell)}{=}_{\mathcal{N}} M_2 \Rightarrow M'_1 \overset{\ell}{=}_{\mathcal{N}} M'_2\big)$. □

Proof of Theorem 9. By using Lemma 2 and Lemma 5. □

Proof of Theorem 10. Similar to the proof of Theorem 3. □

Proof of Theorem 11. We start with showing that $\mathcal{P}, \Gamma_1, pc \vdash c : t \Rightarrow \mathcal{P}', \Gamma'_1, pc \vdash c' : t$, where $c' = Canonical(c)$ and we extend the typing context Γ_1 to Γ'_1 and the labeling function $\mathcal{P}$ to $\mathcal{P}'$ by adding temp and sink variables with the same mappings for any variable x of the program, i.e., $\forall x \in Var_c.\mathcal{P}'(x) = \mathcal{P}'(x_{temp}) = \mathcal{P}'(x_{sink}) = \mathcal{P}(x) \wedge \Gamma'_1(x) = \Gamma'_1(x_{temp}) = \Gamma'_1(x_{sink}) = \Gamma_1(x)$.

As discussed in Lemma 1, the program is partitioned in three parts: $c' = init_{c'}; orig_{c'}; final_{c'}$. By induction on the derivation of $init_{c'}$ and using the two rules (NT-Write) and (NT-Seq), we have $\mathcal{P}', \Gamma'_1, pc \vdash init_{c'} : t$ because statements are assignments of the form $x_{temp} := x$ and $\Gamma'_1(x) = \Gamma'_1(x_{temp})$. Also, since $\mathcal{P}, \Gamma_1, pc \vdash c : t$ holds, then $\forall \ell \in \Gamma_1(x) \trianglerighteq \mathcal{P}(x)$, and thus $\forall \ell \in \Gamma'_1(x_{temp}).\ell \trianglerighteq \mathcal{P}'(x_{temp})$.

We know that c and $orig(c')$ are identical up to α-renaming of variables $x \in Var_c$ with x_{temp}. Therefore, $\mathcal{P}, \Gamma_1, pc \vdash c : t \Rightarrow \mathcal{P}', \Gamma'_1, pc \vdash c' : t$ because $\Gamma_1(x) = \Gamma'_1(x_{temp})$, $\mathcal{P}(x) = \mathcal{P}(x_{temp})$, and $x, x_{sink} \notin FV(c')$.

At the final section, statements are the form of $x_{sink} := x_{temp}$. Similar to the init section, because $\Gamma'_1(x_{temp}) = \Gamma'_1(x_{sink})$ and $\forall \ell \in \Gamma'_1(x_{sink}).\ell \trianglerighteq \mathcal{P}'(x_{sink})$, we can write $\mathcal{P}', \Gamma'_1, pc \vdash final_{c'} : t$. Applying the rule (NT-Seq) two times, we conclude $\mathcal{P}', \Gamma'_1, pc \vdash init_{c'}; orig_{c'}; final_{c'} : t$.

Then, we prove $\mathcal{P}', \Gamma'_1, pc \vdash c' : t \Rightarrow pc \vdash \Gamma_2\{c'\}\Gamma_3$ where $c' = Canonical(c)$. Remember that in the transitive type system $L_{\mathcal{T}} \supseteq \{\ell_{src}, \ell_{snk} | \ell \in L_{\mathcal{N}}\} \cup \{\top, \bot\}$ and $\forall \ell, \ell' \in L_{\mathcal{N}}.\ell \trianglerighteq \ell' \iff \ell_{src} \sqsubseteq \ell'_{snk}$ such that $\langle L_{\mathcal{T}}, \sqsubseteq \rangle$ is a lattice. To connect the typing contexts together meaningfully, the following constraints must be considered $\forall x \in Var_c$:

- $\Gamma_3(x_{temp}) \sqsubseteq \bigsqcup_{\ell \in \Gamma_1(x)} \ell_{src}$: The final type of x_{temp} is the join of the set of source labels in the last assignment that flow to the variable in the program c', due to flow-sensitivity of the transitive type system, while $\Gamma'_1(x_{temp})$ is the predicted set of *all* information flows to the variable x_{temp}. Thus, $\Gamma_3(x_{temp})$ should be lower than or equal to the join of corresponding source labels of $\Gamma'_1(x_{temp}) = \bigsqcup_{\ell \in \Gamma_1(x)} \ell_{src}$.
- $\mathcal{P}(x) = \ell \Rightarrow \Gamma_2(x) = \ell_{src}, \Gamma_2(x_{temp}) = \top, \Gamma_2(x_{sink}) = \ell_{snk}$: The conditions are based on the labeling function presented in Definition 15 to adjust the nontransitive mapping to the transitive one.
- $\Gamma_3(x) = \Gamma_2(x), \Gamma_3(x_{sink}) = \Gamma_2(x_{sink})$: As shown in Figure 9a, if the program is well-typed, the types for variables remain untouched except for Var_{temp}.

There is a one-to-one correspondence between typing rules for expressions, which yields the join of $\Gamma(x)$ for free variables $FV(e)$ as the type of the expression e. Thus, $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$.

By induction on the nontransitive typing derivation $\mathcal{P}', \Gamma'_1, pc \vdash c' : t$ and the structure of c':

- Case (NT-Skip): Based on the rule (TT-Skip), $pc \vdash \Gamma_2\{c'\}\Gamma_2$ holds.
- Case (NT-Write): We separate this case for two subcases according to the variable on the left side of the assignment:
  - If $x \in Var_{temp}$, since $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$, based on the rule (TT-Write-I), we write $pc \vdash \Gamma_2\{c'\}\Gamma_2[x \mapsto pc \sqcup t']$.
  - If $x \in Var_{sink}$, we know that $e = x_{temp}$ is the only case in program c' at the $final_{c'}$ section. Because if $\mathcal{P}'(x_{sink}) = \ell'$, then $\forall \ell \in \Gamma'_1(x_{temp}) \cup pc.\ell \in \Gamma'_1(x_{sink}) \wedge \ell \trianglerighteq \ell' \Rightarrow pc \sqcup \Gamma_3(x_{temp}) \sqsubseteq \ell'_{snk} \Rightarrow pc \sqcup \Gamma_3(x_{temp}) \sqsubseteq \Gamma_3(x_{sink})$. Hence, based on the rule (TT-Write-II), $pc \vdash \Gamma_3\{x := e\}\Gamma_3$.
- Case (NT-If): Using the induction hypothesis and $\Gamma'_1 \vdash e : t \Rightarrow \Gamma_2 \vdash e : t'$, the statement $pc \vdash \Gamma_2\{c'\}\Gamma_3$ holds for this case with respect to the rule (TT-If).
- Case (NT-While): Similar to the case (NT-If), and according to the rule (TT-While).
- Case (NT-Seq): Using the induction hypothesis, $pc \vdash \Gamma_2\{c_1\}\Gamma_3$ and $pc \vdash \Gamma_3\{c_2\}\Gamma_4$, then $pc \vdash \Gamma_2\{c_1; c_2\}\Gamma_4$ by using the rule (TT-Seq).
- Case (NT-Sub-II): Using the induction hypothesis, we write $pc_1 \vdash \Gamma_2\{c\}\Gamma'_2$. Since $pc_2 \sqsubseteq pc_1$, $\Gamma_3 \sqsubseteq \Gamma_2$, $\Gamma'_2 \sqsubseteq \Gamma'_3$ and in combination with the rule (NT-Sub-I), $pc_2 \vdash \Gamma_3\{c\}\Gamma'_3$ holds.

□

Proof of Lemma 6. Clearly, the transformation only modifies the labels in the input and output commands of the given program, thus the behavior of the rest of the program stays unaffected. The changes in the labels of the input commands can be formulated as $\forall \ell . I(\ell) = I'(\ell_{src})$, where I is the input for the program c and I' is the input for the program c'.

By induction on the semantic rules shown in Figure 23, it is proven that c' progresses the same as c with the difference that outputs are sent to the channel ℓ_{snk} instead of ℓ. Therefore, we formulate it for the two outputs O and O' of programs c and c' respectively as $O' = O[v_\ell \mapsto v_{\ell_{snk}}]$. Thus the only difference between the output sequences O and O' are the labels of output values; ones with the label ℓ in O are recorded at the same index in O' with the label ℓ_{snk}.

□

Proof of Theorem 12. Let $c' = Transform(c)$, $L_{\mathcal{T}} \supseteq \{\ell_{src}, \ell_{snk} | \ell \in L_{\mathcal{N}}\} \cup \{\top, \bot\}$ and $\forall \ell, \ell' \in L_{\mathcal{N}} . \ell \trianglerighteq \ell' \iff \ell_{src} \sqsubseteq \ell'_{snk}$ ($\trianglerighteq$ is reflexive) such that $\langle L_{\mathcal{T}}, \sqsubseteq \rangle$ is a lattice, and $\forall x \in Var_c . \Gamma_{\mathcal{N}}(x) = \ell \implies \Gamma_{\mathcal{T}}(x) = \ell_{src}$.

1) We have $\forall \ell \in L_{\mathcal{N}} . \forall I_1, I_2 . I_1 \overset{C(\ell)}{=}_{\mathcal{N}} I_2 \iff \forall \ell' \in L_{\mathcal{T}} . \forall I'_1, I'_2 . I'_1 \overset{\ell'}{=}_{\mathcal{T}} I'_2$ because of Definitions 8 and 12. Based on Lemma 6, we also know $\forall \ell \in L_{\mathcal{N}} . I(\ell) = I'(\ell_{src})$.
2) According to Definitions 10 and 14, and the semantic relation presented in Lemma 6, the statement $\forall M . \Big(\forall \ell \in L_{\mathcal{N}} . \forall I_1, I_2 . \big(I_1 \overset{C(\ell)}{=}_{\mathcal{N}} I_2 \wedge \langle c, M, I_1, \emptyset \rangle \rightsquigarrow O_1\big) \implies \exists O_2 . \langle c, M, I_2, \emptyset \rangle \rightsquigarrow O_2 \wedge O_1 \overset{\ell}{=}_{\mathcal{N}} O_2\Big) \iff \Big(\forall \ell' \in L_{\mathcal{T}} . \forall I'_1, I'_2 . \big(I'_1 \overset{\ell'}{=}_{\mathcal{T}} I'_2 \wedge \langle c', M, I'_1, \emptyset \rangle \rightsquigarrow O'_1\big) \implies \exists O'_2 . \langle c', M, I'_2, \emptyset \rangle \rightsquigarrow O'_2 \wedge O'_1 \overset{\ell'}{=}_{\mathcal{T}} O'_2\Big)$ holds if $O_1 \overset{\ell}{=}_{\mathcal{N}} O_2 \iff O'_1 \overset{\ell'}{=}_{\mathcal{T}} O'_2$.
3) As stated in Lemma 6, we conclude $O'_1 = O_1[v_\ell \mapsto v_{\ell_{snk}}] \wedge O'_2 = O_2[v_\ell \mapsto v_{\ell_{snk}}]$. Hence, $O_1 \overset{\ell}{=}_{\mathcal{N}} O_2 \iff O'_1 \overset{\ell'}{=}_{\mathcal{T}} O'_2$ and consequently, the theorem holds.

□