Dynamic Noninterference Analysis Using Context Sensitive Static Analyses

Gurvan Le Guernic

Dynamic Noninterference Analysis Using Context Sensitive Static Analyses

Gurvan Le Guernic

July 14, 2007

Abstract

This report proposes a dynamic noninterference analysis for sequential programs. This analysis is well-suited for the development of a monitor enforcing the absence of information flows between the secret inputs and the public outputs of a program. This implies a sound *detection* of information flows and a sound *correction* of forbidden flows during the execution. The monitor relies on a dynamic information flow analysis. For unexecuted pieces of code, this dynamic analysis uses any context sensitive static information flow analysis which respects a given set of three hypotheses. The soundness of the overall monitoring mechanism with regard to noninterference enforcement is proved, as well as its higher precision than the mechanism proposed in previous work [Le Guernic et al., 2006a].

Contents

1 Introduction **4**
- 1.1 The Language: Syntax & Standard Semantics . . . 5
  - 1.1.1 Syntax of the Language . . . 5
  - 1.1.2 Standard Semantics of the language . . . 6
- 1.2 Monitoring Principles . . . 6

2 The Monitoring Semantics **8**
- 2.1 A semantics Making Use of a Context-sensitive Static Analysis . . . 8
  - 2.1.1 Which static information flow analyses can be used? . . . 11
- 2.2 A Finer Characterization of *Usable* Static Analyses . . . 12

3 Example of Monitored Execution **14**
- 3.1 What happens when variable h is `true`? . . . 15
- 3.2 What happens when variable h is `false`? . . . 16
- 3.3 Conclusion . . . 17

4 Properties of the Monitoring Semantics **18**
- 4.1 Soundness . . . 18
- 4.2 Precision . . . 19

5 Conclusion **20**

A Proofs of the Theorems **21**
- A.1 Usable Analyses . . . 21
  - A.1.1 Hypothesis 2.1 . . . 21
  - A.1.2 Hypothesis 2.2 . . . 25
- A.2 Soundness . . . 35
  - A.2.1 Detection . . . 35
  - A.2.2 Correction . . . 44
- A.3 Transparency . . . 51
  - A.3.1 Lemmas and Figure Reused from Earlier Work . . . 51
  - A.3.2 Proofs of the Partial Transparency . . . 51

Bibliography **61**

1 Introduction

This report presents a secure information flow monitoring mechanism for sequential programs. In previous work [Le Guernic et al., 2006a, Le Guernic, 2007a], the monitoring mechanism is implemented by a security automaton. This automaton, analyzing the information flows during the execution, is separated from the semantics actually executing the monitored program. The automaton and the semantics communicate during the execution in order to enforce the confidentiality of secret data manipulated. However, the automaton and the semantics have their own distinct states. The security automaton is unaware of the content of the current program state, and thus of the precise values of the variables. This approach has the advantage to clearly distinguish the features which are part of the dynamic information flow analysis from those which are part of the program computation. However, this approach has also an impact on the achievable precision of the information flow analysis.

In the program of Figure 1, h is the only private input and l is the only public input. This program initializes the value of the internal variable x to `0`; then, depending on the values of the inputs h and l, it sets x to 1; and finally, it outputs the value of x. This pro-

```
1 x := 0; y := 1;
2 if h then
3   skip
4 else
5   if l then x := y else skip end
6 end;
7 output x
```

Figure 1: Example of a program where context sensitivity is important.

gram is obviously interfering: if l is true, then it outputs `0` if h is true and 1 otherwise. However, executions where l is false are noninterfering; the program always outputs `0`. The monitoring mechanisms presented in previous work [Le Guernic et al., 2006a, Le Guernic, 2007a] are unable to achieve a level of precision which allows to detect that executions where l is false are noninterfering. This is because the static analyses used to analyze unexecuted branches of conditionals whose test carries variety are not context sensitive. When h is true, the program "**if** l **then** x **:=**`1` **else skip end**" is statically analyzed to determine possible implicit indirect flows. In previous work [Le Guernic et al., 2006a, Le Guernic, 2007a], this analysis is done without any knowledge about the value of the input l. Therefore, the static analysis must take into consideration that x may be assigned in an execution where h has a different (`false`) value and l has the same value (even if the current value of l is false). Moreover, the dynamic analysis must compute the same information flows whatever the value of private inputs in order to prevent the creation of a new covert channel by the correction mechanism. Therefore, even if h is false, and then line 5 is executed, the monitoring semantics can not use its knowledge of the value of l to prevent the analysis of a branch which can in no way be executed by any execution started with the same public inputs.

This report presents a monitoring mechanism able to accurately deal with the above example. To do so, implicit indirect flows are taken into account by the dynamic analysis using the results of context-sensitive static analyses of unexecuted pieces of code. The context sensitivity of the static analysis allows it to avoid analyzing pieces of code

which will never be executed in any context low-equivalent to the current one — i.e. contexts in which public data have the same values as in the current context. For example, a context-sensitive analysis of the 5th line of Figure 1 in a context where `l` is `false`, and considered public data, does not analyze the then-branch of the conditional.

The dynamic information flow analysis described in this report is not limited to the use of a sole context-sensitive static analysis. A whole series of context-sensitive static analyses, characterized by a set of axioms, can be used in combination with the dynamic analysis to accurately track information flows. Those axioms emphasize the two main requirements for a noninterference monitor to *accurately detect* information flows and *safely enforce* noninterference.

Section 1.1 describes the syntax and semantics of the sequential imperative language studied. Section 1.2 formally defines the execution property enforced by the monitor and introduces the monitoring principals, which are then formalized in Section 2. Before stating and proving the main properties of the monitoring mechanism in Section 4, an example of monitored execution is detailed in Section 3. Then this report concludes in Section 5.

1.1 The Language: Syntax & Standard Semantics

The language used to describe programs studied in this report is similar to the one of previous work [Le Guernic et al., 2006a]. It is an imperative language for sequential programs. This report focuses on the increase of precision in the dynamic information flow analysis and not on the correction mechanism. Therefore, for simplicity, the language does not include an output statement.

1.1.1 Syntax of the Language

The grammar of the language studied is given in Figure 2. In this grammar, ⟨*ident*⟩ stands for a variable identifier. ⟨*expr*⟩ is an expression of values and variable identifiers. Expressions in this language are deterministic – their evaluation in a given program state always result in the same value — and are free of side effects — their evaluation has no influence on the program state.

⟨*prog*⟩ ::= **skip**
| ⟨*ident*⟩ **:=** ⟨*expr*⟩
| ⟨*prog*⟩ ; ⟨*prog*⟩
| **if** ⟨*expr*⟩ **then** ⟨*prog*⟩ **else** ⟨*prog*⟩ **end**
| **while** ⟨*expr*⟩ **do** ⟨*prog*⟩ **done**

Figure 2: Grammar of the language

A program expressed with this language is either a skip statement (**skip**) which has no effect, an assignment of the value of an expression to a variable (⟨*ident*⟩ **:=** ⟨*expr*⟩), a sequence of programs (⟨*prog*⟩ ; ⟨*prog*⟩), a conditional executing one program — out of two — depending on the value of a given expression (**if** ⟨*expr*⟩ **then** ⟨*prog*⟩ **else** ⟨*prog*⟩ **end**),

or a loop executing repetitively a given program as long as a given expression is true (**while** ⟨*expr*⟩ **do** ⟨*prog*⟩ **done**).

1.1.2 Standard Semantics of the language

The standard semantics of the language is given in Figure 3. The evaluation symbol ($\Downarrow$) is given a subscript letter in order to distinguish between the standard semantics ($\mathcal{S}$) and the monitoring one ($\mathcal{M}$). The standard semantics is based on rules written in the format:

$$\sigma \vdash P \Downarrow_{\mathcal{S}} \sigma'$$

Those rules means that, with the initial program state σ, the evaluation of the program P yields the final program state σ'. Let $\mathbb{X}$ be the domain of variable identifiers and $\mathbb{D}$ be the semantics domain of values. A program state is a value store σ ($\mathbb{X} \to \mathbb{D}$) mapping variable identifiers to their respective value. The definition of value stores is extended to expressions, so that $\sigma(e)$ is the value of the expression e in the program state σ.

$$\frac{}{\sigma \vdash \textbf{skip} \Downarrow_{\mathcal{S}} \sigma} \quad (\mathrm{E}_{\mathcal{S}}\text{-SKIP})$$

$$\frac{}{\sigma \vdash x := e \Downarrow_{\mathcal{S}} \sigma[x \mapsto \sigma(e)]} \quad (\mathrm{E}_{\mathcal{S}}\text{-ASSIGN})$$

$$\frac{\sigma \vdash P^{\mathrm{h}} \Downarrow_{\mathcal{S}} \sigma^{\mathrm{h}} \qquad \sigma^{\mathrm{h}} \vdash P^{\mathrm{t}} \Downarrow_{\mathcal{S}} \sigma^{\mathrm{t}}}{\sigma \vdash P^{\mathrm{h}} ; P^{\mathrm{t}} \Downarrow_{\mathcal{S}} \sigma^{\mathrm{t}}} \quad (\mathrm{E}_{\mathcal{S}}\text{-SEQUENCE})$$

$$\frac{\sigma(e) = v \qquad \sigma \vdash P^{v} \Downarrow_{\mathcal{S}} \sigma'}{\sigma \vdash \textbf{if } e \textbf{ then } P^{\mathtt{true}} \textbf{ else } P^{\mathtt{false}} \textbf{ end} \Downarrow_{\mathcal{S}} \sigma'} \quad (\mathrm{E}_{\mathcal{S}}\text{-IF})$$

$$\frac{\sigma(e) = \mathtt{true} \qquad \sigma \vdash P^{\mathrm{l}} ; \textbf{ while } e \textbf{ do } P^{\mathrm{l}} \textbf{ done} \Downarrow_{\mathcal{S}} \sigma'}{\sigma \vdash \textbf{while } e \textbf{ do } P^{\mathrm{l}} \textbf{ done} \Downarrow_{\mathcal{S}} \sigma'} \quad (\mathrm{E}_{\mathcal{S}}\text{-WHILE}_{\mathtt{true}})$$

$$\frac{\sigma(e) = \mathtt{false}}{\sigma \vdash \textbf{while } e \textbf{ do } P^{\mathrm{l}} \textbf{ done} \Downarrow_{\mathcal{S}} \sigma} \quad (\mathrm{E}_{\mathcal{S}}\text{-WHILE}_{\mathtt{false}})$$

Figure 3: Rules of the standard semantics

1.2 Monitoring Principles

The remainder of the current section gives some formal definitions which are used to formalize the execution property enforced by the monitoring mechanism. Subsequently, it introduces succinctly the mechanisms used by the monitoring semantics in order to enforce secure information flows.

A "safe" execution is a noninterfering execution. Unlike languages of previous work [Le Guernic et al., 2006a, Le Guernic, 2007a], the language studied in this report does not include an output statement. Therefore, in this report, noninterference is not defined as the absence of strong dependencies [Cohen, 1977] between the secret (or private) inputs and a sequence of outputs, as done in the two previous works. Rather, as commonly done, noninterference is defined as the absence of strong dependencies between the secret inputs of an execution and the final values of some variables which are considered to be publicly observable at the end of the execution.

For every execution of a given program P, two sets of variable identifiers are defined. The set of variables corresponding to the secret inputs of the program is designated by $\mathcal{S}(\mathtt{P})$. The set of variables whose final value are publicly observable at the end of the execution is designated by $\mathcal{O}(\mathtt{P})$. No requirements are put on $\mathcal{S}(\mathtt{P})$ and $\mathcal{O}(\mathtt{P})$ other than requiring them to be subsets of $\mathbb{X}$. A variable x is even allowed to belong to both sets. In such a case, in order to be noninterfering, the program P would be required to, at least, reset the value of x.

In the following definitions, we consider that a program state may contain more than just a value store. This is the reason why a distinction is done between program states (X) and value stores (σ). Following Definition 1.1, two program states X_1, respectively X_2, containing the value stores σ_1, respectively σ_2, are *low equivalent* with regards to a set of variables V, written $X_1 \overset{V}{=} X_2$, if and only if the value of any variable belonging to V is the same in σ_1 and σ_2.

Definition 1.1 (Low Equivalent States).
Two states X_1, respectively X_2, containing the value stores σ_1, respectively σ_2, are low equivalent *with regards to a set of variables V, written $X_1 \overset{V}{=} X_2$, if and only if the value of any variable belonging to V is the same in σ_1 and σ_2:*

$$X_1 \overset{V}{=} X_2 \quad \Longleftrightarrow \quad \forall x \in V : \sigma_1(x) = \sigma_2(x)$$

Definition 1.2 (Noninterfering Execution).
Let $\Downarrow_s$ denote a big-step semantics. Let $\mathcal{S}(\mathtt{P})^c$ be the complement of $\mathcal{S}(\mathtt{P})$ in the set $\mathbb{X}$. For all programs P, program states X_1 and X'_1, an execution with the semantics $\Downarrow_s$ of the program P in the initial state X_1 and yielding the final state X'_1 is noninterfering, written $ni(\mathtt{P}, s, X_1)$, if and only if, for every program states X_2 and X'_2 such that the execution with the semantics $\Downarrow_s$ of the program P in the initial state X_2 yields the final state X'_2:

$$X_1 \overset{\mathcal{S}(\mathtt{P})^c}{=} X_2 \;\Rightarrow\; X'_1 \overset{\mathcal{O}(\mathtt{P})}{=} X'_2$$

The monitoring mechanism is based on a flow and context sensitive static analysis. As with the automaton-based noninterference monitors of previous work [Le Guernic et al., 2006a, Le Guernic, 2007a], the monitoring semantics treats directly the direct and explicit indirect flows. For implicit indirect flows, a static analysis is run on the unexecuted branch of every conditional whose test carries variety — i.e. is influenced by the secret inputs of the program.

The static analysis is context sensitive. An unexecuted branch P is analyzed in the context of the program state at the time the test of the conditional, to which P belongs, has been evaluated. The static analysis is then aware of the exact value of the variables which do not carry variety. During the analysis, the context is modified to reflect loss of knowledge. The static analysis does not compute the values of variables. Therefore,

when analyzing an assignment to a variable x, the context of the static analysis is modified to reflect the fact that the static analysis does not anymore have knowledge of the precise value of the variable x. When analyzing a conditional whose test value can be computed using the current context, only the branch designated by the test is analyzed. As the value of any variable which does not carry variety depends only on the public inputs, branches which are not designated by the test value would never be executed by any execution started with the same public inputs as the monitored execution. As the static analysis detects implicit indirect flows more accurately than context insensitive analyses, explicit indirect flows can also be treated more accurately. Implicit indirect flows and explicit indirect flows must be treated with the same precision in order to prevent the creation of a new covert channel [Le Guernic and Jensen, 2005].

For the example on page 4, if h is true then "**if** l **then** x **:=**1 **else skip end**" is analyzed using as context the program state before the evaluation of line 2. As l is a public input, it does not carry variety and the static analysis can take into consideration its value in the provided context. If l is true then the static analysis would analyze the assignment and conclude that there is an implicit indirect flow from h to x. Otherwise, the static analysis would only analyze the statement "**skip**". It would then accurately conclude that there is no "bad" flows in an execution where l is false.

The next section formalizes the mechanisms presented above. It starts by presenting a monitoring semantics making use of static information flow analyses. Then, it describes which static analyses can be used in combination with this monitoring semantics.

2 The Monitoring Semantics

In previous works, the dynamic information flow analysis is defined independently of the monitoring semantics. Therefore, the static analysis used by the dynamic analysis on unexecuted branches can not be sensitive to the context of the monitoring semantics (which is inaccessible to the dynamic analysis). In this report, the dynamic information flow and the monitoring semantics are intrinsically linked. This allows the analysis of unexecuted branches with a static analysis sensitive to the context of execution. The dynamic information flow analysis and the monitoring semantics are defined together in Figure 4.

In previous works, information flows are tracked using a set of variables. At any execution step, this set contains the variables that may carry *variety* — i.e. whose value may be influenced by the secret inputs of the program. In this report, information flows are tracked using tags. At any execution step, every variable has a tag which reflects the fact that this variable may carry variety or not.

2.1 A semantics Making Use of a Context-sensitive Static Analysis

Let $\mathbb{X}$ be the domain of variable identifiers, $\mathbb{D}$ be the semantics domain of values, and $\mathbb{T}$ be the domain of tags. In the remainder of this report, $\mathbb{T}$ is equal to $\{\top, \bot\}$. Those tags form a lattice such that $\bot \sqsubset \top$. $\top$ is the tag associated to variables that *may* carry variety.

The monitoring semantics described in Figure 4 is based on rules written in the format:

$$\zeta,\ t^{\mathrm{pc}} \ \vdash\ \mathtt{P} \ \Downarrow_{\mathcal{M}}\ \zeta'$$

This reads as follows: in the monitoring execution state ζ, with a program counter tag equal to t^{pc}, program P yields the monitoring execution state ζ'. The program counter tag (t^{pc}) is a tag which reflects the security level of the information carried by the control flow. A monitoring execution state ζ is a pair $(\sigma,\ \rho)$ composed of a value store σ and a tag store ρ. A value store ($\mathbb{X} \to \mathbb{D}$) maps variable identifiers to values. A tag store ($\mathbb{X} \to \mathbb{T}$) maps variable identifiers to tags. The definitions of value store and tag store are extended to expressions. $\sigma(e)$ is the value of the expression e in a program state whose value store is σ. Similarly, $\rho(e)$ is the tag of the expression e in a program state whose tag store is ρ. $\rho(e)$ is formally defined as follows, with $\mathcal{V}(e)$ being the set of free variables appearing in the expression e:

$$\rho(e) = \bigsqcup_{x \in \mathcal{V}(e)} \rho(x)$$

The semantics rules make use of static analyses results. In Figure 4, application of a static information flow analysis to the piece of code P in the context ζ is written: $[\![\zeta \vdash \mathtt{P}]\!]^{\sharp_{\mathcal{G}}}$. The analysis of a program P in a monitoring execution state ζ must return a pair $(\mathfrak{D}, \mathfrak{X})$. $\mathfrak{D}$, which belongs to $\mathcal{P}(\mathbb{X} \to \mathbb{X})$, is an over-approximation of the dependencies between the initial and final values of the variables created by an execution of P in the context ζ. $\mathfrak{D}(x)$, which is equal to $\{y \| (x, y) \in \mathfrak{D}\}$, is the set of variables whose initial value may influence the final value of x after execution of P in the context ζ. $\mathfrak{X}$, which belongs to $\mathcal{P}(\mathbb{X})$, is an over-approximation of the set of variables which are potentially defined in an execution of P in the context ζ. This static information flow analysis can be any such analysis that satisfies a set of formal constraints which are stated below.

For example, let $\mathtt{P}^5$ be the 5^{th} line of the program in Figure 1. $(\emptyset,\ \emptyset)$ is a valid result of the static analysis of $\mathtt{P}^5$ in a context where l is `false` and does not carry variety. However, it is not a valid result in a context where l is `true` or may carry variety. In such a context, $(\{(x, y), (x, l)\},\ \{x\})$ is a valid result of the static analysis of $\mathtt{P}^5$.

The monitoring semantics rules are straightforward. As can be expected, the execution of a **skip** statement with the semantics given in Figure 4 yields a final state equal to the initial state. The monitored execution of the assignment of the value of the expression e to the variable x yields a monitored execution state $(\sigma',\ \rho')$. The final value store (σ') is equal to the initial value store (σ) except for the variable x. The final value store maps the variable x to the value of the expression e evaluated with the initial value store ($\sigma(e)$). Similarly, the final tag store (ρ') is equal to the initial tag store (ρ) except for the variable x. The tag of x after the execution of the assignment is the least upper bound of the program counter tag (t^{pc}) and the tag of the expression computed using the initial tag store ($\rho(e)$). $\rho(e)$ represent the level of the information flowing into x through direct flows. t^{pc} represent the level of the information flowing into x through explicit indirect flows.

The monitored execution of a conditional whose test expression does not carry variety ($\rho(e) = \bot$) follows the same scheme as with a standard semantics. For a conditional whose test expression e carries variety, the branch (P^v) designated by the value of e is executed and the other one ($P^{\neg v}$) is analyzed. The final value store is the one returned by the execution of P^v. The final tag store (ρ') is the least upper bound of the tag store returned by the execution of P^v and two new tag stores ($\rho^{\neg v}$ and ρ^e) generated from the result of the analysis of $P^{\neg v}$ ($(\mathfrak{D}, \mathfrak{X})$). By definition, $\rho \sqcup \rho'$ is equal to $\lambda x.\rho(x) \sqcup \rho'(x)$. The first new tag store ($\rho^{\neg v}$) is computed from the dependencies ($\mathfrak{D}$), were $P^{\neg v}$ executed, between the final value of variables in $P^{\neg v}$ and their initial values. In $\rho^{\neg v}$, the tag

$$\frac{}{\zeta,\ t^{\mathrm{pc}} \ \vdash\ \textbf{skip}\ \Downarrow_{\mathcal{M}}\ \zeta} \qquad (\mathrm{E}_{\mathcal{M}}\text{-SKIP})$$

$$\frac{}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ x := e\ \Downarrow_{\mathcal{M}}\ (\sigma[x \mapsto \sigma(e)],\ \rho[x \mapsto \rho(e) \sqcup t^{\mathrm{pc}}])} \qquad (\mathrm{E}_{\mathcal{M}}\text{-ASSIGN})$$

$$\frac{\zeta,\ t^{\mathrm{pc}} \ \vdash\ P^{\mathrm{h}}\ \Downarrow_{\mathcal{M}}\ \zeta^{\mathrm{h}} \qquad \zeta^{\mathrm{h}},\ t^{\mathrm{pc}} \ \vdash\ P^{\mathrm{t}}\ \Downarrow_{\mathcal{M}}\ \zeta^{\mathrm{t}}}{\zeta,\ t^{\mathrm{pc}} \ \vdash\ P^{\mathrm{h}}\ ;\ P^{\mathrm{t}}\ \Downarrow_{\mathcal{M}}\ \zeta^{\mathrm{t}}} \qquad (\mathrm{E}_{\mathcal{M}}\text{-SEQUENCE})$$

$$\frac{\rho(e) = \bot \qquad \sigma(e) = v \qquad (\sigma,\ \rho),\ t^{\mathrm{pc}} \sqcup \bot \ \vdash\ P^{v}\ \Downarrow_{\mathcal{M}}\ \zeta'}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{if}\ e\ \textbf{then}\ P^{\mathtt{true}}\ \textbf{else}\ P^{\mathtt{false}}\ \textbf{end}\ \Downarrow_{\mathcal{M}}\ \zeta'} \qquad (\mathrm{E}_{\mathcal{M}}\text{-IF}_{\bot})$$

$$\frac{\begin{array}{c}\rho(e) = \top \qquad \sigma(e) = v \\ (\sigma,\ \rho),\ t^{\mathrm{pc}} \sqcup \top \ \vdash\ P^{v}\ \Downarrow_{\mathcal{M}}\ (\sigma^{v},\ \rho^{v}) \qquad [\![(\sigma,\ \rho) \vdash P^{\neg v}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X}) \\ \rho^{\neg v} = \lambda x.\ \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) \qquad \rho^{\mathrm{e}} = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{X} - \mathfrak{X}) \times \{\bot\})\end{array}}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{if}\ e\ \textbf{then}\ P^{\mathtt{true}}\ \textbf{else}\ P^{\mathtt{false}}\ \textbf{end}\ \Downarrow_{\mathcal{M}}\ (\sigma^{v},\ \rho^{v} \sqcup \rho^{\neg v} \sqcup \rho^{\mathrm{e}})} \qquad (\mathrm{E}_{\mathcal{M}}\text{-IF}_{\top})$$

$$\frac{\rho(e) = \bot \qquad \sigma(e) = \mathtt{false}}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ (\sigma,\ \rho)} \qquad (\mathrm{E}_{\mathcal{M}}\text{-WHILE}_{skip})$$

$$\frac{\begin{array}{c}\rho(e) = \bot \qquad \sigma(e) = \mathtt{true} \\ (\sigma,\ \rho),\ t^{\mathrm{pc}} \sqcup \bot \ \vdash\ P^{\mathrm{l}}\ ;\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ \zeta'\end{array}}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ \zeta'} \qquad (\mathrm{E}_{\mathcal{M}}\text{-WHILE}_{true_{\bot}})$$

$$\frac{\begin{array}{c}\rho(e) = \top \qquad \sigma(e) = \mathtt{true} \\ (\sigma,\ \rho),\ t^{\mathrm{pc}} \sqcup \top \ \vdash\ P^{\mathrm{l}}\ ;\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ (\sigma',\ \rho')\end{array}}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ (\sigma',\ \rho \sqcup \rho')} \qquad (\mathrm{E}_{\mathcal{M}}\text{-WHILE}_{true_{\top}})$$

$$\frac{\begin{array}{c}\rho(e) = \top \qquad \sigma(e) = \mathtt{false} \\ [\![(\sigma,\ \rho) \vdash P^{\mathrm{l}}\ ;\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X}) \\ \rho^{\mathrm{l}} = \lambda x.\ \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) \qquad \rho^{\mathrm{e}} = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{X} - \mathfrak{X}) \times \{\bot\})\end{array}}{(\sigma,\ \rho),\ t^{\mathrm{pc}} \ \vdash\ \textbf{while}\ e\ \textbf{do}\ P^{\mathrm{l}}\ \textbf{done}\ \Downarrow_{\mathcal{M}}\ (\sigma,\ \rho \sqcup \rho^{\mathrm{l}} \sqcup \rho^{\mathrm{e}})} \qquad (\mathrm{E}_{\mathcal{M}}\text{-WHILE}_{false_{\top}})$$

Figure 4: Rules of the monitoring semantics

of a variable x is the least upper bound of the initial tags ($\rho(y)$) of the variables whose initial value may influence the final value of x in an execution of $P^{\neg\nu}$. The other new tag store (ρ^e) reflects the implicit indirect flows between the value of the test of the conditional and the variables ($\mathfrak{X}$) which may be defined in an execution of $P^{\neg\nu}$. In ρ^e, the tag of a variable x is equal to the initial tag of the test expression of the conditional ($\rho(e)$) if and only if x belongs to $\mathfrak{X}$; otherwise, its tag is $\bot$.

2.1.1 Which static information flow analyses can be used?

The static analysis used to determine the information flows in unexecuted branches is not formally defined. In fact, the dynamic analysis can use any static information flow analysis which complies with the three following hypotheses and returns a pair, whose first element is a relation between variables — i.e. a set of pairs of variables — and second element is a set of variables.

The first hypothesis ensures an "exact" detection of information flows. Hypothesis 2.1 simply requires the static analysis used to be a *sound* information flow analysis. More precisely, it requires that the second element of the static analysis result ($\mathfrak{X}$) contains all the variables which may be defined by an execution of the analyzed program with the same public values as those used for the analysis. This is a straightforward requirement as the result of the static analysis is used to take into account implicit indirect flows.

For example, in a context where l is `true` or may carry variety ($\rho(l) = \top$), Hypothesis 2.1 requires that x *must* belong to the second element ($\mathfrak{X}$) of the result of any analysis of $\mathtt{P}^5$ (the 5^{th} line of the program in Figure 1).

Hypothesis 2.1 (Correctness of the static analysis for information flow detection).

For all monitoring execution states $(\sigma,\ \rho)$, $(\sigma_i,\ \rho_i)$ *and* $(\sigma_o,\ \rho_o)$, *program counter tags* t^{pc}, *programs* P, *and analysis results* $(\mathfrak{D}, \mathfrak{X})$ *such that:*

1. $\forall x : (\rho(x) = \bot) \Rightarrow \sigma_i(x) = \sigma(x)$,
2. $(\sigma_i,\ \rho_i),\ t^{pc} \vdash P \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$,

the following holds:

$$[\![(\sigma,\ \rho) \vdash P]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X}) \quad \Rightarrow \quad \forall x \notin \mathfrak{X}.\ \sigma_o(x) \ =\ \sigma_i(x)$$

The second and third hypotheses allow a "safe" correction of information flows. Hypothesis 2.2 requires the static information flow analysis to be as precise as the dynamic information flow analysis when the program counter carries variety. This is required in order to be able to correct "bad" flows. If this hypothesis does not hold then it may be possible for the attacker to deduce from the behavior of the correction mechanism which branch of a conditional whose test carries variety has been analyzed and whose branch has been executed. This information would allow the attacker to deduce the value of the test of the conditional.

Hypothesis 2.2 (Correctness of the static analysis for information flow correction).

For all monitoring execution states $(\sigma,\ \rho)$ *and* $(\sigma',\ \rho')$, *programs* P, *and analysis results* $(\mathfrak{D}, \mathfrak{X})$ *such that:*

1. $(\sigma,\ \rho),\ \top\ \vdash\ P\ \Downarrow_{\mathcal{M}}\ (\sigma',\ \rho')$,

2. $[\![(\sigma,\ \rho) \vdash P]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$,

the following holds:

$$\rho' \quad = \quad (\ \lambda x.\ \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)\ ) \quad \sqcup \quad (\ (\mathfrak{X} \times \{\top\})\ \cup\ (\mathfrak{X}^c \times \{\bot\})\ )$$

The last hypothesis (Hypothesis 2.3) requires the result of the static analysis to be determined solely by the value of public data. This hypothesis is needed for a similar reason than for the second hypothesis. In order to prevent the correction mechanism to be used as a covert channel, the result of the static analysis must be independent from the value of secret data.

For example, Hypothesis 2.3 requires that the result of the analysis of the 5$^{\text{th}}$ line of the program in Figure 1 must depend only on the values of variables *l*, *x* and *y*; and not on the value of variable *h*.

Hypothesis 2.3 (Static analysis is deterministic with regard to public data).
For all monitoring execution states $(\sigma,\ \rho)$ *and* $(\sigma',\ \rho')$*, and programs* P *such that:*

1. $\rho = \rho'$,

2. $\forall x : (\rho(x) \sqsubseteq \bot) \Rightarrow \sigma(x) = \sigma'(x)$,

the following holds:

$$[\![(\sigma,\ \rho) \vdash P]\!]^{\sharp_{\mathcal{G}}} \ = \ [\![(\sigma',\ \rho') \vdash P]\!]^{\sharp_{\mathcal{G}}}$$

The fact that the static analysis is deterministic with regard to public data is not strictly required. The least requirement is that the set of potential results must be determined by the value of public data. For simplicity, this hypothesis also requires the analysis to be deterministic with regard to public data..

2.2 A Finer Characterization of *Usable* Static Analyses

The above hypotheses define which static information flow analyses are *usable*, i.e. which static analyses can be used with the monitoring semantics given in Figure 4. However, Hypotheses2.1 and 2.2 are stated using the monitoring semantics itself. This make it more difficult to prove that a given static analysis satisfies those hypotheses.

Figure 5 defines a set of *acceptability* rules. The result $(\mathfrak{D}, \mathfrak{X})$ of a static information flow analysis of a given program (P) in a given context (ζ) is *acceptable* for the monitoring semantics only if the result satisfies those rules. This is written in the format:

$$(\mathfrak{D}, \mathfrak{X}) \models (\zeta \vdash \mathrm{P})$$

In the definitions of those rules, $\mathcal{I}d$ denotes the identity relation. $\circ$ is the operation of composition of relations. It belongs to $(\mathcal{P}(\mathbb{B} \times \mathbb{C}) \times \mathcal{P}(\mathbb{A} \times \mathbb{B})) \rightarrow \mathcal{P}(\mathbb{A} \times \mathbb{C})$ with $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ being sets. Its formal definition follows.

$$(S \circ R) = \bigcup_{(a,b) \in R} \{(a, c) \mid (b, c) \in S\}$$

As used in the acceptability rules, $\mathbb{A}$, $\mathbb{B}$ and $\mathbb{C}$ are equal to the set of variable identifiers $\mathbb{X}$. $[\![S^{\mathtt{true}}, S^{\mathtt{false}}]\!]^t_v$ returns either the set $S^{\mathtt{true}}$, the set $S^{\mathtt{false}}$ or the union of both depending on the value of the tag t and the value v. Its formal definition follows.

$$[\![S^{\mathtt{true}}, S^{\mathtt{false}}]\!]^t_v = \begin{cases} S^{\mathtt{true}} \cup S^{\mathtt{false}} & \text{iff } t = \top \\ S^v & \text{iff } t = \bot \end{cases}$$

$(\mathfrak{D}, \mathfrak{X}) \models (\ (\sigma,\ \rho) \vdash \textbf{skip}\ )$
iff $\mathfrak{D} = \mathcal{I}d \quad \wedge \quad \mathfrak{X} = \emptyset$

$(\mathfrak{D}, \mathfrak{X}) \models (\ (\sigma,\ \rho) \vdash x := e\ )$
iff $\mathfrak{D} = \mathcal{I}d[x \mapsto \mathcal{V}(e)] \quad \wedge \quad \mathfrak{X} = \{x\}$

$(\mathfrak{D}, \mathfrak{X}) \models \left(\ (\sigma,\ \rho) \vdash \mathrm{P}^{\mathrm{h}}\ ;\ \mathrm{P}^{\mathrm{t}}\ \right)$
iff there exist $(\mathfrak{D}^{\mathrm{h}}, \mathfrak{X}^{\mathrm{h}})$ and $(\mathfrak{D}^{\mathrm{t}}, \mathfrak{X}^{\mathrm{t}})$ such that:
$(\mathfrak{D}^{\mathrm{h}}, \mathfrak{X}^{\mathrm{h}}) \models ((\sigma,\ \rho) \vdash \mathrm{P}^{\mathrm{h}})$
let $\rho' = \rho \sqcup ((\mathfrak{X}^{\mathrm{h}} \times \top) \cup ((\mathfrak{X}^{\mathrm{h}})^{\mathrm{c}} \times \bot))$ in $(\mathfrak{D}^{\mathrm{t}}, \mathfrak{X}^{\mathrm{t}}) \models ((\sigma, \rho') \vdash \mathrm{P}^{\mathrm{t}})$
$\mathfrak{D} = \mathfrak{D}^{\mathrm{h}} \circ \mathfrak{D}^{\mathrm{t}} \quad \wedge \quad \mathfrak{X} = \mathfrak{X}^{\mathrm{h}} \cup \mathfrak{X}^{\mathrm{t}}$

$(\mathfrak{D}, \mathfrak{X}) \models \left(\ (\sigma,\ \rho) \vdash \textbf{if } e \textbf{ then } \mathrm{P}^{\mathtt{true}} \textbf{ else } \mathrm{P}^{\mathtt{false}} \textbf{ end}\ \right)$
iff there exist $(\mathfrak{D}^{\mathtt{true}}, \mathfrak{X}^{\mathtt{true}})$ and $(\mathfrak{D}^{\mathtt{false}}, \mathfrak{X}^{\mathtt{false}})$ such that:
$(\mathfrak{D}^{\mathtt{true}}, \mathfrak{X}^{\mathtt{true}}) \models ((\sigma,\ \rho) \vdash \mathrm{P}^{\mathtt{true}})$
$(\mathfrak{D}^{\mathtt{false}}, \mathfrak{X}^{\mathtt{false}}) \models ((\sigma,\ \rho) \vdash \mathrm{P}^{\mathtt{false}})$
$\mathfrak{D} = [\![\mathfrak{D}^{\mathtt{true}}, \mathfrak{D}^{\mathtt{false}}]\!]^{\rho(e)}_{\sigma(e)} \cup ([\![\mathfrak{X}^{\mathtt{true}}, \mathfrak{X}^{\mathtt{false}}]\!]^{\rho(e)}_{\sigma(e)} \times \mathcal{V}(e))$
$\mathfrak{X} = [\![\mathfrak{X}^{\mathtt{true}}, \mathfrak{X}^{\mathtt{false}}]\!]^{\rho(e)}_{\sigma(e)}$

$(\mathfrak{D}, \mathfrak{X}) \models \left(\ (\sigma,\ \rho) \vdash \textbf{while } e \textbf{ do } \mathrm{P}^{\mathrm{l}} \textbf{ done}\ \right)$
iff there exists $(\mathfrak{D}^{\mathrm{l}}, \mathfrak{X}^{\mathrm{l}})$ such that:
let $\rho' = \rho \sqcup ((\mathfrak{X}^{\mathrm{l}} \times \top) \cup ((\mathfrak{X}^{\mathrm{l}})^{\mathrm{c}} \times \bot))$ in $(\mathfrak{D}^{\mathrm{l}}, \mathfrak{X}^{\mathrm{l}}) \models ((\sigma, \rho') \vdash \mathrm{P}^{\mathrm{l}})$
$\mathfrak{D} = [\![\mathfrak{D}^{\mathrm{l}} \circ (\mathfrak{D} \cup \mathcal{I}d), \mathcal{I}d]\!]^{\rho(e)}_{\sigma(e)} \quad \wedge \quad \mathfrak{X} = [\![\mathfrak{X}^{\mathrm{l}}, \emptyset]\!]^{\rho(e)}_{\sigma(e)}$

Figure 5: Acceptability rules for *usable* static information flow analysis results

Using the acceptability rules of Figure 5, it is possible to characterize some static information flow analyses which are *usable* with the monitoring semantics without referring to the monitoring semantics itself. It is also possible to generate a *usable* static information flow analysis by fix-point computation on the acceptability rules; in fact, only on the rule for loop statements. However, those acceptability rules do not define a most precise usable static analysis.

As stated by Theorem 2.1, any *acceptable* static information flow analysis result satisfies Hypothesis 2.1.

Theorem 2.1 (Acceptability rules imply detection abilities).
For all monitoring execution states ζ, programs P, and analysis result $(\mathfrak{D}, \mathfrak{X})$ such that $(\mathfrak{D}, \mathfrak{X}) \models (\zeta \vdash P)$, the Hypothesis 2.1 holds.

Proof. The theorem follows directly from Lemma A.1.

Theorem 2.2 states that any *acceptable* static information flow analysis result, which satisfies Hypothesis 2.4, satisfies Hypothesis 2.2. Hypothesis 2.4 requires the dependencies between initial and final values computed by the analysis to be coherent with the set of potentially defined variables also computed by the analysis. The final value of a variable which may not be defined by a program is required to depend only on its own initial value. Hypothesis 2.4 holds if, but not only if, $\mathfrak{D}$ is the smallest fixed point obtained from the acceptability rules of Figure 5.

Hypothesis 2.4 (Static analysis results are coherent).
For all monitoring execution states (σ, ρ), programs P, and analysis result $(\mathfrak{D}, \mathfrak{X})$:

$$[\![\zeta \vdash P]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X}) \quad \Rightarrow \quad \forall x \in \mathfrak{X}^c. \quad \mathfrak{D}(x) = \{x\}$$

Theorem 2.2 (Acceptability rules imply correction abilities).
For all monitoring execution states (σ, ρ), programs P, and analysis result $(\mathfrak{D}, \mathfrak{X})$ if $(\mathfrak{D}, \mathfrak{X}) \models (\zeta \vdash P)$ and the Hypothesis 2.4 holds then the Hypothesis 2.2 holds.

Proof. The theorem follows directly from Lemma A.4.

Therefore, a static information flow analysis, which satisfies Hypotheses 2.3 and 2.4 and whose results are acceptable ($[\![\zeta \vdash \mathtt{P}]\!]^{\sharp_{\mathcal{G}}} \models (\zeta \vdash \mathtt{P})$), is *usable* by the monitoring semantics — i.e. it satisfies Hypotheses 2.1, 2.2 and 2.3.

Theorem 2.3 (Existence of a compliant static analysis). *There exists a static analysis $[\![]\!]^{\sharp_{\mathcal{G}}}$ such that, for all commands C, value stores σ, and tag stores ρ, there exists an analysis result $(\mathfrak{D}, \mathfrak{X})$ such that $[\![\sigma, \rho \vdash C]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$ and Hypotheses 2.3 and 2.4 hold.*

Proof. The theorem follows directly from the constructive proof of Lemma A.15.

3 Example of Monitored Execution

Figure 6 contains an adaptation of the motivating example given in Section 1. In this program (P), variables h and l contain the program inputs at the beginning of the execution. h contains the only secret input ($\mathcal{S}(\mathtt{P}) = \{h\}$); l contains a public input. The only publicly observable behavior of the program is the value of the variable x at the end of the computation ($\mathcal{O}(\mathtt{P}) = \{x\}$). The final value of x is `1` if h is `false` and l is `true`. Otherwise, the final value of x is `0`. This program is obviously interfering as, if the final value of x is `1`, it is possible to deduce that the secret input h is `true`. However, any execution where l is `false` sets the final value of x to `0`. Therefore, any execution where l is `false` is noninterfering.

As stated in the introduction (Section 1), none of the monitoring mechanisms of previous work [Le Guernic et al., 2006b, Le Guernic, 2007b] are able to detect that executions where l is `false` are noninterfering. The monitor proposed in this report is able to detect it. This section presents the behavior of the semantics presented in Figure 4 on executions of the program in Figure 6.

```
1 x := 0;
2 if h then
3   skip
4 else
5   if l then x := 1 else skip end
6 end
```

Figure 6: The motivating program of the Introduction.

3.1 What happens when variable h is `true`?

Figure 7 presents the evaluation tree for executions where h is `true`. Such executions start by setting the value of x to `0`. Then, as h is `true`, the then-branch (**skip**) is executed and the else-branch (**if** l **then** x **:=** `1` **else skip end**) is analyzed. In Figure 7, the result of the static analysis of the else-branch is represented by the wild card $\mathfrak{R}$.

$$\dfrac{\dfrac{}{\zeta_1,\ \bot\ \vdash\ x := 0\ \Downarrow_{\mathcal{M}}\ \zeta_2} \qquad \dfrac{\dfrac{}{\zeta_2,\ \top\ \vdash\ \textbf{skip}\ \Downarrow_{\mathcal{M}}\ \zeta_2} \qquad [\![\zeta_2 \vdash \mathrm{P}^\sharp]\!]^\sharp_{\mathcal{G}} = \mathfrak{R}}{\zeta_2,\ \bot\ \vdash\ \textbf{if}\ h\ \textbf{then skip else}\ \mathrm{P}^\sharp\ \textbf{end}\ \Downarrow_{\mathcal{M}}\ \zeta_3}}{\zeta_1,\ \bot\ \vdash\ x := 0\ ;\ \textbf{if}\ h\ \textbf{then skip else}\ \mathrm{P}^\sharp\ \textbf{end}\ \Downarrow_{\mathcal{M}}\ \zeta_3}$$

Figure 7: Evaluation tree of the program in Figure 6 for h = `true`.

And variable l is `true`? Table 1 shows the values of the wild cards used in Figure 7 (ζ_1, ζ_2, ζ_3, $\mathrm{P}^\sharp$ and $\mathfrak{R}$) for an execution where the initial value of h is `true`, the initial value of l is `true` and the initial value of x is `3`.

Wild Card	Means	Value
ζ_1	$(\sigma_1,\ \rho_1)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{true},\ x \mapsto \texttt{3}],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \bot]\ )$
ζ_2	$(\sigma_2,\ \rho_2)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{true},\ x \mapsto \texttt{0}],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \bot]\ )$
$\mathrm{P}^\sharp$	**if** l **then** x **:=** `1` **else skip end**	
$\mathfrak{R}$	$(\mathfrak{D}_l, \mathfrak{X}_l)$	$(\mathcal{I}d[x \mapsto \{l\}],\ \{x\})$
ζ_3	$(\sigma_3,\ \rho_3)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{true},\ x \mapsto \texttt{0}],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \top]\ )$

Table 1: Value of the wild cards of Figure 7 for l = `true`.

In the context of executions where the initial value of h is `true` and the initial value of l is `true`, $\mathfrak{R}$ is the result of the analysis of **if** l **then** x **:=** `1` **else skip end** in the context $(\sigma_2,\ \rho_2)$ knowing that $\sigma_2(l)$ is `true` and $\rho_2(l)$ is $\bot$. $\mathfrak{R}$ stands for $(\mathfrak{D}_l, \mathfrak{X}_l)$. This analysis result, which is equal to $(\mathcal{I}d[x \mapsto \{l\}],\ \{x\})$ in our example, satisfies Hypotheses 2.1 and 2.2. Therefore, under the assumption that the static analysis used to compute this result satisfies Hypothesis 2.3, the result $(\mathfrak{D}_l, \mathfrak{X}_l)$ can be used by the dynamic analysis to monitor the information flows. For the purpose of this example, $(\mathfrak{D}_l, \mathfrak{X}_l)$ has been computed by manually solving the constraints implied by the *acceptability* rules of Figure 5. Using $(\mathfrak{D}_l, \mathfrak{X}_l)$ with the rule ($\mathrm{E}_{\mathcal{M}}$-$\mathrm{IF}_\top$), it comes out that $\rho_3(x) = \rho_2(x) \sqcup \rho_2(l) \sqcup \top$.

And variable l is `false`? Table 2 shows the values of the wild cards used in Figure 7 for an execution where the initial value of h is `true`, the initial value of l is `false` and the initial value of x is 3.

Wild Card	Means	Value
ζ_1	$(\sigma_1,\ \rho_1)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{false},\ x \mapsto 3],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \bot]\ )$
ζ_2	$(\sigma_2,\ \rho_2)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{false},\ x \mapsto \texttt{0}],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \bot]\ )$
$\texttt{P}^\sharp$	**if** l **then** x **:=** 1 **else skip end**	
$\mathfrak{R}$	$(\mathfrak{D}_{\neg l}, \mathfrak{X}_{\neg l})$	$(\mathcal{I}d,\ \emptyset)$
ζ_3	$(\sigma_3,\ \rho_3)$	$(\ [h \mapsto \texttt{true},\ l \mapsto \texttt{false},\ x \mapsto \texttt{0}],\quad [h \mapsto \top,\ l \mapsto \bot,\ x \mapsto \bot]\ )$

Table 2: Value of the wild cards of Figure 7 for $l = \texttt{false}$.

In the context of executions where the initial value of h is `true` and the initial value of l is `false`, $\mathfrak{R}$ is the result of the analysis of **if** l **then** x **:=** 1 **else skip end** in the context $(\sigma_2,\ \rho_2)$ knowing that $\sigma_2(l)$ is `false` and $\rho_2(l)$ is $\bot$. $\mathfrak{R}$ stands for $(\mathfrak{D}_{\neg l}, \mathfrak{X}_{\neg l})$. In our example, this analysis result is equal to $(\mathcal{I}d,\ \emptyset)$. Using $(\mathfrak{D}_{\neg l}, \mathfrak{X}_{\neg l})$ with the rule ($\mathrm{E}_\mathcal{M}$-IF$_\top$), it comes out that $\rho_3(x) = \rho_2(x) \sqcup \rho_2(x) \sqcup \bot$.

3.2 What happens when variable h is `false`?

Figure 8 and 9 present the evaluation trees for executions where h is `false`. Such executions start by setting the value of x to `0`. Then, as h is `false`, the then-branch (**skip**) is analyzed and the else-branch (**if** l **then** x **:=** 1 **else skip end**) is executed. The analysis of the then-branch yields an analysis result, $(\mathcal{I}d,\ \emptyset)$, emphasizing the fact that there is no information flow created by a potential execution of the then-branch in a context similar to the current execution. Therefore, the tag store after the execution of the conditional branching on the variable h (ρ_4) is equal to the least upper bound of the tag store after the execution of the else-branch (ρ_3) and the tag store before the execution of the conditional (ρ_2): $\forall x.\, \rho_4(x) = \rho_2(x) \sqcup \rho_3(x)$.

And variable l is `true`? Figure 8 presents the evaluation tree for executions where h is `false` and l is `true`. Table 3 shows the values of the wild cards used in Figure 8 for an execution where the initial value of h is `false`, the initial value of l is `true` and the initial value of x is 3.

$$\dfrac{\dfrac{}{\zeta_1,\ \bot \vdash x := \texttt{0} \Downarrow_\mathcal{M} \zeta_2} \qquad \dfrac{\dfrac{\dfrac{}{\zeta_2,\ \top \vdash x := 1 \Downarrow_\mathcal{M} \zeta_3}}{\zeta_2,\ \top \vdash \texttt{P'} \Downarrow_\mathcal{M} \zeta_3} \quad [\![\zeta_2 \vdash \textbf{skip}]\!]^{\sharp_\mathcal{G}} = (\mathcal{I}d, \emptyset)}{\zeta_2,\ \bot \vdash \textbf{if } h \textbf{ then skip else } \texttt{P'} \textbf{ end} \Downarrow_\mathcal{M} \zeta_4}}{\zeta_1,\ \bot \vdash x := \texttt{0}\ ;\ \textbf{if } h \textbf{ then skip else } \texttt{P'} \textbf{ end} \Downarrow_\mathcal{M} \zeta_4}$$

Figure 8: Evaluation tree of the program in Figure 6 for $h = \texttt{false} \wedge l = \texttt{true}$.

In a context where the initial value of h is `false` and the initial value of l is `true`, the conditional branching on variable l is evaluated with a program counter carrying variety (its tag is $\top$) and yields the assignment "x **:=** 1". This assignment is also evaluated with a program counter tag equal to $\top$. Therefore, after the execution of the assignment, variable x is also considered to carry variety ($\rho_3(x) = \top$).

Wild Card	Means	Value
ζ_1	(σ_1, ρ_1)	$([h \mapsto \texttt{false}, l \mapsto \texttt{true}, x \mapsto 3], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$
ζ_2	(σ_2, ρ_2)	$([h \mapsto \texttt{false}, l \mapsto \texttt{true}, x \mapsto 0], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$
P'	**if** l **then** x **:=** 1 **else skip end**	
ζ_3	(σ_3, ρ_3)	$([h \mapsto \texttt{false}, l \mapsto \texttt{true}, x \mapsto 1], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \top])$
ζ_4	(σ_4, ρ_4)	$([h \mapsto \texttt{false}, l \mapsto \texttt{true}, x \mapsto 1], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \top])$

Table 3: Value of the wild cards of Figure 8.

And variable l is `false`? Figure 9 presents the evaluation tree for executions where h is `false` and l is `false`. Table 4 shows the values of the wild cards used in Figure 9 for an execution where the initial value of h is `false`, the initial value of l is `false` and the initial value of x is 3.

$$\dfrac{\dfrac{}{\zeta_1,\ \bot \vdash x := 0 \Downarrow_{\mathcal{M}} \zeta_2} \qquad \dfrac{\dfrac{\dfrac{}{\zeta_2,\ \top \vdash \textbf{skip} \Downarrow_{\mathcal{M}} \zeta_3}}{\zeta_2,\ \top \vdash \texttt{P'} \Downarrow_{\mathcal{M}} \zeta_3} \quad [\![\zeta_2 \vdash \textbf{skip}]\!]^{\sharp_{\mathcal{G}}} = (\mathcal{I}d, \emptyset)}{\zeta_2,\ \bot \vdash \textbf{if } h \textbf{ then skip else } \texttt{P'} \textbf{ end} \Downarrow_{\mathcal{M}} \zeta_4}}{\zeta_1,\ \bot \vdash x := 0\ ;\ \textbf{if } h \textbf{ then skip else } \texttt{P'} \textbf{ end} \Downarrow_{\mathcal{M}} \zeta_4}$$

Figure 9: Evaluation tree of the program in Figure 6 for $h = \texttt{false} \wedge l = \texttt{false}$.

Wild Card	Means	Value
ζ_1	(σ_1, ρ_1)	$([h \mapsto \texttt{false}, l \mapsto \texttt{false}, x \mapsto 3], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$
ζ_2	(σ_2, ρ_2)	$([h \mapsto \texttt{false}, l \mapsto \texttt{false}, x \mapsto 0], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$
P'	**if** l **then** x **:=** 1 **else skip end**	
ζ_3	(σ_3, ρ_3)	$([h \mapsto \texttt{false}, l \mapsto \texttt{false}, x \mapsto 0], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$
ζ_4	(σ_4, ρ_4)	$([h \mapsto \texttt{false}, l \mapsto \texttt{false}, x \mapsto 0], \quad [h \mapsto \top, l \mapsto \bot, x \mapsto \bot])$

Table 4: Value of the wild cards of Figure 9.

In a context where the initial value of h is `false` and the initial value of l is `false`, the conditional branching on variable l is evaluated with a program counter carrying variety (its tag is $\top$) and yields the statement "**skip**". The unexecuted branch of this conditional ("x **:=** 1") is not analyzed. The reason is that its test (l) does not carry variety. Therefore, the test of the conditional has the same value for any execution started with the same public inputs. Hence, the assignment contained in the unexecuted branch is never executed by any initially low equivalent executions. Consequently, the monitored execution state after the execution of the conditional branching on l (ζ_3) is equal to the monitored execution state before the execution of this conditional (ζ_2).

3.3 Conclusion

Table 5 summarizes the possible final values ($\sigma_f(x)$) and tags ($\rho_f(x)$) of the variable x depending on the initial values of the inputs h ($\sigma_i(h)$) and l ($\sigma_i(l)$). As can be seen in this table, if l is `false` then the final value of x is always `0` independently from the initial value of h. Therefore, whenever l is `false`, there is no flow from h to x. This

$\sigma_i(l)$	$\sigma_i(h)$	$\sigma_f(x)$	$\rho_f(x)$
`true`	`true`	0	$\top$
`true`	`false`	1	$\top$
`false`	`true`	0	$\bot$
`false`	`false`	0	$\bot$

Table 5: Final values and tags of x depending on the initial values of l and h.

is well detected by the dynamic analysis which sets the final tag of x to $\bot$ whenever l is `false`. On the other side, if l is `true` then the final value of x depends on the initial value of h. This is well detected by the dynamic analysis which sets the final tag of x to $\top$ whenever l is `true`. Additionally, the dynamic analysis behaves securely as it does not create any new covert channel. Indeed, the final tag associated to the variable x never depends on the initial value of the only secret input h.

4 Properties of the Monitoring Semantics

Section 2 formally defined the dynamic information flow analysis proposed in this report. The soundness of this analysis with regard to the notion of noninterfering execution (Definition 1.2) is proved in Section 4.1. Section 3 demonstrated on an example that the dynamic analysis proposed in this report is sometimes more precise than the dynamic analysis proposed in previous work [Le Guernic et al., 2006a], and also more precise than the majority of information flow analyses. Section 4.2 proves that the dynamic analysis developed in this report is always at least as precise as the dynamic information flow analysis of previous work [Le Guernic et al., 2006a].

4.1 Soundness

The monitoring mechanism is proved to be sound with regard to the notion of noninterfering execution given in Definition 1.2. This means that, at the end of any two monitored executions of a given program P started with the same public inputs (variables which do not belong to $\mathcal{S}(\mathrm{P})$), the final values of observable outputs (variables which belong to $\mathcal{O}(\mathrm{P})$) are the same for both executions.

Theorem 4.1 proves that the monitoring mechanism is sound with regard to information flow detection. Any variable, whose tag at the end of the execution is $\bot$, has the same final value for any initially low equivalent executions.

Theorem 4.1 (Sound Detection).
Assume that the monitoring semantics $\Downarrow_{\mathcal{M}}$ uses a static information flow analysis ($[\![]\!]^{\sharp_{\mathcal{G}}}$) for which Hypotheses 2.1, 2.2 and 2.3 hold. For all programs P, monitoring execution states (σ_1, ρ_1), (σ'_1, ρ'_1), (σ_2, ρ_2) and (σ'_2, ρ'_2) such that:

- $(\sigma_1, \rho_1), \bot \vdash P \Downarrow_{\mathcal{M}} (\sigma'_1, \rho'_1)$,
- $(\sigma_2, \rho_2), \bot \vdash P \Downarrow_{\mathcal{M}} (\sigma'_2, \rho'_2)$,
- $\forall x \notin \mathcal{S}(P).\ \sigma_1(x) = \sigma_2(x)$,
- $\forall x \in \mathcal{S}(P).\ \rho_1(x) = \top$

the following holds:

$$\forall x. \quad (\rho_1'(x) = \bot) \;\Rightarrow\; (\sigma_1'(x) = \sigma_2'(x))$$

Proof. Follows directly from Lemma A.7.

Variables whose final tags are not $\bot$ may have different final values for two initially low equivalent executions. Therefore, the monitoring mechanism must modify the final value of any variable belonging to $O(\mathrm{P})$ and whose final tag is not $\bot$. However, as emphasized by Le Guernic and Jensen [2005], if the correction of "bad" information flows is done without enough care, the correction mechanism itself can become a new covert channel carrying secret information. Theorem 4.2 proves that any variable belonging to $O(\mathrm{P})$ and whose final tag is not $\bot$ can safely be reset to a default value because the final tag of a variable does not depend on the secret inputs of the program.

Theorem 4.2 (Sound Correction).
Assume that the monitoring semantics $\Downarrow_{\mathcal{M}}$ uses a static information flow analysis ($[\![]\!]^{\sharp_{\mathcal{G}}}$) for which Hypotheses 2.1, 2.2 and 2.3 hold. For all programs P, monitoring execution states $(\sigma_1,\ \rho_1)$, $(\sigma_1',\ \rho_1')$, $(\sigma_2,\ \rho_2)$ and $(\sigma_2',\ \rho_2')$ such that:

- $(\sigma_1,\ \rho_1),\ \bot \ \vdash\ P \ \Downarrow_{\mathcal{M}}\ (\sigma_1',\ \rho_1')$,
- $(\sigma_2,\ \rho_2),\ \bot \ \vdash\ P \ \Downarrow_{\mathcal{M}}\ (\sigma_2',\ \rho_2')$,
- $\forall x \notin \mathcal{S}(P).\ \sigma_1(x) = \sigma_2(x)$,
- $\forall x \in \mathcal{S}(P).\ \rho_1(x) = \top$
- $\rho_1 = \rho_2$

the following holds: $\rho_1' = \rho_2'$.

Proof. Follows directly from Lemma A.9.

4.2 Precision

The remainder of this section compares the precision of the dynamic information flow analysis proposed in this report with the precision of the monitoring mechanism proposed in previous work [Le Guernic et al., 2006a]. Theorem 4.2 proves that a correction mechanism can be based on the tags computed by the dynamic analysis without creating new covert channels. As neither of the dynamic analyses of previous work [Le Guernic et al., 2006a] and work presented in this chapter modifies the internal state of the program (the value store), the correction mechanism of previous work [Le Guernic et al., 2006a] can be used in combination with the dynamic analysis proposed in this report. However, as this report does not formally specifies a correction mechanism, only the precision of the dynamic information flow analyses of the dynamic information flow analysis presented in this report is compared to previous work [Le Guernic et al., 2006a].

Theorem 4.3 states that, for any program P without output statements, if an execution monitored by the mechanism of previous work [Le Guernic et al., 2006a] terminates concluding that variables not belonging to V' do not contain secret information, then an execution started with the same inputs and monitored by the dynamic analysis of this report does also terminates and concludes that variables not belonging to V' do not contain secret information.

Theorem 4.3 (Improved Precision).
Assume that the monitoring semantics $\Downarrow_{\mathcal{M}}$ uses a static information flow analysis ($[\![]\!]^{\sharp_{\mathcal{G}}}$) which is acceptable *and for which Hypotheses 2.3 and 2.4 hold. For all programs P, whose set of secret inputs is $\mathcal{S}(P)$, value stores σ and σ', monitoring automaton states (V', w'), and tag stores ρ and ρ' such that:*

- $((\mathcal{S}(P), \epsilon), \sigma) \Vdash P \overset{\epsilon}{\Rightarrow} ((V', w'), \sigma')$,
- $\forall x \notin \mathcal{S}(P),\ \rho(x) = \bot$,

the following holds:

- $(\sigma,\ \rho),\ \bot \vdash P \Downarrow_{\mathcal{M}} (\sigma',\ \rho')$,
- $\forall x \notin V',\ \rho(x) = \bot$.

Proof. Follows directly from Lemma A.18.

5 Conclusion

This report, as the previous ones, addresses the problem of monitoring executions in order to enforce the confidentiality of secret data. The confidentiality property to monitor is expressed using the property of noninterference between secret inputs of the execution and its public outputs. The language taken into consideration is a sequential language with assignments and conditionals (including loops) as in previous work [Le Guernic et al., 2006a]. The main difference between the monitoring mechanism proposed in this report and the ones of previous works lies in the static analysis used to detect implicit indirect flows. The static information flow analyses used by the monitor in this report are context-sensitive, which is not the case in previous works. This allows to increase the precision of the overall dynamic information flow analysis for the detection of implicit and explicit indirect flows.

Another distinguishing feature of the dynamic information flow analysis developed in this report lies in the definition of the static analysis used for the detection of implicit indirect flows. No static information flow analysis is formally defined to be used by the dynamic analysis, even if such an analysis can easily be extracted from the acceptability rules of Figure 5 by fix point computation. Instead, three hypotheses are defined. It has been proved that any static information flow analysis respecting those hypotheses can be used by the noninterference monitor proposed in this report. The first hypothesis (Hypothesis 2.1) simply requires the static analysis to be sound with regard to the detection of the variables whose value may be modified by the execution of a given piece of code in a given context. The second hypothesis (Hypothesis 2.2) requires the static analysis to have a precision similar to the overall dynamic analysis for pieces of code executed in a secret context. Finally, Hypothesis 2.3 requires the static analysis to be deterministic with regard to public data given in the analysis context. Hypotheses 2.2 and 2.3 are not required for the overall dynamic analysis to be sound with regard to the detection of information flows. However, they are required in order to prevent the "bad flows" correction mechanism from creating new covert channels which may leak secret information.

In Section 4, the proposed noninterference monitor is proved to be sound both with regard to the detection of information flows and with regard to their correction when necessary. In the same section, Theorem 4.3 states that the monitor proposed in this

report is more precise than the monitor of previous work [Le Guernic et al., 2006a] and, by transitivity, more precise than the type system developed by Volpano et al. [1996].

A Proofs of the Theorems

Some of the proofs in this chapter use a semantics rule for the evaluation of expressions. This semantics rule is defined below.

Definition A.1 (Semantics rule for expression evaluation). *For all value store σ, tag store ρ, and expression e:*

$$\sigma; \rho \vdash e \Downarrow \sigma(e) : \rho(e)$$

A.1 Usable Analyses

A.1.1 Hypothesis 2.1

Lemma A.1 (Constraints in Figure 5 enforce the Hypothesis 2.1).
For all value stores σ, σ_i and σ_o, tag stores ρ, ρ_i and ρ_o, program counter tag t^{pc}, statement S, and analysis result $(\mathfrak{D}, \mathfrak{X})$ such that:

$\star_1$ $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho \vdash S)$,

$\star_2$ $\forall x : (\rho(x) = \bot) \Rightarrow \sigma_i(x) = \sigma(x)$,

$\star_3$ $(\sigma_i,\ \rho_i),\ t^{pc} \vdash S \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$

it is true that:

- $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$

Proof. The proof goes by induction on the derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$ ". Assume the lemma holds for any sub-derivation tree, if the last rule used is:

(**E − SKIP**) then we can conclude that :

(1) S is "**skip**" and $\sigma o = \sigma_i$.
It follows directly from the definition of the rule (**E − SKIP**).

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusion (1).

(**E − ASSIGN**) then we can conclude that :

(1) S is "id **:=** e" and $\forall x \neq id. \sigma o(x) = \sigma_i(x)$.
It follows directly from the definition of the rule (**E − ASSIGN**).

(2) $id \in \mathfrak{X}$.
If follows from the global hypothesis $\star_1$ and the local conclusion (1).

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusions (1) and (2).

(**E − SEQUENCE**) then we can conclude that :

(1) S is "S_1 ; S_2" and there exist a value store σ_1, a tag store ρ_1, and two identifiers sets X_1 and X_2 such that:

- $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} (\sigma_1,\ \rho_1)$ is a sub-derivation tree of "$S_1\ ;\ S_2$"
- $(\sigma_1,\ \rho_1),\ t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$ is a sub-derivation tree of "$S_1\ ;\ S_2$"

It follows directly from the definition of the rule (**E − SEQUENCE**).

(2) There exists two analysis results $(\mathfrak{D}_1, \mathfrak{X}_1)$ and $(\mathfrak{D}_2, \mathfrak{X}_2)$ such that:

- $(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_1)$
- $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho' \vdash S_2)$ with $\rho' = \rho \sqcup ((\mathfrak{X}_1 \times \top) \cup ((\mathfrak{X}_1)^c \times \bot))$
- $\mathfrak{X}_1 \cup \mathfrak{X}_2 \subseteq \mathfrak{X}$

It follows from the global hypothesis $\star_1$, the local conclusion (1) and the definition of the relation $\models$.

(3) $\forall x \notin \mathfrak{X}_1.\ \sigma_1(x) = \sigma_i(x)$.
This result is obtained by applying the inductive hypothesis on the derivation "$(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} (\sigma_1,\ \rho_1)$" (sub-derivation tree of "$(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$") using the fact that "$(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_1)$" (from the local conclusion (2)).

(4) $\forall x : (\rho'(x) = \bot) \Rightarrow \sigma_1(x) = \sigma(x)$.
For all variable x, from the definition of ρ' in the local conclusion (2) and the local conclusion (3), we can conclude that "$\rho'(x) = \bot$" implies "$\rho(x) = \bot$" and "$\sigma_1(x) = \sigma_i(x)$". Using the global hypothesis $\star_2$ and the fact that "$\rho(x) = \bot$", it is possible to show that "$\sigma_i(x) = \sigma(x)$". Hence, we can conclude that "$\rho'(x) = \bot$" implies "$\sigma_1(x) = \sigma(x)$".

(5) $\forall x \notin \mathfrak{X}_2.\ \sigma_o(x) = \sigma_1(x)$.
This result is obtained by applying the inductive hypothesis on the derivation "$(\sigma_1,\ \rho_1),\ t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$" (sub-derivation tree of "$(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$") using the fact that "$(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho' \vdash S_2)$" (from the local conclusion (2)). The inductive hypothesis can be applied thanks to the local conclusion (4).

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusions (2), (3) and (5).

(E − IF$_\bot$) then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exists a value $v \in \{true, false\}$ such that:

- $\sigma_i(e) = v$ and $\rho_i(e) = \bot$,
- " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S_v \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$ " is a sub-derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$ "

It follows directly from the definition of the rule (**E − IF$_\bot$**).

(2) There exist $\mathfrak{X}_{true}$ and $\mathfrak{X}_{false}$ such that:

- $(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$,
- $\mathfrak{X} \supseteq [\![\mathfrak{X}_{true}, \mathfrak{X}_{false}]\!]^{\rho(e)}_{\sigma(e)}$

It follows from the global hypothesis $\star_1$, the local conclusions (1), and the definition of $\models$.

(3) $\forall x \notin \mathfrak{X}_v.\ \sigma_o(x) = \sigma_i(x)$.
This result is obtained by applying the inductive hypothesis on "$(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$" and "$(\sigma_i,\ \rho_i),\ t^{\text{pc}} \vdash S_v \Downarrow_{\mathcal{M}} (\sigma_o,\ \rho_o)$".

(4) $\mathfrak{X}_v \subseteq \mathfrak{X}$.
It follows directly from the local conclusions (2) and (1).

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusions (3) and (4).

$(\mathbf{E-IF}_{\top})$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exist a value $v \in \{true, false\}$ and a tag store ρ' such that:

- $\sigma_i(e) = v$ and $\rho_i(e) = \top$,
- " $(\sigma_i,\ \rho_i),\ \top\ \vdash\ S_v\ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho')$ " is a sub-derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}}\ \vdash\ S\ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_o)$ "

It follows directly from the definition of the rule $(\mathbf{E-IF}_{\top})$.

(2) There exist $\mathfrak{X}_{true}$ and $\mathfrak{X}_{false}$ such that:

- $(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$,
- $\mathfrak{X} \supseteq \ \{\!\!\{\mathfrak{X}_{true}, \mathfrak{X}_{false}\}\!\!\}^{\rho(e)}_{\sigma(e)}$

It follows from the global hypothesis $\star_1$, the local conclusions (1), and the definition of $\models$.

(3) $\forall x \notin \mathfrak{X}_v.\ \sigma_o(x) = \sigma_i(x)$.
This result is obtained by applying the inductive hypothesis on "$(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$" and "$(\sigma_i,\ \rho_i),\ \top\ \vdash\ S_v\ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho')$".

(4) $\mathfrak{X}_v \subseteq \mathfrak{X}$.
It follows directly from the local conclusions (2) and (1).

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusions (3) and (4).

$(\mathbf{E-WHILE}_{\mathbf{skip}})$ then we can conclude that :

(1) $\sigma_o = \sigma_i$.
It follows directly from the definition of the rule $(\mathbf{E-WHILE}_{\mathbf{skip}})$.

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusion (1).

$(\mathbf{E-WHILE}_{\mathbf{true}_{\perp}})$ then we can conclude that :

(1) S is "**while** e **do** S_l **done**" and there exist a tag t_e and a tag store ρ_t:

- $\sigma_i(e) = \texttt{true}$ and $\rho_i(e) = t_e$
- " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \sqcup t_e\ \vdash\ S_l$; **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_t)$ " is a sub-derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}}\ \vdash\ S\ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_o)$ "

It follows directly from the definition of the rule $(\mathbf{E-WHILE}_{\mathbf{true}})$.

(2) There exist an analysis result $(\mathfrak{D}_l, \mathfrak{X}_l)$ such that:

- $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ with $\rho' = \rho\ \sqcup\ ((\mathfrak{X}_l \times \top)\ \cup\ ((\mathfrak{X}_l)^c \times \perp))$,
- $\mathfrak{X} = \mathfrak{X}_l$,

It follows from the global hypotheses $\star_1$ and $\star_2$, the local conclusions (1) (S is "**while** e **do** S_l **done**" and $\sigma_i(e) = \texttt{true}$), and the definition of $\models$.

(3) There exist a value store σ_1 and a tag store ρ_1 such that:

- " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \sqcup t_e \ \vdash\ S_l \ \Downarrow_{\mathcal{M}}\ (\sigma_1,\ \rho_1)$ " is a sub-derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \sqcup t_e \ \vdash\ S \ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_o)$ ",
- " $(\sigma_1,\ \rho_1),\ t^{\text{pc}} \sqcup t_e \ \vdash\ \textbf{while}\ e\ \textbf{do}\ S_l\ \textbf{done} \ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_t)$ " is a sub-derivation tree of " $(\sigma_i,\ \rho_i),\ t^{\text{pc}} \sqcup t_e \ \vdash\ S \ \Downarrow_{\mathcal{M}}\ (\sigma_o,\ \rho_o)$ ",

It follows directly from the local conclusion (1) and the definition of the rule (**E − SEQUENCE**).

(4) $\forall x : (\rho'(x) = \bot) \Rightarrow (\sigma_i(x) = \sigma(x))$.
From the definition of ρ' (given in the local conclusion (2)), $\rho'(x) = \bot$ implies $\rho(x) = \bot$. Hence, the global hypothesis $\star_2$ implies the above result.

(5) $\forall x \notin \mathfrak{X}_l.\ \sigma_1(x) = \sigma_i(x)$.
This result follows from the inductive hypothesis applied on "$(\sigma_i,\ \rho_i),\ t^{\text{pc}} \sqcup t_e \ \vdash\ S_l \ \Downarrow_{\mathcal{M}}\ (\sigma_1,\ \rho_1)$" (given in the local conclusion (3)) and $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ (given in the local conclusion (2)). The inductive hypothesis can be applied because of the local conclusion (4).

Lets define ρ_l and ρ'_l such that:

$\diamond_1\ \forall x \in \mathfrak{X}_l, \rho_l(x) = \top$,

$\diamond_2\ \forall x \notin \mathfrak{X}_l, \rho_l(x) = \rho(x)$,

$\diamond_3\ \rho'_l = \rho_l \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$.

Lets define $\mathfrak{D}_t$ to be $[\![\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d), \mathcal{I}d]\!]^{\rho_l(e)}_{\sigma(e)}$.

(6) $\rho'_l = \rho'$.
It follows from the definitions of ρ'_l and ρ' (given in $\diamond_3$ and (2)). For all x in $\mathfrak{X}_l$, $\rho'_l(x) = \top = \rho'(x)$. For all x not in $\mathfrak{X}_l$, from $\diamond_2$, $\rho_l(x) = \rho(x)$; hence, $\rho'_l(x) = \rho_l(x) = \rho(x) = \rho'(x)$.

(7) $(\mathfrak{D}_t, \mathfrak{X}_l) \models (\sigma, \rho_l \vdash \textbf{while}\ e\ \textbf{do}\ S_l\ \textbf{done})$.
From the local conclusions (2) and (6), we can conclude that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho'_l \vdash S_l)$. From the local definition $\diamond_3$, $\rho'_l = \rho_l \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$. The local definitions $\diamond_1$ and $\diamond_2$ and the global hypothesis $\star_2$ imply $\rho_l(e) \neq \top \Rightarrow \rho(e) \neq \top \Rightarrow \sigma(e) = \sigma_i(e)$; therefore, as $\sigma_i(e) = \texttt{true}$, $\mathfrak{X}_l = [\![\mathfrak{X}_l, \emptyset]\!]^{\rho(e)}_{\sigma(e)}$. Hence, from the definition of $\models$ for while-statements, $(\mathfrak{D}_t, \mathfrak{X}_l) \models (\sigma, \rho_l \vdash \textbf{while}\ e\ \textbf{do}\ S_l\ \textbf{done})$.

(8) $\forall x : (\rho(x) = \bot) \Rightarrow (\sigma_1(x) = \sigma(x))$.
From $\diamond_1$, $\rho_l(x) = \bot$ implies $x \notin \mathfrak{X}_l$. Hence, local conclusion (5) implies $\sigma_1(x) = \sigma_i(x)$. From the property "$\rho_l(x) = \bot$ implies $x \notin \mathfrak{X}_l$", the local definition $\diamond_2$, and the global hypothesis $\star_2$, it is possible to show that $\rho_l(x) = \bot$ implies $\sigma_i(x) = \sigma(x)$. Hence, $\rho_l(x) = \bot$ implies $\sigma_1(x) = \sigma(x)$.

(9) $\forall x \notin \mathfrak{X}_l.\ \sigma_o(x) = \sigma_1(x)$.
The inductive hypothesis, using the local conclusions (7), (8), and (3), implies the desired result.

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
From the local conclusions (9) and (5), $\forall x \notin \mathfrak{X}_l.\ \sigma_o(x) = \sigma_i(x)$. The fact $\mathfrak{X} = \mathfrak{X}_l$ of the local conclusion (2) completes the proof.

(**E − WHILE$_{\textbf{true}_\top}$**) then we can conclude that :

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
The proof goes exactly the same way as for ($\mathbf{E - WHILE_{true_\top}}$).

($\mathbf{E - WHILE_{false_\top}}$) then we can conclude that :

(•) $\forall x \notin \mathfrak{X}.\ \sigma_o(x) = \sigma_i(x)$.
The proof goes exactly the same way as for ($\mathbf{E - WHILE_{skip}}$).

A.1.2 Hypothesis 2.2

Lemma A.2 (Analysis is tag store monotone for assigned variables).
For all value store σ, tag stores ρ_1 and ρ_2, statement S, and analysis result $(\mathfrak{D}_1, \mathfrak{X}_1)$ such that:

$\star_1$ $(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho_1 \vdash S)$,

$\star_2$ $\rho_2 \sqsubseteq \rho_1$

then there exists an analysis result $(\mathfrak{D}_2, \mathfrak{X}_2)$ such that:

- $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho_2 \vdash S)$ *and*
- $\mathfrak{X}_2 \subseteq \mathfrak{X}_1$

Proof. The proof, which goes by structural induction on the statement S, is direct. It mainly relies on the following property. For all tag stores ρ_1, ρ_1', ρ_2 and ρ_2' and variable sets $\mathfrak{X}_1$ and $\mathfrak{X}_2$ such that $\rho_1' = \rho_1 \sqcup ((\mathfrak{X}_1 \times \top) \cup ((\mathfrak{X}_1)^c \times \bot))$ and $\rho_2' = \rho_2 \sqcup ((\mathfrak{X}_2 \times \top) \cup ((\mathfrak{X}_2)^c \times \bot))$, if $\rho_2 \sqsubseteq \rho_1$ and $\mathfrak{X}_2 \subseteq \mathfrak{X}_1$ then $\rho_2' \sqsubseteq \rho_1'$.

Lemma A.3 (Constraints are high-values independent).
For all value stores σ and σ', tag store ρ, statement S, and analysis result $(\mathfrak{D}, \mathfrak{X})$ such that:

$\star_1$ $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho \vdash S)$,

$\star_2$ $\forall x : (\rho(x) = \bot) \Rightarrow (\sigma'(x) = \sigma(x))$

it is true that:

- $(\mathfrak{D}, \mathfrak{X}) \models (\sigma', \rho \vdash S)$

Proof. The proof, which goes by structural induction on the statement S, is direct. It mainly relies on the following property. For all value stores σ and σ' and tag stores ρ, such that $\forall x : (\rho(x) = \bot) \Rightarrow (\sigma'(x) = \sigma(x))$, the following holds: $[\![X, Y]\!]^{\rho(e)}_{\sigma(e)} = [\![X, Y]\!]^{\rho(e)}_{\sigma'(e)}$.

Lemma A.4 (Constraints in Figure 5 enforce the Hypothesis 2.2).
For all value store σ, tag store ρ, statement S, analysis result $(\mathfrak{D}, \mathfrak{X})$, and variable x such that:

$\star_1$ $\sigma; \rho; \top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$,

$\star_2$ $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho \vdash S)$,

it is true that:

$$\rho_o(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) \quad \sqcup \quad ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x)$$

Proof. The proof goes by induction on the derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ". Assume the lemma holds for any sub-derivation tree, if the last rule used is:

(**E − SKIP**) then we can conclude that :

(1) S is "**skip**" and $\rho_o = \rho$.
It follows directly from the definition of the rule (**E − SKIP**).

(2) $\mathfrak{D} = \mathcal{I}d$ and $\mathfrak{X} = \emptyset$.
It follows directly from the global hypothesis $\star_2$ and the definition of $\models$.

(•) $\rho_o = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
It follows from the local conclusions (1) and (2).

(**E − ASSIGN**) then we can conclude that :

(1) S is "id **:=** e" and $\rho_o = \rho[id \mapsto \top]$.
It follows directly from the definition of the rule (**E − ASSIGN**).

(2) $\mathfrak{D} = \mathcal{I}d[id \mapsto \mathcal{V}(e)]$ and $\mathfrak{X} = \{id\}$.
It follows directly from the global hypothesis $\star_2$ and the definition of $\models$.

(•) $\rho_o = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
It follows from the local conclusions (1) and (2).

(**E − SEQUENCE**) then we can conclude that :

(1) S is "S_1 ; S_2" and there exist a value store σ_1 and a tag store ρ_1 such that:

- $\sigma;\rho;\top \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ is a sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $\sigma_1;\rho_1;\top \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ is a sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule (**E − SEQUENCE**).

(2) There exist $(\mathfrak{D}_1, \mathfrak{X}_1)$ and $(\mathfrak{D}_2, \mathfrak{X}_2)$ such that:

- $(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_1)$
- $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho' \vdash S_2)$ with $\rho' = \rho \sqcup ((\mathfrak{X}_1 \times \top) \cup ((\mathfrak{X}_1)^c \times \bot))$
- $\mathfrak{D} = \mathfrak{D}_1 \circ \mathfrak{D}_2$
- $\mathfrak{X} = \mathfrak{X}_1 \cup \mathfrak{X}_2$

It follows from the global hypothesis $\star_2$, the local conclusion (1) and the definition of the relation $\models$.

(3) $\rho_1 = (\lambda x. \bigsqcup_{y \in \mathfrak{D}_1(x)} \rho(y)) \sqcup ((\mathfrak{X}_1 \times \{\top\}) \cup (\mathfrak{X}_1^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on the derivation " $\sigma;\rho;\top \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " (sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") using the fact that "$(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_1)$" (from the local conclusion (2)).

(4) $\rho' = \rho_1$.
For all x in $\mathfrak{X}_1$, $\rho'(x) = \top = \rho_1(x)$ (it follows from the definitions of ρ' and ρ_1 in the local conclusions (2) and (3)). From the local conclusion (2) and hypothesis 2.4, if a variable x is not in $\mathfrak{X}_1$ then $\mathfrak{D}_1(x) = \{x\}$. Hence, from the local conclusion (3), if a variable x is not in $\mathfrak{X}_1$ then $\rho_1(x) = \rho(x)$. For all x not in $\mathfrak{X}_1$, $\rho'(x) = \rho(x)$ (it follows from the definition of ρ' in the local conclusion (2)).

(5) $\forall x : (\rho'(x) = \bot) \Rightarrow (\sigma_1(x) = \sigma(x))$.
From the definition of ρ' in the local conclusion (2), for all variable x if $\rho'(x) = \bot$ then x is not in $\mathfrak{X}_1$. Hence, from the local conclusions (1) and (2) and lemma A.1, for all variable x $\rho'(x) = \bot$ implies $\sigma_1(x) = \sigma(x)$.

(6) $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho' \vdash S_2) \Rightarrow (\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma_1, \rho_1 \vdash S_2)$.
It follows directly from the local conclusions (4) and (5) and lemma A.3.

(7) $\rho_o = (\lambda x. \bigsqcup_{y \in \mathfrak{D}_2(x)} \rho_1(y)) \sqcup ((\mathfrak{X}_2 \times \{\top\}) \cup (\mathfrak{X}_2^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on the derivation " $\sigma_1; \rho_1; \top \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma; \rho; \top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") using the fact that "$(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma_1, \rho_1 \vdash S_2)$" (from the local conclusion (2)).

(8) $((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\})) = ((\mathfrak{X}_1 \times \{\top\}) \cup (\mathfrak{X}_1^c \times \{\bot\})) \sqcup ((\mathfrak{X}_2 \times \{\top\}) \cup (\mathfrak{X}_2^c \times \{\bot\}))$.
It follows from the local conclusion (2) and the fact that $\bot \sqsubseteq \top$.

(9) $\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) = \lambda x. \bigsqcup_{z \in \mathfrak{D}_2(x)} (\bigsqcup_{y \in \mathfrak{D}_1(z)} \rho(y))$.
It follows from the local conclusion (2).

(•) $(\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\})) = \rho_o$.
For all x, let $t(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x)$, then:

$$
\begin{aligned}
t(x) = & \bigsqcup_{z \in \mathfrak{D}_2(x)} \Big(\bigsqcup_{y \in \mathfrak{D}_1(z)} \rho(y) \Big) \\
& \sqcup \quad ((\mathfrak{X}_1 \times \{\top\}) \cup (\mathfrak{X}_1^c \times \{\bot\}))(x) \sqcup ((\mathfrak{X}_2 \times \{\top\}) \cup (\mathfrak{X}_2^c \times \{\bot\}))(x) \\
= & \bigsqcup_{z \in \mathfrak{D}_2(x)} \Big(\quad \Big(\bigsqcup_{y \in \mathfrak{D}_1(z)} \rho(y) \Big) \quad \sqcup \quad ((\mathfrak{X}_1 \times \{\top\}) \cup (\mathfrak{X}_1^c \times \{\bot\}))(x) \quad \Big) \\
& \sqcup \quad ((\mathfrak{X}_2 \times \{\top\}) \cup (\mathfrak{X}_2^c \times \{\bot\}))(x) \\
= & \bigsqcup_{z \in \mathfrak{D}_2(x)} \Big(\quad \rho_1(z) \quad \Big) \quad \sqcup \quad ((\mathfrak{X}_2 \times \{\top\}) \cup (\mathfrak{X}_2^c \times \{\bot\}))(x) \\
= & \ \rho_o(x)
\end{aligned}
\tag{1}
$$

The equation (1) follows from the local conclusions (9), (8), (3), and (7).

$(\mathbf{E} - \mathbf{IF}_\bot)$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exist a value $v \in \{true, false\}$ such that:

- " $\sigma; \rho \vdash e \Downarrow_{\mathcal{M}} v : \bot$ ",
- " $\sigma; \rho; t^{\text{pc}} \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma; \rho; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule $(\mathbf{E} - \mathbf{IF}_\bot)$.

(2) There exist $(\mathfrak{D}_{true}, \mathfrak{X}_{true})$ and $(\mathfrak{D}_{false}, \mathfrak{X}_{false})$ such that:

- $(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$,
- $\mathfrak{D} = [\![\mathfrak{D}_{true}, \mathfrak{D}_{false}]\!]_{\sigma(e)}^{\rho(e)} \cup ([\![\mathfrak{X}_{true}, \mathfrak{X}_{false}]\!]_{\sigma(e)}^{\rho(e)} \times \mathcal{FV}(e))$,
- $\mathfrak{X} = [\![\mathfrak{X}_{true}, \mathfrak{X}_{false}]\!]_{\sigma(e)}^{\rho(e)}$

It follows from the global hypothesis $\star_2$, the local conclusions (1), and the definition of $\models$.

(3) $\rho_o = (\lambda x.\bigsqcup_{y\in\mathfrak{D}_v(x)}\rho(y)) \sqcup ((\mathfrak{X}_v \times \{\top\}) \cup (\mathfrak{X}_v^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on "$(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$" and "$\sigma; \rho; t^{\text{pc}} \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$".

(4) It is true that:

- $\mathfrak{D} = \mathfrak{D}_v \cup (\mathfrak{X}_v \times \mathcal{FV}(e))$,
- $\mathfrak{X} = \mathfrak{X}_v$

It follows from the local conclusion (2) and the facts that $\rho(e) = \bot$ and $\sigma(e) = v$ (from the local conclusion (1)).

(5) $\forall y \in \mathcal{FV}(e), \rho(y) = \bot$.
It follows from the fact that $\rho(e) = \bot$ (from the local conclusion (1)) and the semantics rules for expressions.

(•) $\rho_o = (\lambda x.\bigsqcup_{y\in\mathfrak{D}(x)}\rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
It follows from the local conclusions (3), (4), and (5)

$(\mathbf{E} - \mathbf{IF}_\top)$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exist a value $v \in \{true, false\}$ such that:

- " $\sigma; \rho \vdash e \Downarrow_{\mathcal{M}} v : \top$ ",
- " $\sigma; \rho; \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " is a sub-derivation tree of " $\sigma; \rho; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ",
- $(\mathfrak{D}_{\neg v}, \mathfrak{X}_{\neg v}) \models (\sigma, \rho \vdash S_{\neg v})$,
- $\rho_o = \rho_v \sqcup (\lambda x.\bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)}\rho(y)) \sqcup ((\mathfrak{X}_{\neg v} \times \{\top\}) \cup (\mathfrak{X}_{\neg v}^c \times \{\bot\}))$,

It follows directly from the definition of the rule $(\mathbf{E} - \mathbf{IF}_\top)$.

(2) There exist $(\mathfrak{D}_{true}, \mathfrak{X}_{true})$ and $(\mathfrak{D}_{false}, \mathfrak{X}_{false})$ such that:

- $(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$,
- $(\mathfrak{D}_{\neg v}, \mathfrak{X}_{\neg v}) \models (\sigma, \rho \vdash S_{\neg v})$,
- $\mathfrak{D} = (\mathfrak{D}_v \cup \mathfrak{D}_{\neg v}) \cup ((\mathfrak{X}_v \cup \mathfrak{X}_{\neg v}) \times \mathcal{FV}(e))$,
- $\mathfrak{X} = \mathfrak{X}_v \cup \mathfrak{X}_{\neg v}$

It follows from the global hypothesis $\star_2$, the local conclusions (1), the definition of $\models$, and the fact that $\rho(e) = \top$ (from the local conclusion (1)).

(3) $\rho_v = (\lambda x.\bigsqcup_{y\in\mathfrak{D}_v(x)}\rho(y)) \sqcup ((\mathfrak{X}_v \times \{\top\}) \cup (\mathfrak{X}_v^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on the derivation " $\sigma; \rho; \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " (sub-derivation tree of " $\sigma; \rho; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") using the fact that "$(\mathfrak{D}_v, \mathfrak{X}_v) \models (\sigma, \rho \vdash S_v)$" (from the local conclusion (2)).

(4) $((\mathfrak{X}\times\{\top\})\cup(\mathfrak{X}^c\times\{\bot\})) = ((\mathfrak{X}_v\times\{\top\})\cup(\mathfrak{X}_v^c\times\{\bot\}))\sqcup((\mathfrak{X}_{\neg v}\times\{\top\})\cup(\mathfrak{X}_{\neg v}^c\times\{\bot\}))$.
It follows from the local conclusion (2) and the fact that $\bot \sqsubseteq \top$.

(5) $\exists y \in \mathcal{FV}(e) : \rho(y) = \top$.
It follows directly from the fact that $\rho(e) = \top$ (given in the local conclusion (1)) and the semantics for expressions.

(6) $\forall x, \quad \bigsqcup_{y\in(\mathfrak{X}\times\mathcal{FV}(e))(x)}\rho(y) = ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x)$.
It follows from the local conclusion (2) and (5).

(•) $(\lambda x. \bigsqcup_{y\in\mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\})) = \rho_o$.
For all x, let $t(x) = \bigsqcup_{y\in\mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x)$, then:

$$
\begin{aligned}
t(x) &= \bigsqcup_{y\in(\mathfrak{D}_v \cup \mathfrak{D}_{\neg v} \cup (\mathfrak{X}\times\mathcal{FV}(e)))(x)} \rho(y) \quad \sqcup \quad ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) \\
&= \bigsqcup_{y\in\mathfrak{D}_v(x)} \rho(y) \quad \sqcup \quad \bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)} \rho(y) \quad \sqcup \quad \bigsqcup_{y\in(\mathfrak{X}\times\mathcal{FV}(e))(x)} \rho(y) \\
&\quad \sqcup \quad ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) \\
&= \bigsqcup_{y\in\mathfrak{D}_v(x)} \rho(y) \quad \sqcup \quad \bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)} \rho(y) \\
&\quad \sqcup \quad ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) \quad \sqcup \quad ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) \\
&= \bigsqcup_{y\in\mathfrak{D}_v(x)} \rho(y) \quad \sqcup \quad \bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)} \rho(y) \\
&\quad \sqcup \quad ((\mathfrak{X}_v \times \{\top\}) \cup (\mathfrak{X}_v^c \times \{\bot\}))(x) \quad \sqcup \quad ((\mathfrak{X}_{\neg v} \times \{\top\}) \cup (\mathfrak{X}_{\neg v}^c \times \{\bot\}))(x) \\
&= \bigsqcup_{y\in\mathfrak{D}_v(x)} \rho(y) \quad \sqcup \quad ((\mathfrak{X}_v \times \{\top\}) \cup (\mathfrak{X}_v^c \times \{\bot\}))(x) \\
&\quad \sqcup \quad \bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)} \rho(y) \quad \sqcup \quad ((\mathfrak{X}_{\neg v} \times \{\top\}) \cup (\mathfrak{X}_{\neg v}^c \times \{\bot\}))(x) \\
&= \rho_v(x) \quad \sqcup \quad \bigsqcup_{y\in\mathfrak{D}_{\neg v}(x)} \rho(y) \quad \sqcup \quad ((\mathfrak{X}_{\neg v} \times \{\top\}) \cup (\mathfrak{X}_{\neg v}^c \times \{\bot\}))(x) \\
&= \rho_o(x)
\end{aligned}
\tag{2}
$$

The equation (2) follows from the local conclusions (2), (6), (4), (3), and (1).

(**E − WHILE$_{\text{skip}}$**) then we can conclude that :

(1) S is "e ; S_l" and:

- " $\sigma;\rho \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \bot$ ",
- $\rho_o = \rho$

It follows directly from the definition of the rule (**E − WHILE$_{\text{skip}}$**).

(2) It is true that:

- $\mathfrak{D} = \mathcal{I}d$,
- $\mathfrak{X} = \emptyset$

It follows from the global hypothesis $\star_2$, the local conclusion (1), the definition of $\models$, and the facts that $\rho(e) = \bot$ and $\sigma(e) = \texttt{false}$ (from the local conclusion (1)).

(•) $\rho_o = (\lambda x. \bigsqcup_{y\in\mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
It follows directly from the local conclusions (1) and (2).

(**E − WHILE$_{\text{true}_\bot}$**) then we can conclude that :

(1) S is "**while** e **do** S_l **done**" and:

- " $\sigma;\rho \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \bot$ "

- " $\sigma;\rho;\top \vdash S_l$; **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E-WHILE_{true_\perp}}$).

(2) There exist $(\mathfrak{D}_l, \mathfrak{X}_l)$ such that:

- $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ with $\rho' = \rho \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$,
- $\mathfrak{D} = \mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)$,
- $\mathfrak{X} = \mathfrak{X}_l$

It follows from the global hypothesis $\star_2$, the local conclusion (1), the definition of $\models$, and the facts that $\rho(e) = \bot$ and $\sigma(e) = \texttt{true}$ (from the local conclusion (1)).

(3) There exist a value store σ_1 and a tag store ρ_1:

- " $\sigma;\rho;\top \vdash S_l \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " is a sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ",
- " $\sigma_1;\rho_1;\top \vdash$ **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma;\rho;\top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the local conclusion (1) and the definition of the rule ($\mathbf{E-SEQUENCE}$).

(4) There exists an analysis result $(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l})$ such that:

- $(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l}) \models (\sigma, \rho \vdash S_l)$,
- $\mathfrak{X}_{S_l} \subseteq \mathfrak{X}_l$

It follows from the lemma A.2, the fact that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ (from the local conclusion (2)), and the fact that $\rho \sqsubseteq \rho'$ (from the definition of ρ' in the local conclusion (2)).

(5) $\rho_1 = (\lambda x. \bigsqcup_{y \in \mathfrak{D}_{S_l}(x)} \rho(y)) \sqcup ((\mathfrak{X}_{S_l} \times \{\top\}) \cup (\mathfrak{X}^c_{S_l} \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on " $\sigma;\rho;\top \vdash S_l \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " (from the local conclusion (3)) and "$(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l}) \models (\sigma, \rho \vdash S_l)$" (from the local conclusion (4)).

(6) $\forall x \in \mathfrak{X}^c_l, \rho(x) = \rho_1(x)$.
From the local conclusion (4), $x \in \mathfrak{X}^c_{S_l}$. Hence, from the hypothesis 2.4, $\mathfrak{D}_{S_l}(x) = \{x\}$. Then, using the local conclusion (5), we can conclude that $\rho_1(x) = \rho(x)$.

Lets define ρ'_1 such that:

$\diamond_1$ $\rho'_1 = \rho_1 \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$.

(7) $\rho'_1 = \rho'$.
It follows from the local conclusion (6) and the definitions of ρ'_1 and ρ' (given in $\diamond_1$ and the local conclusion (2)). For all x in $\mathfrak{X}_l$, $\rho'_1(x) = \top = \rho'(x)$. For all x not in $\mathfrak{X}_l$, the local conclusion (6) implies $\rho'_1(x) = \rho_1(x) = \rho(x) = \rho'(x)$.

(8) $\forall x,\ \rho_1(x) = \bot \Rightarrow \sigma_1(x) = \sigma(x)$.
It follows from lemma A.6 and the local conclusion (3).

(•) $\rho_o = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.

Case 1: $\rho_1(e) = \bot$

(a) $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$.
From the local conclusions (2) and (7), we can conclude that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho_1' \vdash S_l)$ with $\rho_1' = \rho_1 \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$. From the local conclusion (2), $\mathfrak{D} = \mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)$ and $\mathfrak{X} = \mathfrak{X}_l$, from the local conclusion (1), $\sigma(e) = \texttt{true}$, and, from the case hypothesis, $\rho_1(e) = \bot$, then we can conclude that $\mathfrak{D} = [\![\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d), \mathcal{I}d]\!]^{\rho_1(e)}_{\sigma(e)}$ and $\mathfrak{X} = [\![\mathfrak{X}_l, \emptyset]\!]^{\rho_1(e)}_{\sigma(e)}$. Hence, from the definition of $\models$ for while-statements, $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$.

(b) $(\mathfrak{D}, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$.
It follows from hypothesis 2.3 and the local conclusions (8) and (a).

(c) $\rho_o = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on " $\sigma_1; \rho_1; \top \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done} \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (from the local conclusion (3)) and "$(\mathfrak{D}, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$" (from the local conclusion (b)).

Lets define $\mathcal{F}_1^{\rho_o}$ and $\mathcal{F}^{\rho_o}$ such that:

- $\diamond_1$ $\mathcal{F}_1^{\rho_o} = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$,
- $\diamond_2$ $\mathcal{F}^{\rho_o} = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.

(•) $\rho_o(x) = (\bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) = \mathcal{F}^{\rho_o}(x)$.
From the local conclusion (c), $\rho_o(x) = \mathcal{F}_1^{\rho_o}(x)$, and, from the local conclusion (2), $\mathfrak{X} = \mathfrak{X}_l$. If x belongs to $\mathfrak{X}_l$, then $\mathcal{F}_1^{\rho_o}(x) = \top = \mathcal{F}^{\rho_o}(x)$. If x does not belong to $\mathfrak{X}_l$, then x does not belong to $\mathfrak{X}$ (from the local conclusion (2)) and, from the hypothesis 2.4, $\mathfrak{D}(x) = \{x\}$. Hence, if x does not belong to $\mathfrak{X}_l$, $\mathcal{F}_1^{\rho_o}(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho_1(y) = \rho_1(x) = \rho(x)$ (the last equality comes from the local conclusion (6)). For similar reasons, if x does not belong to $\mathfrak{X}_l$ then $\mathcal{F}^{\rho_o}(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) = \rho(x)$.

Case 2: $\rho_1(e) \neq \bot$ (which implies that $\rho_1(e) = \top$)

(a) $\mathfrak{D} \cup \mathcal{I}d = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d)) \cup \mathcal{I}d$.
It follows directly from the fact $\mathfrak{D} = \mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)$ in local conclusion (2).

(b) $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$.
From the local conclusions (2) and (7), we can conclude that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho_1' \vdash S_l)$ with $\rho_1' = \rho_1 \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$. From the local conclusion (a), $\mathfrak{D} \cup \mathcal{I}d = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d)) \cup \mathcal{I}d$, from the local conclusion (2), $\mathfrak{X} = \mathfrak{X}_l$, and from the case hypothesis, $\rho_1(e) = \top$, then we can conclude that $\mathfrak{D} \cup \mathcal{I}d = [\![\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d), \mathcal{I}d]\!]^{\rho_1(e)}_{\sigma(e)}$ and $\mathfrak{X} = [\![\mathfrak{X}_l, \emptyset]\!]^{\rho_1(e)}_{\sigma(e)}$. Hence, from the definition of $\models$ for while-statements, $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$. Finally, the hypothesis 2.3 and the local conclusion (8) imply $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$.

(c) $\rho_o = (\lambda x. \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on " $\sigma_1; \rho_1; \top \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done} \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (from the local conclusion (3)) and "$(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$" (from the local conclusion (b)).

Lets define $\mathcal{F}_1^{\rho_o}$ and $\mathcal{F}^{\rho_o}$ such that:

$\diamond_1$ $\mathcal{F}_1^{\rho_o} = (\lambda x. \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$,

$\diamond_2$ $\mathcal{F}^{\rho_o} = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.

(•) $\rho_o(x) = (\bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) = \mathcal{F}^{\rho_o}(x)$. From the local conclusion (c), $\rho_o(x) = \mathcal{F}_1^{\rho_o}(x)$, and, from the local conclusion (2), $\mathfrak{X} = \mathfrak{X}_l$. If x belongs to $\mathfrak{X}_l$, then $\mathcal{F}_1^{\rho_o}(x) = \top = \mathcal{F}^{\rho_o}(x)$. If x does not belong to $\mathfrak{X}_l$, then x does not belong to $\mathfrak{X}$ (from the local conclusion (2)) and, from the hypothesis 2.4, $\mathfrak{D}(x) = \{x\}$. Hence, if x does not belong to $\mathfrak{X}_l$, $\mathcal{F}_1^{\rho_o}(x) = \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y) = \rho_1(x) = \rho(x)$ (the last equality comes from the local conclusion (6)). For similar reasons, if x does not belong to $\mathfrak{X}_l$ then $\mathcal{F}^{\rho_o}(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) = \rho(x)$.

($\mathbf{E - WHILE}_{\mathbf{true}_\top}$) then we can conclude that :

(1) S is "**while** e **do** S_l **done**" and:

- " $\sigma; \rho \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \top$ ",
- " $\sigma; \rho; \top \vdash S_l$; **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho'_o$ " is a sub-derivation tree of " $\sigma; \rho; \top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ",
- $\rho_o = \rho \sqcup \rho'_o$.

It follows directly from the definition of the rule ($\mathbf{E - WHILE}_{\mathbf{true}_\top}$).

(2) There exist $(\mathfrak{D}_l, \mathfrak{X}_l)$ such that:

- $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ with $\rho' = \rho \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$,
- $\mathfrak{D} = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)) \cup \mathcal{I}d$,
- $\mathfrak{X} = \mathfrak{X}_l$

It follows from the global hypothesis $\star_2$, the local conclusion (1), the definition of $\models$, and the facts that $\rho(e) = \bot$ and $\sigma(e) = \texttt{true}$ (from the local conclusion (1)).

(3) There exist a value store σ_1 and a tag store ρ_1:

- " $\sigma; \rho; \top \vdash S_l \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " is a sub-derivation tree of " $\sigma; \rho; \top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ",
- " $\sigma_1; \rho_1; \top \vdash$ **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho'_o$ " is a sub-derivation tree of " $\sigma; \rho; \top \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the local conclusion (1) and the definition of the rule ($\mathbf{E - SEQUENCE}$).

(4) There exists an analysis result $(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l})$ such that:

- $(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l}) \models (\sigma, \rho \vdash S_l)$,
- $\mathfrak{X}_{S_l} \subseteq \mathfrak{X}_l$

It follows from the lemma A.2, the fact that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ (from the local conclusion (2)), and the fact that $\rho \sqsubseteq \rho'$ (from the definition of ρ' in the local conclusion (2)).

(5) $\forall x \in \mathfrak{X}_l^c, \rho(x) = \rho_1(x)$.
By applying the inductive hypothesis on " $\sigma; \rho; \top \vdash S_l \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " (from the local conclusion (3)) and "$(\mathfrak{D}_{S_l}, \mathfrak{X}_{S_l}) \models (\sigma, \rho \vdash S_l)$" (from the local conclusion (4)), we get that $\rho_1 = (\lambda x. \bigsqcup_{y \in \mathfrak{D}_{S_l}(x)} \rho(y)) \sqcup ((\mathfrak{X}_{S_l} \times \{\top\}) \cup (\mathfrak{X}_{S_l}^c \times \{\bot\}))$. From the local conclusion (4), $x \in \mathfrak{X}_{S_l}^c$. Hence, from the hypothesis 2.4, $\mathfrak{D}_{S_l}(x) = \{x\}$. Then, we can conclude that $\rho_1(x) = \rho(x)$.

Lets define ρ_1' such that:

$\diamond_1$ $\rho_1' = \rho_1 \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$.

(6) $\rho_1' = \rho'$.
It follows from the local conclusion (5) and the definitions of ρ_1' and ρ' (given in $\diamond_1$ and the local conclusion (2)). For all x in $\mathfrak{X}_l$, $\rho_1'(x) = \top = \rho'(x)$. For all x not in $\mathfrak{X}_l$, the local conclusion (5) implies $\rho_1'(x) = \rho_1(x) = \rho(x) = \rho'(x)$.

(7) $\forall x,\ \rho_1(x) = \bot \Rightarrow \sigma_1(x) = \sigma(x)$.
It follows from lemma A.6 and the local conclusion (3).

(8) $\mathfrak{D} \cup \mathcal{I}d = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d)) \cup \mathcal{I}d$.
It follows directly from the fact $\mathfrak{D} = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)) \cup \mathcal{I}d$ in local conclusion (2).

(9) $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash$ **while** e **do** S_l **done**$)$.
From the local conclusions (2) and (6), we can conclude that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho_1' \vdash S_l)$ with $\rho_1' = \rho_1 \sqcup ((\mathfrak{X}_l \times \top) \cup ((\mathfrak{X}_l)^c \times \bot))$. From the local conclusion (8), $\mathfrak{D} \cup \mathcal{I}d = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d)) \cup \mathcal{I}d$, from the local conclusion (2), $\mathfrak{X} = \mathfrak{X}_l$, and , from the local conclusion (1), $\sigma(e) = \texttt{true}$, then we can conclude that $\mathfrak{D} \cup \mathcal{I}d = \{\!\!\{(\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d \cup \mathcal{I}d)) \cup \mathcal{I}d, \mathcal{I}d\}\!\!\}^{\rho_1(e)}_{\sigma(e)}$ and $\mathfrak{X} = \{\!\!\{\mathfrak{X}_l, \emptyset\}\!\!\}^{\rho_1(e)}_{\sigma(e)}$. Hence, from the definition of $\models$ for while-statements, $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma, \rho_1 \vdash$ **while** e **do** S_l **done**$)$. Finally, the hypothesis 2.3 and the local conclusion (7) imply $(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash$ **while** e **do** S_l **done**$)$.

(10) $\rho_o' = (\lambda x. \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
This result is obtained by applying the inductive hypothesis on "$\sigma_1; \rho_1; \top \vdash$ **while** e **do** S_l **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o'$ " (from the local conclusion (3)) and "$(\mathfrak{D} \cup \mathcal{I}d, \mathfrak{X}) \models (\sigma_1, \rho_1 \vdash$ **while** e **do** S_l **done**$)$" (from the local conclusion (9)).

Lets define $\mathcal{F}_1^{\rho_o'}$ and $\mathcal{F}^{\rho_o}$ such that:

$\diamond_1$ $\mathcal{F}_1^{\rho_o'} = (\lambda x. \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$,
$\diamond_2$ $\mathcal{F}^{\rho_o} = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.

(•) $\rho_o(x) = (\bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) = \mathcal{F}^{\rho_o}(x)$.
From the local conclusion (10), $\rho_o'(x) = \mathcal{F}_1^{\rho_o'}(x)$, and, from the local conclusion (2), $\mathfrak{X} = \mathfrak{X}_l$. If x belongs to $\mathfrak{X}_l$, then $\rho(x) \sqcup \mathcal{F}_1^{\rho_o'}(x) = \top = \mathcal{F}^{\rho_o}(x)$. If x does not belong to $\mathfrak{X}_l$, then x does not belong to $\mathfrak{X}$ (from the local conclusion (2)) and, from the hypothesis 2.4, $\mathfrak{D}(x) = \{x\}$. Hence, if x does not belong to $\mathfrak{X}_l$, $\rho(x) \sqcup \mathcal{F}_1^{\rho_o'}(x) = \rho(x) \sqcup \bigsqcup_{y \in (\mathfrak{D} \cup \mathcal{I}d)(x)} \rho_1(y) = \rho(x) \sqcup \rho_1(x) = \rho(x)$ (the last equality comes from the local conclusion (5)). For similar reasons, if x does not belong to $\mathfrak{X}_l$ then $\mathcal{F}^{\rho_o}(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) = \rho(x)$.

(**E − WHILE$_{\mathbf{false}_\top}$**) then we can conclude that :

(1) S is "**while** e **do** S_l **done**" and there exists $(\mathfrak{D}_S, \mathfrak{X}_S)$ such that:

- "$\sigma; \rho \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \top$ "
- $(\mathfrak{D}_S, \mathfrak{X}_S) \models (\sigma, \rho \vdash S_l\ ;$ **while** e **do** S_l **done**$)$,

- $\rho_o = \rho \sqcup (\lambda x. \bigsqcup_{y \in \mathfrak{D}_S(x)} \rho(y)) \sqcup ((\mathfrak{X}_S \times \{\top\}) \cup (\mathfrak{X}_S^c \times \{\bot\}))$,

It follows directly from the definition of the rule ($\mathbf{E-WHILE_{false_\top}}$).

(2) There exist $(\mathfrak{D}_l, \mathfrak{X}_l)$ such that:

- $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ with $\rho' = \rho \sqcup ((\mathfrak{X}_l \times \{\top\}) \cup ((\mathfrak{X}_l)^c \times \{\bot\}))$,
- $\mathfrak{D} = (\mathfrak{D}_l \circ (\mathfrak{D} \cup \mathcal{I}d)) \cup \mathcal{I}d$,
- $\mathfrak{X} = \mathfrak{X}_l$

It follows from the global hypothesis $\star_2$, the local conclusion (1), the definition of $\models$, and the fact that $\rho(e) = \top$ (from the local conclusion (1)).

(3) There exist $(\mathfrak{D}_1, \mathfrak{X}_1)$ and $(\mathfrak{D}_2, \mathfrak{X}_2)$ such that:

- $(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_l)$
- $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho'' \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$ with $\rho'' = \rho \sqcup ((\mathfrak{X}_1 \times \{\top\}) \cup ((\mathfrak{X}_1)^c \times \{\bot\}))$
- $\mathfrak{D}_S = \mathfrak{D}_1 \circ \mathfrak{D}_2$
- $\mathfrak{X}_S = \mathfrak{X}_1 \cup \mathfrak{X}_2$

It follows from the fact that $(\mathfrak{D}_S, \mathfrak{X}_S) \models (\sigma, \rho \vdash S_l \,;\, \textbf{while } e \textbf{ do } S_l \textbf{ done})$ (given in the local conclusion (1)) and the definition of $\models$.

(4) There exist $(\mathfrak{D}_3, \mathfrak{X}_3)$ such that:

- $(\mathfrak{D}_3, \mathfrak{X}_3) \models (\sigma, \rho''' \vdash S_l)$ with $\rho''' = \rho'' \sqcup ((\mathfrak{X}_3 \times \{\top\}) \cup ((\mathfrak{X}_3)^c \times \{\bot\}))$,
- $\mathfrak{D}_2 = (\mathfrak{D}_3 \circ (\mathfrak{D}_2 \cup \mathcal{I}d)) \cup \mathcal{I}d$,
- $\mathfrak{X}_2 = \mathfrak{X}_3$

It follows from the fact that $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho'' \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$ (given in the local conclusion (3)), the definition of $\models$, and the fact that $\rho(e) = \top$ (from the local conclusion (1)).

(5) $\mathfrak{X}_1 \subseteq \mathfrak{X}$ and $\forall x \in \mathfrak{X}_1^c,\ \mathfrak{D}_1(x) = \{x\}$.
The lemma A.2, the facts that $(\mathfrak{D}_l, \mathfrak{X}_l) \models (\sigma, \rho' \vdash S_l)$ (from the local conclusion (2)) and $(\mathfrak{D}_1, \mathfrak{X}_1) \models (\sigma, \rho \vdash S_l)$ (from the local conclusion (3)), and the fact that $\rho \sqsubseteq \rho'$ (from the definition of ρ' in the local conclusion (2)) imply $\mathfrak{X}_1 \subseteq \mathfrak{X}_l$. Hence, from the local conclusion (2), $\mathfrak{X}_1 \subseteq \mathfrak{X}$. Finally, the hypothesis 2.4 and the local conclusion (3) imply $\forall x \in \mathfrak{X}_1^c, \mathfrak{D}_1(x) = \{x\}$.

(6) $\mathfrak{X} \subseteq \mathfrak{X}_2$.
It follows from the lemma A.2, the facts that $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$ (from the global hypothesis $\star_2$ and the local conclusion (1)) and $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho'' \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$ (from the local conclusion (3)), and the fact that $\rho \sqsubseteq \rho''$ (from the definition of ρ'' in the local conclusion (3)).

(7) $\rho''' = \rho \sqcup ((\mathfrak{X}_3 \times \{\top\}) \cup ((\mathfrak{X}_3)^c \times \{\bot\}))$.
The definitions of ρ'' and ρ''' (given in the local conclusions (3) and (4)) imply $\rho''' = \rho \sqcup ((\mathfrak{X}_1 \times \{\top\}) \cup ((\mathfrak{X}_1)^c \times \{\bot\})) \sqcup ((\mathfrak{X}_3 \times \{\top\}) \cup ((\mathfrak{X}_3)^c \times \{\bot\}))$. Additionally, the local conclusions (5), (6) and (4) imply $\mathfrak{X}_1 \subseteq \mathfrak{X}_3$. Hence, $((\mathfrak{X}_1 \times \{\top\}) \cup ((\mathfrak{X}_1)^c \times \{\bot\})) \sqcup ((\mathfrak{X}_3 \times \{\top\}) \cup ((\mathfrak{X}_3)^c \times \{\bot\}))$ equals $((\mathfrak{X}_3 \times \{\top\}) \cup ((\mathfrak{X}_3)^c \times \{\bot\}))$.

(8) $\mathfrak{D} = \mathfrak{D}_2$ and $\mathfrak{X} = \mathfrak{X}_2$.
The local conclusions (4) and (7) and the fact that $\rho(e) = \top$ (from the local conclusion (1)) imply $(\mathfrak{D}_2, \mathfrak{X}_2) \models (\sigma, \rho \vdash \textbf{while } e \textbf{ do } S_l \textbf{ done})$. Hence, the hypothesis 2.3 implies $(\mathfrak{D}, \mathfrak{X})$ equals $(\mathfrak{D}_2, \mathfrak{X}_2)$.

(9) $\mathfrak{D}_S = \mathfrak{D}_l \circ \mathfrak{D}$ and $\mathfrak{X}_S = \mathfrak{X}$.
The local conclusions (3) and (8) imply $\mathfrak{D}_S = \mathfrak{D}_l \circ \mathfrak{D}$. Finally, the local conclusions (5), (6), and (8) imply $\mathfrak{X}_1 \subseteq \mathfrak{X}$; hence, the local conclusion (3) implies $\mathfrak{X}_S = \mathfrak{X}$.

(10) $\rho_o = (\lambda x. \bigsqcup_{y \in (\mathfrak{D}_S \cup \mathcal{I}d)(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.
It follows directly from the local conclusions (1) and (9).

Lets define $\mathcal{F}_S^{\rho_o}$ and $\mathcal{F}^{\rho_o}$ such that:

$\diamond_1$ $\mathcal{F}_S^{\rho_o} = (\lambda x. \bigsqcup_{y \in (\mathfrak{D}_S \cup \mathcal{I}d)(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$,

$\diamond_2$ $\mathcal{F}^{\rho_o} = (\lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))$.

(•) $\rho_o(x) = (\bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)) \sqcup ((\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\}))(x) = \mathcal{F}^{\rho_o}(x)$.
From the local conclusion (10), $\rho_o(x) = \mathcal{F}_S^{\rho_o}(x)$ and, from the local conclusion (9), $\mathfrak{X}_S = \mathfrak{X}$. If x belongs to $\mathfrak{X}$, then $\mathcal{F}_S^{\rho_o}(x) = \top = \mathcal{F}^{\rho_o}(x)$. If x does not belong to $\mathfrak{X}$, then , from the hypothesis 2.4, $\mathfrak{D}_S(x) = \{x\}$. Hence, if x does not belong to $\mathfrak{X}$, $\mathcal{F}_S^{\rho_o}(x) = \bigsqcup_{y \in (\mathfrak{D}_S \cup \mathcal{I}d)(x)} \rho(y) = \rho(x)$. For similar reasons, if x does not belong to $\mathfrak{X}$ then $\mathcal{F}^{\rho_o}(x) = \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y) = \rho(x)$.

A.2 Soundness

A.2.1 Detection

Lemma A.5 (Public expressions are stable).
For all tag stores ρ, value stores σ_1 and σ_2, and expression e such that for all variable x the following holds $(\rho(x) = \bot) \Rightarrow \sigma_1(x) = \sigma_2(x)$, if $\rho(e) = \bot$ then $\sigma_1(e) = \sigma_2(e)$.

Proof. The proof is straightforward. It follows directly from the facts that expression evaluation is deterministic, the tag of an expression is the least upper bound of the tags of its free variables and that public ($\bot$) variables have the same value in σ_1 and σ_2.

Lemma A.6 (Tag of assigned variables contains $t^{\mathbf{pc}}$).
For all value stores σ_i, tag stores ρ_i, and statement S such that:

- $\sigma_i; \rho_i; t^{pc} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$

it is true that:

- $\forall x : t^{pc} \not\sqsubseteq \rho_o(x) \Rightarrow (\sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x))$

Proof. The proof goes by induction on the derivation tree of " $\sigma_i; \rho_i; t^{\mathrm{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ". Assume the lemma holds for any sub-derivation tree, if the last rule used is:

$(\mathbf{E}_{\mathcal{M}} - \mathbf{IF}_{\bot})$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and " $\sigma_i; \rho_i; t^{\mathrm{pc}} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\mathrm{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ".
It follows directly from the definition of the rule $(\mathbf{E}_{\mathcal{M}} - \mathbf{IF}_{\bot})$.

(•) $\forall x : t^{\mathrm{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
This is obtained by a simple induction because $t^{\mathrm{pc}} \sqcup \bot = t^{\mathrm{pc}}$.

$(\mathbf{E}_{\mathcal{M}} - \mathbf{IF}_{\top})$ then we can conclude that :

(1) S is "**if** e **then** S_1 **else** S_2 **end**" and there exist S_v and ρ_v such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $\forall x : \rho_v(x) \sqsubseteq \rho_o(x)$

It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}} - \mathbf{IF}_{\top}$).

(2) $\forall x : t^{\text{pc}} \sqcup \top \sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_v(x)$.
It follows directly from the inductive hypothesis and the local conclusion (1).

(•) $\forall x : t^{\text{pc}} \sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
As $t^{\text{pc}} \sqsubseteq \rho_o(x)$ implies $t^{\text{pc}} \sqcup \top \sqsubseteq \rho_o(x)$, the above result follows directly from the local conclusions (1) and (2).

($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{skip}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and $\sigma_o = \sigma_i \wedge \rho_i = \rho_o$.
It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{skip}}$).

(•) $\forall x : t^{\text{pc}} \sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
It follows directly from the local conclusion (1).

($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{true}_{\bot}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and there exist S', t'_e and ρ' such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S' \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{true}_{\bot}}$).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
It follows directly from the inductive hypothesis and the local conclusion (1).

($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{true}_{\top}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and there exist S', t'_e and ρ' such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup t'_e \vdash S' \Downarrow_{\mathcal{M}} \sigma_o : \rho'$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $\forall x : \rho'(x) \sqsubseteq \rho_o(x)$

It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{true}_{\top}}$).

(2) $\forall x : t^{\text{pc}} \sqcup t'_e \not\sqsubseteq \rho'(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho'(x)$.
It follows directly from the inductive hypothesis and the local conclusion (1).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
As $t^{\text{pc}} \not\sqsubseteq \rho_o(x)$ and $\rho'(x) \sqsubseteq \rho_o(x)$ imply $t^{\text{pc}} \sqcup t'_e \not\sqsubseteq \rho'(x)$, the above result follows from the local conclusions (1) and (2).

($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{false}_{\top}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**", $\sigma_o = \sigma_i$ and $\forall x : \rho_i(x) \sqsubseteq \rho_o(x)$.
It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}} - \mathbf{WHILE}_{\mathbf{false}_{\top}}$).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
It follows directly from the local conclusion (1).

($\mathbf{E}_{\mathcal{M}}$ – **SEQUENCE**) then we can conclude that :

(1) S is "S_1 ; S_2" and there exist σ_1 and ρ_1 such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- " $\sigma_1; \rho_1; t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}}$ – **SEQUENCE**).

(2) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_1(x) \Rightarrow \sigma_1(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_1(x)$ and $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_1(x) \wedge \rho_1(x) \sqsubseteq \rho_o(x)$.
It follows directly from the inductive hypothesis and the local conclusion (1).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
From the local conclusion (1), if $t^{\text{pc}} \not\sqsubseteq \rho_o(x)$ then $\rho_1(x) \sqsubseteq \rho_o(x)$. Hence, $t^{\text{pc}} \not\sqsubseteq \rho_1(x)$. Then, combining the results of the local conclusion (2), "$\sigma_o(x) = \sigma_1(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_1(x) \sqsubseteq \rho_o(x)$"

($\mathbf{E}_{\mathcal{M}}$ – **ASSIGN**) then we can conclude that :

(1) S is "id **:=** e" and there exist v_e, and t_e such that $\sigma_o = \sigma_i[id \mapsto v_e]$ and $\rho_o = \rho_i[id \mapsto t_e \sqcup t^{\text{pc}}]$.
It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}}$ – **ASSIGN**).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
From (1), if $x = id$ then $t^{\text{pc}} \sqsubseteq \rho_o(x)$. Otherwise, $x \neq id$ and $\sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) = \rho_o(x)$

($\mathbf{E}_{\mathcal{M}}$ – **SKIP**) then we can conclude that :

(1) S is "**skip**" and $\sigma_o = \sigma_i \wedge \rho_i = \rho_o$.
It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}}$ – **SKIP**).

(•) $\forall x : t^{\text{pc}} \not\sqsubseteq \rho_o(x) \Rightarrow \sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
It follows directly from the local conclusion (1).

Lemma A.7 (Correctness for information flow detection with semantics $\Downarrow_{\mathcal{M}}$).
For all:

- *variable x,*
- *value stores σ_i, σ_o, σ'_i, and σ'_o,*
- *tag stores ρ_i, ρ_o, ρ'_i, and ρ'_o,*
- *tags t^{pc} and $t^{pc'}$,*
- *and statement S*

such that:

$\star_1$ $\forall y : (\rho_i(y) = \bot) \Rightarrow \sigma_i(y) = \sigma'_i(y)$,

$\star_2$ $\sigma_i; \rho_i; t^{pc} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$,

$\star_3$ $\sigma'_i;\rho'_i;t^{pc'} \vdash S \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$,

$\star_4$ $\rho_o(x) = \bot$

it is true that:

- $\sigma_o(x) = \sigma'_o(x)$.

Proof. The proof goes by induction on the derivation tree of " $\sigma_i;\rho_i;t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ". Assume the lemma holds for any sub-derivation tree, if the last rule used is:

$(\mathbf{E-IF}_{\bot})$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exists $v \in \{true, false\}$ such that:

- " $\sigma_i;\rho_i \vdash e \Downarrow_{\mathcal{M}} v : \bot$ "
- " $\sigma_i;\rho_i;t^{\text{pc}} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i;\rho_i;t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule $(\mathbf{E-IF}_{\bot})$.

(2) There exist a value v', a tag t', and a tag store ρ' such that:

- " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'$ "
- " $\sigma'_i;\rho'_i;t^{\text{pc}'} \sqcup t' \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'$ "

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**if** e **then** S_{true} **else** S_{false} **end**" ($(\mathbf{E-IF}_{\bot})$ and $(\mathbf{E-IF}_{\top})$).

(3) $v = v'$.
This result comes from lemma A.5 applied to the derivations " $\sigma_i;\rho_i \vdash e \Downarrow_{\mathcal{M}} v : \bot$ " and " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'$ ". It is possible to apply it because of the global hypothesis $\star_1$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on " $\sigma_i;\rho_i;t^{\text{pc}} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma_i;\rho_i;t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i;\rho'_i;t^{\text{pc}'} \sqcup t' \vdash S_v \Downarrow_{\mathcal{M}} \sigma'_o : \rho'$ " (because $v = v'$).

$(\mathbf{E-IF}_{\top})$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exist $v \in \{true, false\}$, two tag store ρ_v and ρ_e, and an analysis result $(\mathfrak{D}, \mathfrak{X})$ such that:

- " $\sigma_i;\rho_i;t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " is a sub-derivation tree of " $\sigma_i;\rho_i;t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $[\![\sigma_i;\rho_i \vdash S_{\neg v}]\!]^{\sharp_G} = (\mathfrak{D}, \mathfrak{X})$
- $\rho_v(x) \sqsubseteq \rho_o(x)$
- $\rho_e = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}) \times \{\bot\})$
- $\rho_e(x) \sqsubseteq \rho_o(x)$

It follows directly from the definition of the rule $(\mathbf{E-IF}_{\bot})$.

(2) There exist a value $v' \in \{true, false\}$, a tag store $\rho'_{v'}$, and a tag t'_e such that:

- " $\sigma'_i;\rho'_i;t^{\text{pc}'} \sqcup t'_e \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_{v'}$ "

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**if** e **then** S_{true} **else** S_{false} **end**" ($(\mathbf{E-IF}_\bot)$ and $(\mathbf{E-IF}_\top)$).

Case 1: $v = v'$

(a) $\rho_v(x) = \bot$.
It follows from the local conclusion (1) and the global hypothesis $\star_4$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on "$\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$" (sub-derivation tree of "$\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$") and "$\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_{v'}$". It is possible to apply the inductive hypothesis because of the local conclusion (a) and the fact that, by the case hypothesis, $v = v'$).

Case 2: $v \neq v'$ (which implies that $v' = \neg v$)

(a) $\sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.
From the global hypothesis $\star_4$, $\rho_o(x) = \bot$. Hence, from lemma A.6 applied to "$\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$", $\sigma_o(x) = \sigma_i(x) \wedge \rho_i(x) \sqsubseteq \rho_o(x)$.

(b) $\sigma'_o(x) = \sigma'_i(x)$.
From the local conclusion (1), the global hypothesis $\star_4$, and the definition of ρ_e in the local conclusion (1), $x \notin \mathfrak{X}$. Hence, from the local conclusion (1) and the hypothesis 2.1 applied to "$\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_{v'}$", $\sigma'_o(x) = \sigma'_i(x)$.

(c) $\sigma_i(x) = \sigma'_i(x)$.
From the local conclusion (a) and the global hypothesis $\star_4$, we get that $\rho_i(x) = \bot$. Hence, from the global hypothesis $\star_1$, we get that $\sigma_i(x) = \sigma'_i(x)$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (a), (b), and (c).

($\mathbf{E-WHILE}_{\mathbf{skip}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and:

- "$\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \bot$"
- $\sigma_i = \sigma_o$ and $\rho_i = \rho_o$

It follows directly from the definition of the rule $(\mathbf{E-IF}_\bot)$.

(2) There exist a value v' and a tag t' such that:

- "$\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'$"

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" ($(\mathbf{E-WHILE}_{\mathbf{skip}})$, $(\mathbf{E-WHILE}_{\mathbf{true}_\bot})$, $(\mathbf{E-WHILE}_{\mathbf{true}_\top})$ and $(\mathbf{E-WHILE}_{\mathbf{false}_\top})$).

(3) $\sigma'_i = \sigma'_o$.
From lemma A.5 and the local conclusions (1) and (2), we get $v' = false$. Hence, from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever "$\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : t'$" ($(\mathbf{E-WHILE}_{\mathbf{skip}})$ and $(\mathbf{E-WHILE}_{\mathbf{false}_\top})$), $\sigma'_i = \sigma'_o$.

(4) $\sigma_i(x) = \sigma'_i(x)$.
From the local conclusion (1) and the global hypothesis $\star_4$, we get that $\rho_i(x) = \bot$. Hence, from the global hypothesis $\star_1$, we get that $\sigma_i(x) = \sigma'_i(x)$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (1), (3), and (4).

($\mathbf{E-WHILE_{true_\bot}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \bot$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E-WHILE_{true_\bot}}$).

(2) There exist a value v' and a tag t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" (($\mathbf{E-WHILE_{skip}}$), ($\mathbf{E-WHILE_{true_\bot}}$), ($\mathbf{E-WHILE_{true_\top}}$) and ($\mathbf{E-WHILE_{false_\top}}$)).

(3) $v' = \texttt{true}$.
It follows directly from the local conclusions (1) and (2), and lemma A.5.

(4) There exists a tag store ρ'_l such that:

- " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ "

It follows from the global hypothesis $\star_3$, the local conclusions (1), (2) and (3), and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : t'_e$ " (($\mathbf{E-WHILE_{true_\bot}}$) and ($\mathbf{E-WHILE_{true_\top}}$)).

(•) $\sigma_o(x) = \sigma'_o(x)$.
By applying the inductive hypothesis on the derivations " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ ", it is possible to deduce that $\sigma_o(x) = \sigma'_o(x)$.

($\mathbf{E-WHILE_{true_\top}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and there exist a tag t_e and a tag store ρ_l such that:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \top$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $\rho_l \sqsubseteq \rho_o$

It follows directly from the definition of the rule ($\mathbf{E-WHILE_{true_\top}}$).

(2) $\rho_l(x) = \bot$.
It follows from the local conclusion (1) and the global hypothesis $\star_4$.

(3) There exist a value v' and a tag t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" ($(\mathbf{E-WHILE_{skip}})$, $(\mathbf{E-WHILE_{true_\perp}})$, $(\mathbf{E-WHILE_{true_\top}})$ and $(\mathbf{E-WHILE_{false_\top}})$).

Case 1: $v' = \texttt{true}$

(a) There exists a tag store ρ'_l such that:

- " $\sigma'_i;\rho'_i;t^{pc'} \sqcup t'_e \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ "

It follows from the global hypothesis $\star_3$, the local conclusions (1) and (3), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : t'_e$ " ($(\mathbf{E-WHILE_{true_\perp}})$ and $(\mathbf{E-WHILE_{true_\top}})$).

(•) $\sigma_o(x) = \sigma'_o(x)$.
By applying the inductive hypothesis on the derivations " $\sigma_i;\rho_i;t^{pc} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ " and " $\sigma'_i;\rho'_i;t^{pc'} \sqcup t'_e \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ ", it is possible to deduce, using the local conclusion (2), that $\sigma_o(x) = \sigma'_o(x)$.

Case 2: $v' = \texttt{false}$

(a) $\sigma_o(x) = \sigma_i(x)$.
It follows directly from the local conclusions (1), the fact that "$t^{pc} \sqcup \top \not\sqsubseteq \rho_l(x)$" (from the local conclusion (2)), and the lemma A.6 applied to " $\sigma_i;\rho_i;t^{pc} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ ".

(b) $\sigma'_o(x) = \sigma'_i(x)$.
From the global hypothesis $\star_3$, the local conclusions (1) and (3), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : t'_e$ " ($(\mathbf{E-WHILE_{skip}})$ and $(\mathbf{E-WHILE_{false_\top}})$), $\sigma'_i = \sigma'_o$.

(c) $\sigma_i(x) = \sigma'_i(x)$.
From the local conclusion (2) and lemma A.6 applied to " $\sigma_i;\rho_i;t^{pc} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ ", $\rho_i(x) = \bot$. Hence, from the global hypothesis $\star_1$, we get that $\sigma_i(x) = \sigma'_i(x)$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (a), (b), and (c).

$(\mathbf{E-WHILE_{false_\top}})$ then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and there exist two tag stores ρ_1 and ρ_e such that:

- " $\sigma_i;\rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \top$ "
- $[\![\sigma_i;\rho_i \vdash S_1$; **while** e **do** S_1 **done**$]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$
- $\sigma_o = \sigma_i$
- $\rho_e = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}) \times \{\bot\})$
- $\rho_e(x) \sqsubseteq \rho_o(x)$ and $\rho_i(x) \sqsubseteq \rho_o(x)$

It follows directly from the definition of the rule $(\mathbf{E-WHILE_{true}})$.

(2) There exist a value $v' \in \{\texttt{true}, \texttt{false}\}$ and a tag t'_e such that:

- " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_3$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" ($(\mathbf{E-WHILE_{skip}})$, $(\mathbf{E-WHILE_{true}})$ and $(\mathbf{E-WHILE_{false_\top}})$).

(3) $\sigma_i(x) = \sigma'_i(x)$.
From the local conclusion (1) ($\rho_i(x) \sqsubseteq \rho_o(x)$) and the global hypothesis $\star_4$, $\rho_i(x) = \bot$. Hence, from the global hypothesis $\star_1$, we get that $\sigma_i(x) = \sigma'_i(x)$.

Case 1: $v' = \texttt{false}$

(a) $\sigma'_o = \sigma'_i$.
From the global hypothesis $\star_3$, the local conclusions (1) and (2), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : t'_e$ " ((**E** − **WHILE**$_{\textbf{skip}}$) and (**E** − **WHILE**$_{\textbf{false}_\top}$)), $\sigma'_i = \sigma'_o$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (1) ($\sigma_o = \sigma_i$), (3), and (a).

Case 2: $v' = \texttt{true}$

(a) $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$
It follows from the global hypothesis $\star_3$, the local conclusions (1) and (2), the case hypothesis, and the definition of the rule applying to "**while** e **do** S_1 **done**" whenever e evaluates to `true` ((**E** − **WHILE**$_{\textbf{true}}$)).

(b) $x \notin \mathfrak{X}$.
From the local conclusion (1) ($\rho_e(x) \sqsubseteq \rho_o(x)$) and the global hypothesis $\star_4$, we get that $\rho_e(x) = \bot$. Hence, from the definition of ρ_e given in the local conclusion (1), $x \notin \mathfrak{X}$.

(c) $\sigma'_o(x) = \sigma'_i(x)$.
From the local conclusion (b), the hypothesis 2.1 applied to the analysis from the local conclusion (1) and the derivation from the local conclusion (a), $\sigma'_o(x) = \sigma'_i(x)$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (1), (3), and (c).

(**E** − **SEQUENCE**) then we can conclude that :

(1) S is "S_1 ; S_2" and there exist σ_1 and ρ_1 such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- " $\sigma_1; \rho_1; t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule (**E**$_{\mathcal{M}}$ − **SEQUENCE**).

(2) There exist σ'_1 and ρ'_1 such that:

- " $\sigma'_i; \rho'_i; t^{\text{pc}'} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma'_1 : \rho'_1$ "
- " $\sigma'_1; \rho'_1; t^{\text{pc}'} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ "

It follows directly from the global hypothesis $\star_3$, the local conclusion (1), and the definition of the only rule applying to "S_1 ; S_2" ((**E**$_{\mathcal{M}}$ − **SEQUENCE**)).

(3) $\forall y : (\rho_1(y) = \bot) \Rightarrow \sigma_1(y) = \sigma'_1(y)$.
This result is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma'_1 : \rho'_1$ " for any variable y such that $\rho_1(y) = \bot$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on " $\sigma_1;\rho_1;t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma_i;\rho_i;t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_1;\rho'_1;t^{\text{pc}'} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ " using the local conclusion (3).

(**E − ASSIGN**) then we can conclude that :

(1) S is "id **:=** e".

Case 1: $x \neq id$

(a) $\sigma_o(x) = \sigma_i(x)$ and $\rho_o(x) = \rho_i(x)$.
It follows directly from the case hypothesis and the definition of the rule (**E − ASSIGN**).

(b) $\sigma'_o(x) = \sigma'_i(x)$.
It follows directly from the global hypothesis $\star_3$, the local conclusion (1), the definition of the only rule applying to "id **:=** e" ((**E − ASSIGN**)), and the case hypothesis.

(•) $\sigma_o(x) = \sigma'_o(x)$.
As $\rho_o(x) = \bot$, from the global hypothesis $\star_4$, the local conclusion (a) implies $\rho_i(x) = \bot$. Then, from the global hypothesis $\star_1$, $\sigma_i(x) = \sigma'_i(x)$. Hence, from the local conclusions (a) and (b), $\sigma_o(x) = \sigma'_o(x)$.

Case 2: $x = id$

(a) There exist v and t_e such that:
- " $\sigma_i;\rho_i \vdash e \Downarrow_{\mathcal{M}} v : t_e$ "
- $\sigma_o(x) = v$
- $t_e \sqsubseteq \rho_o(x)$

It follows directly from the case hypothesis and the definition of the rule (**E − ASSIGN**).

(b) There exist v' and t'_e such that:
- " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "
- $\sigma'_o(x) = v'$

It follows directly from the global hypothesis $\star_3$, the local conclusion (1), the definition of the only rule applying to "id **:=** e" ((**E − ASSIGN**)), and the case hypothesis.

(c) $v = v'$.
This result comes from lemma A.5 applied to the derivations " $\sigma_i;\rho_i \vdash e \Downarrow_{\mathcal{M}} v : t_e$ " and " $\sigma'_i;\rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ ". It is possible to apply it because of the global hypothesis $\star_1$ and the fact that, from the global hypothesis $\star_4$ and the local conclusion (a), $t_e = \bot$.

(•) $\sigma_o(x) = \sigma'_o(x)$.
It follows directly from the local conclusions (a), (b), and (c).

(**E − SKIP**) then we can conclude that :

(1) S is "**skip**", $\sigma_o = \sigma_i$ and $\rho_o = \rho_i$.
It follows directly from the definition of the rule (**E − SKIP**).

(2) $\sigma'_o = \sigma'_i$.
It follows directly from the global hypothesis $\star_3$, the local conclusion (1), and the definition of the only rule applying to "**skip**" ((**E − SKIP**)).

(•) $\sigma_o(x) = \sigma'_o(x)$.
From the local conclusion (1) and the global hypothesis $\star_4$, $\rho_i(x) = \bot$. Then, from the global hypothesis $\star_1$, $\sigma_i(x) = \sigma'_i(x)$. Hence, from the local conclusions (1) and (2), $\sigma_o(x) = \sigma'_o(x)$.

A.2.2 Correction

Lemma A.8 (Expression tag is deterministic).
For all expressions e and tag stores ρ and ρ', if $\rho = \rho'$ then $\rho(e) = \rho'(e)$.

Proof. The proof is straightforward. It follows directly from the facts that the evaluation of an expression's tag is deterministic. It is the least upper bound of the tags of its free variables.

Lemma A.9 (Correctness for information flow correction with semantics $\Downarrow_{\mathcal{M}}$).
For all:

- *variable x,*
- *value stores σ_i, σ_o, σ'_i, and σ'_o,*
- *tag stores ρ_i, ρ_o, ρ'_i, and ρ'_o,*
- *tags t^{pc} and $t^{pc'}$,*
- *and statement S*

such that:

$\star_1$ $\forall y : (\rho_i(y) = \bot) \Rightarrow \sigma_i(y) = \sigma'_i(y)$,

$\star_2$ $\rho_i = \rho'_i$,

$\star_3$ $t^{pc} = t^{pc'}$,

$\star_4$ $\sigma_i; \rho_i; t^{pc} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$,

$\star_5$ $\sigma'_i; \rho'_i; t^{pc'} \vdash S \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$

it is true that:

- $\rho_o(x) = \rho'_o(x)$.

Proof. The proof goes by induction on the derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ". Assume the lemma holds for any sub-derivation tree, if the last rule used is:

$(\mathbf{E - IF}_{\bot})$ then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exists $v \in \{true, false\}$ such that:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} v : \bot$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule $(\mathbf{E - IF}_{\bot})$.

(2) There exist a value v' and a tag t' such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**if** e **then** S_{true} **else** S_{false} **end**" (($\mathbf{E} - \mathbf{IF}_{\bot}$) and ($\mathbf{E} - \mathbf{IF}_{\top}$)).

(3) $t' = \bot$ and $v = v'$.
It follows directly from the local conclusion (1) and (2), lemma A.8, and lemma A.5.

(4) " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ " It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), and the definition of the rule applying to "**if** e **then** S_{true} **else** S_{false} **end**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v : \bot$ " (($\mathbf{E} - \mathbf{IF}_{\bot}$)).

(•) $\rho_o(x) = \rho'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \bot \vdash S_v \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ ".

($\mathbf{E} - \mathbf{IF}_{\top}$) then we can conclude that :

(1) S is "**if** e **then** S_{true} **else** S_{false} **end**" and there exist $v \in \{true, false\}$, two tag stores ρ_v and ρ_e, and an analysis result $(\mathfrak{D}, \mathfrak{X})$ such that:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} v : \top$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $[\![\sigma_i; \rho_i \vdash S_{\neg v}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$
- $\rho_{\neg v} = \lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho_i(y)$
- $\rho_e = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}) \times \{\bot\})$
- $\rho_o = \rho_v \sqcup \rho_{\neg v} \sqcup \rho_e$

It follows directly from the definition of the rule ($\mathbf{E} - \mathbf{IF}_{\top}$).

(2) There exist a value $v' \in \{true, false\}$ and a tag t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**if** e **then** S_{true} **else** S_{false} **end**" (($\mathbf{E} - \mathbf{IF}_{\bot}$) and ($\mathbf{E} - \mathbf{IF}_{\top}$)).

(3) $t'_e = \top$.
It follows directly from the local conclusions (1) and (2), and the lemma A.8.

(4) There exists an analysis result $(\mathfrak{D}', \mathfrak{X}')$ and three tag stores $\rho'_{v'}$, $\rho'_{\neg v'}$ and ρ'_e such that:

- " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_{v'}$ "
- $[\![\sigma'_i; \rho'_i \vdash S_{\neg v'}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}', \mathfrak{X}')$
- $\rho'_{\neg v'} = \lambda x. \bigsqcup_{y \in \mathfrak{D}'(x)} \rho'_i(y)$
- $\rho'_e = (\mathfrak{X}' \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}') \times \{\bot\})$
- $\rho'_o = \rho'_{v'} \sqcup \rho'_{\neg v'} \sqcup \rho'_e$

It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), and the definition of the rule applying to "**if** e **then** S_{true} **else** S_{false} **end**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : \top$ " ($(\mathbf{E-IF}_{\top})$).

Case 1: $v = v'$

(a) $\rho_{\neg v}(x) = \rho'_{\neg v'}(x)$ and $\rho_e(x) = \rho'_e(x)$.
It follows from the definitions of $\rho_{\neg v}$, $\rho'_{\neg v'}$, ρ_e, and ρ'_e (given in the local conclusions (1) and (2)), the case hypothesis, the global hypothesis $\star_2$, and the hypothesis 2.3.

(b) $\rho_v(x) = \rho'_{v'}(x)$.
This result is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_v \Downarrow_{\mathcal{M}} \sigma_o : \rho_v$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup t'_e \vdash S_{v'} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_{v'}$ " using the result of the local conclusion (3) and the case hypothesis.

(•) $\rho_o(x) = \rho'_o(x)$.
It follows directly from the definitions of ρ_o and ρ'_o (given in the local conclusions (1) and (4)) and the local conclusions (a) and (b).

Case 2: $v \neq v'$ (which implies that $v' = \neg v$)

(a) $\rho_v = \rho'_{\neg v'} \sqcup \rho'_e$.
It follows from the definitions of ρ_v, $\rho'_{\neg v'}$, and ρ'_e (given in the local conclusions (1) and (4)), the case hypothesis, and the hypotheses 2.3 and 2.2.

(b) $\rho'_{v'} = \rho_{\neg v} \sqcup \rho'_e$.
It follows from the definitions of $\rho'_{v'}$, $\rho_{\neg v}$, and ρ_e (given in the local conclusions (1) and (4)), the case hypothesis, the local conclusion (3) and the hypotheses 2.3 and 2.2.

(•) $\rho_o(x) = \rho'_o(x)$.
It follows directly from the definitions of ρ_o and ρ'_o (given in the local conclusions (1) and (4)) and the local conclusions (a) and (b).

($\mathbf{E-WHILE}_{\mathbf{skip}}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \bot$ "
- $\rho_i = \rho_o$

It follows directly from the definition of the rule ($\mathbf{E-IF}_{\bot}$).

(2) There exist a value v' and a tag t' such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" ($(\mathbf{E-WHILE}_{\mathbf{skip}})$, $(\mathbf{E-WHILE}_{\mathbf{true}_{\bot}})$, $(\mathbf{E-WHILE}_{\mathbf{true}_{\top}})$ and $(\mathbf{E-WHILE}_{\mathbf{false}_{\top}})$).

(3) $v' = \texttt{false}$ and $t' = \bot$.
It follows directly from the local conclusions (1) and (2), lemma A.5 and lemma A.8.

(4) $\rho'_o = \rho'_i$.
It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \bot$ " ($(\mathbf{E-WHILE}_{\mathbf{skip}})$).

(5) $\rho_i(x) = \rho'_i(x)$.
It follows directly from the global hypothesis $\star_2$.

(•) $\rho_o(x) = \rho'_o(x)$.
It follows directly from the local conclusions (1), (4), and (5).

($\mathbf{E-WHILE}_{\mathbf{true}_\bot}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \bot$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E-WHILE}_{\mathbf{true}_\bot}$).

(2) There exist a value v' and a tag t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" (($\mathbf{E-WHILE}_{\mathbf{skip}}$), ($\mathbf{E-WHILE}_{\mathbf{true}_\bot}$), ($\mathbf{E-WHILE}_{\mathbf{true}_\top}$) and ($\mathbf{E-WHILE}_{\mathbf{false}_\top}$)).

(3) $t'_e = \bot$ and $v' = true$.
It follows directly from the local conclusions (1) and (2), lemma A.8, and lemma A.5.

(4) " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ " It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), and the definition of the rule applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} true : \bot$ " (($\mathbf{E-WHILE}_{\mathbf{true}_\bot}$)).

(•) $\rho_o(x) = \rho'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \bot \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ ".

($\mathbf{E-WHILE}_{\mathbf{true}_\top}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \top$ "
- " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- $\rho_o = \rho_i \sqcup \rho_l$

It follows directly from the definition of the rule ($\mathbf{E-WHILE}_{\mathbf{true}_\top}$).

(2) There exist a value v' and a tag t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" (($\mathbf{E-WHILE}_{\mathbf{skip}}$), ($\mathbf{E-WHILE}_{\mathbf{true}_\bot}$), ($\mathbf{E-WHILE}_{\mathbf{true}_\top}$) and ($\mathbf{E-WHILE}_{\mathbf{false}_\top}$)).

(3) $t'_e = \top$.
It follows directly from the local conclusions (1) and (2), and lemma A.8.

Case 1: $v' = \texttt{true}$

(a) There exists a tag store ρ'_l such that:
- " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ "
- $\rho'_o = \rho'_i \sqcup \rho'_l$

It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \top$ " (($\mathbf{E-WHILE}_{\mathbf{true}_\top}$)).

(b) $\rho_l = \rho'_l$.
This is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{\text{pc}} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma_o : \rho_l$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \top \vdash S_1$; **while** e **do** S_1 **done** $\Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ ".

(•) $\rho_o = \rho'_o$.
It follows directly from the local conclusions (1), (a), and (b), and the global hypothesis $\star_2$.

Case 2: $v' = \texttt{false}$

(a) There exists an analysis result $(\mathfrak{D}', \mathfrak{X}')$ and two tag stores ρ'_l and ρ'_e such that:
- $[\![\sigma'_i; \rho'_i \vdash S_1$; **while** e **do** S_1 **done**$]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}', \mathfrak{X}')$
- $\rho'_l = \lambda x. \bigsqcup_{y \in \mathfrak{D}'(x)} \rho'_i(y)$
- $\rho'_e = (\mathfrak{X}' \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}') \times \{\bot\})$
- $\rho'_o = \rho'_i \sqcup \rho'_l \sqcup \rho'_e$

It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), the case hypothesis, and the definition of the rule applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \top$ " (($\mathbf{E-WHILE}_{\mathbf{false}_\top}$)).

(b) $\rho_l = \rho'_l \sqcup \rho'_e$.
It follows from the definitions of ρ_l, ρ'_l, and ρ'_e (given in the local conclusions (1) and (a)), and the hypotheses 2.3 and 2.2.

(•) $\rho_o = \rho'_o$.
It follows directly from the local conclusions (1), (a), and (b).

($\mathbf{E-WHILE}_{\mathbf{false}_\top}$) then we can conclude that :

(1) S is "**while** e **do** S_1 **done**" and there exist two tag stores ρ_l and ρ_e such that:
- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \top$ "
- $[\![\sigma_i; \rho_i \vdash S_1$; **while** e **do** S_1 **done**$]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$
- $\rho_l = \lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho_i(y)$
- $\rho_e = (\mathfrak{X} \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}) \times \{\bot\})$
- $\rho_o = \rho_i \sqcup \rho_l \sqcup \rho_e$

It follows directly from the definition of the rule ($\mathbf{E-WHILE}_{\mathbf{true}}$).

(2) There exist a value $v' \in \{\texttt{true}, \texttt{false}\}$ and a tag t'_e such that:
- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "

It follows from the global hypothesis $\star_5$, the local conclusion (1), and the definitions of the rules applying to "**while** e **do** S_1 **done**" ((**E – WHILE**$_{\textbf{skip}}$), (**E – WHILE**$_{\textbf{true}_\bot}$), (**E – WHILE**$_{\textbf{true}_\top}$) and (**E – WHILE**$_{\textbf{false}_\top}$)).

(3) $t'_e = \top$.
It follows directly from the local conclusions (1) and (2), and lemma A.8.

Case 1: $v' = \texttt{false}$

(a) there exist two tag stores ρ'_l and ρ'_e such that:

- $[\![\sigma'_i; \rho'_i \vdash S_1 \ ; \ \textbf{while}\ e\ \textbf{do}\ S_1\ \textbf{done}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}', \mathfrak{X}')$
- $\rho'_l = \lambda x. \bigsqcup_{y \in \mathfrak{D}'(x)} \rho'_i(y)$
- $\rho'_e = (\mathfrak{X}' \times \{\top\}) \cup ((\mathbb{I}d - \mathfrak{X}') \times \{\bot\})$
- $\rho'_o = \rho'_i \sqcup \rho'_l \sqcup \rho'_e$

It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{false} : \top$ " ((**E – WHILE**$_{\textbf{false}_\top}$)).

(b) $\rho_l = \rho'_l$ and $\rho_e = \rho'_e$.
It follows from the definitions of ρ_l, ρ'_l, ρ_e, and ρ'_e (given in the local conclusions (1) and (a)), the global hypothesis $\star_2$, and the hypothesis 2.3.

(•) $\rho_o = \rho'_o$.
It follows directly from the definitions of ρ_o and ρ'_o (given in the local conclusions (1) and (a)), the local conclusions (b), and the global hypothesis $\star_2$.

Case 2: $v' = \texttt{true}$

(a) There exists a tag store ρ'_l such that:

- " $\sigma'_i; \rho'_i; t^{\text{pc}'} \sqcup \top \vdash S_1 \ ; \ \textbf{while}\ e\ \textbf{do}\ S_1\ \textbf{done} \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_l$ "
- $\rho'_o = \rho'_i \sqcup \rho'_l$

It follows from the global hypothesis $\star_5$, the local conclusions (1), (2), and (3), the case hypothesis, and the definitions of the rules applying to "**while** e **do** S_1 **done**" whenever " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} \texttt{true} : \top$ " ((**E – WHILE**$_{\textbf{true}_\top}$)).

(b) $\rho'_l = \rho_l \sqcup \rho_e$.
It follows from the definitions of ρ'_l, ρ_l, and ρ_e (given in the local conclusions (1) and (a)), and the hypotheses 2.3 and 2.2.

(•) $\rho_o = \rho'_o$.
It follows directly from the local conclusions (1), (a), and (b), and the global hypothesis $\star_2$

(**E – SEQUENCE**) then we can conclude that :

(1) S is "$S_1 \ ; \ S_2$" and there exist σ_1 and ρ_1 such that:

- " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "
- " $\sigma_1; \rho_1; t^{\text{pc}} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " is a sub-derivation tree of " $\sigma_i; \rho_i; t^{\text{pc}} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ "

It follows directly from the definition of the rule ($\mathbf{E}_{\mathcal{M}}$ − **SEQUENCE**).

(2) There exist σ'_1 and ρ'_1 such that:

- " $\sigma'_i; \rho'_i; t^{pc'} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma'_1 : \rho'_1$ "
- " $\sigma'_1; \rho'_1; t^{pc'} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ "

It follows directly from the global hypothesis $\star_5$, the local conclusion (1), and the definition of the only rule applying to "S_1 ; S_2" (($\mathbf{E}_{\mathcal{M}}$ − **SEQUENCE**)).

(3) $\forall y : (\rho_1(y) \sqsubseteq \bot) \Rightarrow \sigma_1(y) = \sigma'_1(y)$.
This result is obtained by applying lemma A.7 on " $\sigma_i; \rho_i; t^{pc} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " and " $\sigma'_i; \rho'_i; t^{pc'} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma'_1 : \rho'_1$ " for any variable y such that $\rho_1(y) \sqsubseteq \bot$.

(4) $\rho_1 = \rho'_1$.
This result is obtained by applying the inductive hypothesis on " $\sigma_i; \rho_i; t^{pc} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma_1 : \rho_1$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{pc} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_i; \rho'_i; t^{pc'} \vdash S_1 \Downarrow_{\mathcal{M}} \sigma'_1 : \rho'_1$ " for any variable.

(•) $\rho_o(x) = \rho'_o(x)$.
The conclusion is obtained by applying the inductive hypothesis on " $\sigma_1; \rho_1; t^{pc} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ " (sub-derivation tree of " $\sigma_i; \rho_i; t^{pc} \vdash S \Downarrow_{\mathcal{M}} \sigma_o : \rho_o$ ") and " $\sigma'_1; \rho'_1; t^{pc'} \vdash S_2 \Downarrow_{\mathcal{M}} \sigma'_o : \rho'_o$ " using the local conclusions (3) and (4).

(**E** − **ASSIGN**) then we can conclude that :

(1) S is "id **:=** e".

Case 1: $x \neq id$

(a) $\rho_o(x) = \rho_i(x)$.
It follows directly from the case hypothesis and the definition of the rule (**E** − **ASSIGN**).

(b) $\rho'_o(x) = \rho'_i(x)$.
It follows directly from the global hypothesis $\star_5$, the local conclusion (1), the definition of the only rule applying to "id **:=** e" ((**E** − **ASSIGN**)), and the case hypothesis.

(•) $\rho_o(x) = \rho'_o(x)$.
From the global hypothesis $\star_2$, $\rho_i(x) = \rho'_i(x)$. Hence, from the local conclusions (a) and (b), $\rho_o(x) = \rho'_o(x)$.

Case 2: $x = id$

(a) There exist v and t_e such that:

- " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} v : t_e$ "
- $\rho_o(x) = t_e \sqcup t^{pc}$

It follows directly from the case hypothesis and the definition of the rule (**E** − **ASSIGN**).

(b) There exist v' and t'_e such that:

- " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ "
- $\rho'_o(x) = t'_e \sqcup t^{pc'}$

It follows directly from the global hypothesis $\star_5$, the local conclusion (1), the definition of the only rule applying to "id **:=** e" ((**E** − **ASSIGN**)), and the case hypothesis.

(c) $t_e = t'_e$.
This result comes from lemma A.8 applied to the derivations " $\sigma_i; \rho_i \vdash e \Downarrow_{\mathcal{M}} v : t_e$ " and " $\sigma'_i; \rho'_i \vdash e \Downarrow_{\mathcal{M}} v' : t'_e$ ".

(•) $\rho_o(x) = \rho'_o(x)$.
From the global hypothesis $\star_3$, $t^{\text{pc}'} = t^{\text{pc}}$. Then, the above result follows directly from the local conclusions (a), (b), and (c).

(**E − SKIP**) then we can conclude that :

(1) S is "**skip**" and $\rho_o = \rho_i$.
It follows directly from the definition of the rule (**E − SKIP**).

(2) $\rho'_o = \rho'_i$.
It follows directly from the global hypothesis $\star_5$, the local conclusion (1), and the definition of the only rule applying to "**skip**" ((**E − SKIP**)).

(•) $\rho_o(x) = \rho'_o(x)$.
From the global hypothesis $\star_2$, $\rho_i(x) = \rho'_i(x)$. Hence, from the local conclusions (1) and (2), $\rho_o(x) = \rho'_o(x)$.

A.3 Transparency

A.3.1 Lemmas and Figure Reused from Earlier Work

The following figure and lemmas are taken from previous work [Le Guernic et al., 2006b].

Lemma A.10 (Automaton simulates execution in secret context).
For all statement S, automaton state $q = (V, w)$, and value store σ, if $(q, \sigma) \vdash S \overset{o}{\Rightarrow}_{\mathcal{M}} (q_f, \sigma_f)$ and $w \notin \{\bot\}^\star$ then $(q, \texttt{not}\ S) \rightarrow q_f$.

Lemma A.11 (Same context before and after an execution).
For all statement S, automaton states $q = (V, w)$ and $q_f = (V_f, w_f)$, and value store σ, if $(q, \sigma) \vdash S \overset{o}{\Rightarrow}_{\mathcal{M}} (q_f, \sigma_f)$ then $w_f = w$.

Lemma A.12 (WDKL (while-dilemma killer lemma)). *For all statement S, expression e, automaton state $q = (V, w)$, and value store σ, if "$\mathcal{V}(e) \cap V \neq \emptyset$" and "$(q, \sigma) \vdash$ **while** e **do** S **done** $\overset{o}{\Rightarrow}_{\mathcal{M}} (q_f, \sigma_f)$" then:*

- $o = \epsilon$
- *Let $(V_f, w_f) = q_f$ in $V \subseteq V_f$*

Lemma A.13 (Same context before and after an execution). *For all statement S, automaton states $q = (V, w)$ and $q_f = (V_f, w_f)$, and value store σ, if $(q, \sigma) \vdash S \overset{o}{\Rightarrow}_{\mathcal{M}} (q_f, \sigma_f)$ then $w_f = w$.*

A.3.2 Proofs of the Partial Transparency

Lemma A.14 (More Precise Static Analysis).
For all command C, value store σ, tag store ρ, and analysis result $(\mathfrak{D}, \mathfrak{X})$, if $(\mathfrak{D}, \mathfrak{X}) \models (\sigma, \rho \vdash C)$ then, for all variable x in $\mathfrak{X}$, x belongs to modified*(C).*

$((V, w),\ \texttt{branch}\ e) \xrightarrow{ACK} (V, w\top)$ iff $\mathcal{V}(e) \cap V \neq \emptyset$ (T-BRANCH-high)

$((V, w),\ \texttt{branch}\ e) \xrightarrow{ACK} (V, w\bot)$ iff $\mathcal{V}(e) \cap V = \emptyset$ (T-BRANCH-low)

$((V, wa),\ \texttt{exit}) \xrightarrow{ACK} (V, w)$ (T-EXIT)

$((V, w),\ \texttt{not}\ S) \xrightarrow{ACK} (V \cup \mathit{modified}(S), w)$ iff $w \notin \{\bot\}^\star$ (T-NOT-high)

$((V, w),\ \texttt{not}\ S) \xrightarrow{ACK} (V, w)$ iff $w \in \{\bot\}^\star$ (T-NOT-low)

$((V, w),\ \textbf{skip}) \xrightarrow{OK} (V, w)$ (T-SKIP)

$((V, w),\ x := e) \xrightarrow{OK} (V \cup \{x\}, w)$ iff $w \notin \{\bot\}^\star$ or $\mathcal{V}(e) \cap V \neq \emptyset$ (T-ASSIGN-sec)

$((V, w),\ x := e) \xrightarrow{OK} (V \setminus \{x\}, w)$ iff $w \in \{\bot\}^\star$ and $\mathcal{V}(e) \cap V = \emptyset$ (T-ASSIGN-pub)

$((V, w),\ \textbf{output}\ e) \xrightarrow{OK} (V, w)$ iff $w \in \{\bot\}^\star$ and $\mathcal{V}(e) \cap V = \emptyset$ (T-PRINT-ok)

$((V, w),\ \textbf{output}\ e) \xrightarrow{\textbf{output}\ \theta} (V, w)$ iff $w \in \{\bot\}^\star$ and $\mathcal{V}(e) \cap V \neq \emptyset$ (T-PRINT-def)

$((V, w),\ \textbf{output}\ e) \xrightarrow{NO} (V, w)$ iff $w \notin \{\bot\}^\star$ (T-PRINT-no)

Figure 10: Transition function of monitoring automata of [Le Guernic et al., 2006b]

Proof. The proof goes by structural induction on C and follows directly from the rules given in Figure 5 and the definition of $\mathit{modified}(C)$.

Lemma A.15 (Existence of a Static Analysis Result).
For all command C, value store σ, and tag store ρ, there exists an analysis result $(\mathfrak{D}, \mathfrak{X})$ such that $[\![\sigma, \rho \vdash C]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$ and Hypotheses 2.3 and 2.4 hold.

Proof. The proof goes easily by structural induction on C by constructing straightforwardly $(\mathfrak{D}, \mathfrak{X})$ from the constraints. The only difficult case is for while-statements to show that there exists an adequate solution to $\mathfrak{D} = \{\!\!\{\mathfrak{D}^{\mathrm{l}} \circ (\mathfrak{D} \cup \mathcal{I}d), \mathcal{I}d\}\!\!\}^{\rho(e)}_{\sigma(e)}$.

Let $\mathcal{G}_l$, $\mathcal{G}_l^+$ and $\mathcal{G}_l^*$ be graphs whose nodes are $\mathbb{X}$. There exists an edge from x to y in $\mathcal{G}_l$ if and only if (x,y) belongs to $\mathfrak{D}^{\mathrm{l}}$. $\mathcal{G}_l^+$ is the transitive closure of $\mathcal{G}_l$; and $\mathcal{G}_l^*$ is the reflexive closure of $\mathcal{G}_l^+$.

If $\rho(e) = \bot$ and $\sigma(e) = \texttt{false}$ then $\mathcal{I}d$is a solution for $\mathfrak{D}$. It is straightforward to see that it complies with all the requirements.

If $\rho(e) = \bot$ and $\sigma(e) = \texttt{true}$ then $\mathfrak{D}$ is the set of pairs (x,y) such that there exists an edge from x to y in $\mathcal{G}_l^+$. It is easy to see that (x,y) belongs to $\mathfrak{D}^{\mathrm{l}} \circ (\mathfrak{D} \cup \mathcal{I}d)$ if and only if there exists a node z such that there exists an edge (x,z) in $\mathcal{G}_l^*$ and an edge (z,y) in $\mathcal{G}_l$. If this is the case then, as $\mathcal{G}_l^+$ is the transitive closure of $\mathcal{G}_l$, there exists an edge (x,y) in $\mathcal{G}_l^+$; and therefore, (x,y) belongs to $\mathfrak{D}$. Therefore, $\mathfrak{D}^{\mathrm{l}} \circ (\mathfrak{D} \cup \mathcal{I}d)$ is a solution for $\mathfrak{D}$. From the construction mechanism, it is obvious that $(\mathfrak{D}, \mathfrak{X})$ complies with Hypothesis 2.3. If x does not belongs to $\mathfrak{X}^{\mathrm{l}}$ then $\mathfrak{D}^{\mathrm{l}} = \{x\}$ (by induction). Therefore the sole edge in $\mathcal{G}_l$ whose origin is x has for destination x; and hence, there exists a sole edge whose origin is x in $\mathcal{G}_l^+$ (the destination of this edge being x itself). This allows to conclude that, as $\mathfrak{X} = \mathfrak{X}^{\mathrm{l}}$, the Hypothesis 2.4 holds.

If $\rho(e) = \top$ then the set of pairs (x,y) such that there exists an edge from x to y in $\mathcal{G}_l^*$ is a solution for $\mathfrak{D}$. The proof then ends similarly to the preceding case.

Lemma A.16 (No Downgrading Under Secret Context).
For all command C, value store σ and σ', and automaton state (V, w) and (V', w') such that $w \notin \{\bot\}^\star$, if $((V, w), \sigma) \vdash C \overset{o}{\Rightarrow}_{M(O)} ((V', w'), \sigma')$ then $V \subseteq V'$.

Proof. It follows directly from lemma A.10 and the only transition rules applying to "`not` C" ((T-NOT-high) and (T-NOT-low)).

Lemma A.17 (WDKL (while-dilemma killer lemma) bis).
For all command C, expression e, automaton state $q = (V, w)$, and value store σ, if

$\star_1$ $\mathcal{V}(e) \cap V \neq \emptyset$,

$\star_2$ $((V, w), \sigma) \vdash$ **while** e **do** C **done** $\overset{o}{\Rightarrow}_{M(s)} ((V_f, w), \sigma_f)$

*then "$((V, w\top), \sigma) \vdash$ **while** e **do** C **done** $\overset{o}{\Rightarrow}_{M(s)} ((V_f, w\top), \sigma_f)$".*

Proof. $\star_1$ implies that the automaton transition — Figure 10 — on `branch` e is (T-BRANCH-high). The definition of this transition in turn implies that the transition on `not` C, if it occurs, is (T-NOT-high).

Lemma A.18 (Context Sensitive Dynamic Analysis is More Precise).
Assumes that the dynamic information flow analysis uses an acceptable *— Figure 5 — static analysis for which Hypotheses 2.3 and 2.4 hold. For all command C, value stores σ and σ', automaton states (V, w) and (V', w), outputs o, tag stores ρ, and tags t^{pc}, if:*

$\star_1$ $((V, w), \sigma) \vdash C \overset{o}{\Rightarrow}_{M(O)} ((V', w), \sigma')$,

$\star_2$ $\forall x, (\rho(x) = \top) \Rightarrow x \in V$,

$\star_3$ $t^{pc} = \top \Rightarrow w \notin \{\bot\}^\star$,

then there exists a tag store ρ' such that:

- $\sigma; \rho; t^{pc} \vdash C \Downarrow \sigma' : \rho'$,
- $\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$.

Proof. The proof is done by induction on the size of the derivation tree of "$((V, w), \sigma) \vdash C \overset{o}{\Rightarrow}_{M(O)} ((V', w), \sigma')$". Assume the theorem holds for any sub-derivation tree. As we consider the language without print statements, the rules ($E_{M(O)}$-EDIT) and ($E_{M(O)}$-NO) can not be applied. If the last semantics rule used is:

($E_{M(O)}$-OK) then we can conclude that :

(1)
- $((V, w), C) \xrightarrow{OK} (V', w)$,
- $\sigma \vdash C \overset{o}{\Rightarrow}_O \sigma'$,
- C is "**skip**" or "x **:=** e".

It follows from the $M(O)$ semantics rule ($E_{M(O)}$-OK). This rule applies only to actions (skip or assignment statements).

(•) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".

Case 1: C is "**skip**".

(a) $V' = V$ and $\sigma' = \sigma$.
It follows from the case hypothesis, the only rule applying to "**skip**" in the O semantics (E_O-SKIP), and from the only transition in the automaton applying to "**skip**" (T-SKIP).

(b) $\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma : \rho$.
It follows directly from the rule ($E_{\mathcal{M}}$-SKIP).

(c) $\forall x, (\rho(x) = \top) \Rightarrow x \in V'$.
It follows directly from the global hypothesis $\star_2$ and from the local conclusion (a).

(∘) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (b), (a), and (c).

Case 2: C is "x **:=** e"; and $w \notin \{\bot\}^\star$ or $\mathcal{V}(e) \cap V \neq \emptyset$.

(a) $V' = V \cup \{x\}$ and $\sigma' = \sigma[x \mapsto \sigma(e)]$.
It follows from the case hypothesis, the only rule applying to "x **:=** e" in the O semantics (E_O-ASSIGN), and the only transition in the automaton applying to "x **:=** e" whenever $w \notin \{\bot\}^\star$ or $\mathcal{V}(e) \cap V \neq \emptyset$ (T-ASSIGN-sec).

(b) $\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma[x \mapsto \sigma(e)] : \rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)]$.
It follows directly from the rule ($E_{\mathcal{M}}$-ASSIGN) and Definition A.1.

(c) $\forall y,\ (\rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)](y) = \top) \Rightarrow y \in V'$.
From the local conclusion (a), x belongs to V'. For all variable y, different from x, $\rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)](y) = \top$ implies $\rho(y) = \top$. Hence, the global hypothesis $\star_2$ and the local conclusion (a) imply that y belongs to V'.

(∘) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (b), (a), and (c).

Case 3: C is "x **:=** e"; and $w \in \{\bot\}^\star$ and $\mathcal{V}(e) \cap V = \emptyset$.

(a) $V' = V - \{x\}$ and $\sigma' = \sigma[x \mapsto \sigma(e)]$.
It follows from the case hypothesis, the only rule applying to "x **:=** e" in the O semantics (E_O-ASSIGN), and the only transition in the automaton applying to "x **:=** e" whenever $w \in \{\bot\}^\star$ and $\mathcal{V}(e) \cap V = \emptyset$ (T-ASSIGN-sec).

(b) $\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma[x \mapsto \sigma(e)] : \rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)]$.
It follows directly from the rule ($\text{E}_{\mathcal{M}}$-ASSIGN) and Definition A.1.

(c) $t^{\text{pc}} \sqcup \rho(e) = \bot$.
From the case hypothesis and the global hypothesis $\star_3$, $t^{\text{pc}} = \bot$. From the case hypothesis and the global hypothesis $\star_2$, $\rho(e) = \bot$. Those two properties implying the desired result.

(d) $\forall y,\ (\rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)](y) = \top) \Rightarrow y \in V'$.
From the local conclusion (c), the variable x respects the desired property. For all variable y, different from x, $\rho[x \mapsto t^{\text{pc}} \sqcup \rho(e)](y) = \top$ implies $\rho(y) = \top$. Hence, the global hypothesis $\star_2$ and the local conclusion (a) imply that y belongs to V'.

(∘) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (b), (a), and (d).

($\text{E}_{M(O)}$-IF) then we can conclude that :

(1) There exist automaton states (V_1, w_1), (V_2, w_1), and (V_3, w_2) such that:

- C is "**if** e **then** C_{true} **else** C_{false} **end**",
- $\sigma(e) = v$,
- $((V, w),\ \texttt{branch}\ e) \xrightarrow{OK} (V_1, w_1)$,
- $((V_1, w_1),\ \sigma) \vdash C_v \overset{o}{\Rightarrow}_{M(O)} ((V_2, w_1),\ \sigma')$,
- $((V_2, w_1),\ \texttt{not}\ C_{\neg v}) \xrightarrow{OK} (V_3, w_2)$,
- $((V_3, w_2),\ \texttt{exit}) \xrightarrow{OK} (V', w)$.

It follows directly from the rule ($\text{E}_{M(O)}$-IF) and lemma A.13.

(2) $V_1 = V$ and $(w \notin \{\bot\}^\star \Rightarrow w_1 \notin \{\bot\}^\star)$.
It follows directly from the only two transitions applying to "`branch` e" ((T-BRANCH-high) and (T-BRANCH-low)).

(3) There exists a tag t_e such that $\sigma; \rho \vdash e \Downarrow v : t_e$.
It follows from the local conclusion (1) and Definition A.1.

(•) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".

Case 1: $t_e = \bot$.

(a) There exists a tag store ρ_v such that "$\sigma; \rho; t^{\text{pc}} \vdash C_v \Downarrow \sigma' : \rho_v$" and "$\forall x, (\rho_v(x) = \top) \Rightarrow x \in V_2$".
It follows from the inductive hypothesis, the local conclusions (1) and (2), and the global hypotheses $\star_2$ and $\star_3$.

(b) $\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho_v$.
It follows from the rule ($\text{E}_{\mathcal{M}}$-IF$_\bot$), the case hypothesis, and the local conclusion (a).

(c) $\forall x, (\rho_v(x) = \top) \Rightarrow x \in V'$.
From the local conclusion (1) and the transition rules (T-NOT-high), (T-NOT-low), and (T-EXIT), $V_2 \subseteq V'$. Hence, the desired result follow from the local conclusion (a).

(○) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (b) and (c).

Case 2: $t_e = \top$.

(a) $\mathcal{V}(e) \cap V \neq \emptyset$.
It follows from the local conclusion (3), Definition A.1, the case hypothesis, and the global hypothesis $\star_2$.

(b) $w_1 \notin \{\bot\}^\star$.
Transition rule (T-BRANCH-high) and the local conclusions (1) and (a) imply $w_1 \notin \{\bot\}^\star$.

(c) There is a tag store ρ_v such that "$\sigma; \rho; \top \vdash C_v \Downarrow \sigma' : \rho_v$" and "$\forall x, (\rho_v(x) = \top) \Rightarrow x \in V_2$".
It follows from the inductive hypothesis, the local conclusions (1), (2) and (b), and the global hypothesis $\star_2$.

(d) $V' = V_2 \cup \mathit{modified}(C_{\neg v})$.
The local conclusions (1) and (b) and the transition rules (T-NOT-high) and (T-EXIT) imply the desired result.

Let the analysis result $(\mathfrak{D}, \mathfrak{X})$ be such that $[\![\sigma, \rho \vdash C_{\neg v}]\!]^{\sharp_{\mathcal{G}}} = (\mathfrak{D}, \mathfrak{X})$, and the tag stores $\rho_{\neg v}$ and ρ_e be such that $\rho_{\neg v} = \lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)$ and $\rho_e = (\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\})$.

(e) There exists a tag store ρ_f such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho_f$" and "$\rho_f = \rho_v \sqcup \rho_{\neg v} \sqcup \rho_e$".
It follows from the rule ($\text{E}_{\mathcal{M}}$-IF$_\top$), the case hypothesis, the local conclusions (3) and (c), and the lemma A.15.

(f) $\forall x, (\rho_f(x) = \top) \Rightarrow x \in V'$.
For all variable x such that $\rho_v(x) = \top$, the local conclusions (c) and (d) imply that x belongs to V'. For all variable x such that $\rho_e(x) = \top$, lemma A.14 implies that x belongs to $\mathit{modified}(C_{\neg v})$; and hence, the local conclusion (d) implies that x belongs to V'. For all variable x such that $\rho_{\neg v}(x) = \top$ and $\mathfrak{D}(x) \neq \{x\}$, Hypothesis 2.4 implies that x belongs to $\mathfrak{X}$; and hence, from what has been said before, x belongs to V'. For all variable x such that $\rho_{\neg v}(x) = \top$ and $\mathfrak{D}(x) = \{x\}$, from the definition of $\rho_{\neg v}$, $\rho(x) = \top$; which, combined with the global hypothesis $\star_2$, the local conclusions (2), (b), and (1), lemma A.16, and local conclusion (d), implies that x belongs to V'.

(∘) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (e) and (f).

($\mathbf{E}_{M(O)}$-WHILE$_{true}$) then we can conclude that :

(1) There exist a value store σ_1 and automaton states (V, w_1), (V_1, w_1), and (V_1, w) such that:

- C is "**while** e **do** C_l **done**",
- $\sigma(e) = \texttt{true}$,
- $((V, w),\ \texttt{branch}\ e) \xrightarrow{OK} (V, w_1)$,
- $((V, w_1),\ \sigma) \vdash C_l \overset{o_l}{\Longrightarrow}_{M(O)} ((V_1, w_1),\ \sigma_1)$,
- $((V_1, w_1),\ \texttt{exit}) \xrightarrow{OK} (V_1, w)$,
- $((V_1, w_1),\ \sigma_1) \vdash$ **while** e **do** C_l **done** $\overset{o_w}{\Longrightarrow}_{M(O)} ((V', w),\ \sigma')$.

It follows directly from the rule ($\text{E}_{M(O)}$-WHILE$_{true}$), the transitions of the monitoring automaton, and lemma A.13.

(2) There exists a tag t_e such that $\sigma; \rho \vdash e \Downarrow \texttt{true} : t_e$.
It follows from the local conclusion (1) and Definition A.1.

(3) $(t_e = \top) \Rightarrow \mathcal{V}(e) \cap V \neq \emptyset$.
It follows from the local conclusion (2), Definition A.1, and the global hypothesis $\star_2$.

(4) There exists a tag store ρ_1 such that "$\sigma; \rho; t^{\text{pc}} \sqcup t_e \vdash C_l \Downarrow \sigma_1 : \rho_1$" and "$\forall x, (\rho_1(x) = \top) \Rightarrow x \in V_1$".
From the local conclusion (1) and the transitions (T-BRANCH-high) and (T-BRANCH-low), there exists $a \in \{\top, \bot\}$ such that $w_1 = wa$. Thus, from the global hypothesis $\star_3$, $(t^{\text{pc}} = \top) \Rightarrow w_1 \notin \{\bot\}^\star$. From the transition (T-BRANCH-high) and the local conclusions (3) and (1), $(t_e = \top) \Rightarrow w_1 \notin \{\bot\}^\star$. Hence, $(t^{\text{pc}} \sqcup t_e = \top) \Rightarrow w_1 \notin \{\bot\}^\star$. Then, using the inductive hypothesis, the local conclusion (1) and the global hypothesis $\star_2$ imply the desired result.

(•) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".

Case 1: $t_e = \bot$.

(a) There is ρ_2 such that "$\sigma_1; \rho_1; t^{\text{pc}} \vdash$ **while** e **do** C_l **done** $\Downarrow \sigma' : \rho_2$" and "$\forall x, (\rho_2(x) = \top) \Rightarrow x \in V'$".
This result is implied by the inductive hypothesis, the local conclusions (1) and (4), and the global hypothesis $\star_3$.

(b) $\sigma; \rho; t^{\text{pc}} \vdash C_l$; **while** e **do** C_l **done** $\Downarrow \sigma' : \rho_2$.
It follows directly from the case hypothesis, the semantics rule ($\text{E}_{\mathcal{M}}$-SEQUENCE) and the local conclusions (4) and (a).

(∘) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the semantics rule ($\text{E}_{\mathcal{M}}$-WHILE$_{true_\bot}$), the case hypothesis, and the local conclusions (2), (b), and (a).

Case 2: $t_e = \top$.

(a) $((V_1, w\top),\ \sigma_1) \vdash \textbf{while}\ e\ \textbf{do}\ C_l\ \textbf{done} \overset{o_w}{\Longrightarrow}_{M(O)} ((V', w\top),\ \sigma')$.
It follows directly from the local conclusion (1) and lemma A.17.

(b) There is ρ_2 such that "$\sigma_1; \rho_1; t^{\text{pc}} \sqcup t_e \vdash \textbf{while}\ e\ \textbf{do}\ C_l\ \textbf{done} \Downarrow \sigma' : \rho_2$" and "$\forall x, (\rho_2(x) = \top) \Rightarrow x \in V'$".
This result is implied by the inductive hypothesis, the local conclusions (a) and (4), and the fact that $w\top$ does not belong to $\{\bot\}^\star$.

(c) $\sigma; \rho; t^{\text{pc}} \sqcup t_e \vdash C_l\ ;\ \textbf{while}\ e\ \textbf{do}\ C_l\ \textbf{done} \Downarrow \sigma' : \rho_2$.
It follows directly from the case hypothesis, the semantics rule ($\text{E}_{\mathcal{M}}$-SEQUENCE) and the local conclusions (4) and (b).

(d) $\forall x, (\rho(x) = \top) \Rightarrow x \in V'$.
From the case hypothesis, the local conclusions (3) and (1), and the transition (T-BRANCH-high), $w_1 \notin \{\bot\}^\star$. Hence, from lemma A.16, $V \subseteq V_1$. From lemma A.12, the case hypothesis, and the local conclusions (3) and (1), $V_1 \subseteq V'$. Thus, the desired result follows from the global hypothesis $\star_2$.

(∘) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the semantics rule ($\text{E}_{\mathcal{M}}$-WHILE$_{true_\top}$), the case hypothesis, and the local conclusions (2), (c), (d) and (b).

($\text{E}_{M(O)}$-WHILE$_{false}$) then we can conclude that :

(1) There exist automaton states (V, w_1) and (V', w_1) such that:

- C is "**while** e **do** C_l **done**",
- $\sigma(e) = \texttt{false}$,
- $((V, w),\ \texttt{branch}\ e) \xrightarrow{OK} (V, w_1)$,
- $((V, w_1),\ \texttt{not}\ C_l) \xrightarrow{OK} (V', w_1)$,
- $((V', w_1),\ \texttt{exit}) \xrightarrow{OK} (V', w)$,
- $\sigma' = \sigma$.

It follows directly from the rule ($\text{E}_{M(O)}$-WHILE$_{false}$) and the transitions of the monitoring automaton.

(2) There exists a tag t_e such that $\sigma; \rho \vdash e \Downarrow \texttt{false} : t_e$.
It follows from the local conclusion (1) and Definition A.1.

(•) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".

Case 1: $t_e = \bot$.

(a) $\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma : \rho$.
It follows from the rule ($\text{E}_{\mathcal{M}}$-WHILE$_{skip}$), the case hypothesis, and the local conclusion (2).

(b) $\forall x, (\rho(x) = \top) \Rightarrow x \in V'$.
From the local conclusion (1) and the transition rules (T-NOT-high) and (T-NOT-low), $V \subseteq V'$. Hence, the desired result follows from the global hypothesis $\star_2$.

(∘) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (a), (1) and (b).

Case 2: $t_e = \top$.

(a) $\mathcal{V}(e) \cap V \neq \emptyset$.
It follows from the local conclusion (2), Definition A.1, the case hypothesis, and the global hypothesis $\star_2$.

(b) $V' = V \cup \mathit{modified}(C_l)$.
Transition rule (T-BRANCH-high) and the local conclusions (1) and (a) imply $w_1 \notin \{\bot\}^\star$. Hence, the local conclusion (1) and transition (T-NOT-high) imply the desired result.

Let the analysis result $(\mathfrak{D}, \mathfrak{X})$ be such that $[\![\sigma, \rho \vdash C_l \ ; \ \textbf{while}\ e\ \textbf{do}\ C_l\ \textbf{done}]\!]^{\sharp_\mathcal{G}} = (\mathfrak{D}, \mathfrak{X})$, and the tag stores ρ_l and ρ_e be such that $\rho_l = \lambda x. \bigsqcup_{y \in \mathfrak{D}(x)} \rho(y)$ and $\rho_e = (\mathfrak{X} \times \{\top\}) \cup (\mathfrak{X}^c \times \{\bot\})$.

(c) There exists a tag store ρ_f such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho_f$" and "$\rho_f = \rho \sqcup \rho_l \sqcup \rho_e$".
It follows from the rule ($\text{E}_\mathcal{M}$-WHILE$_{false_\top}$), the case hypothesis, the local conclusions (2) and (1), and the Lemma A.15.

(d) $\forall x, (\rho_f(x) = \top) \Rightarrow x \in V'$.
For all variable x such that $\rho(x) = \top$, the global hypothesis $\star_2$ and the local conclusion (b) imply that x belongs to V'. For all variable x such that $\rho_e(x) = \top$, lemma A.14 implies that x belongs to $\mathit{modified}(C_l)$; and hence, the local conclusion (b) implies that x belongs to V'. For all variable x such that $\rho_l(x) = \top$ and $\mathfrak{D}(x) \neq \{x\}$, Hypothesis 2.4 implies that x belongs to $\mathfrak{X}$; and hence, from what has been said before, x belongs to V'. For all variable x such that $\rho_l(x) = \top$ and $\mathfrak{D}(x) = \{x\}$, from the definition of ρ_l, $\rho(x) = \top$; which, combined with the global hypothesis $\star_2$ and the local conclusion (b), implies that x belongs to V'.

(∘) There is a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the local conclusions (c) and (d).

($\text{E}_{M(O)}$-SEQ) then we can conclude that :

(1) There exists a value store σ_1 and an automaton state (V_1, w) such that:

- C is "$C_1 \ ; \ C_2$",
- $((V, w), \sigma) \vdash C_1 \overset{o_1}{\Longrightarrow}_{M(O)} ((V_1, w_1), \sigma_1)$,
- $((V_1, w), \sigma_1) \vdash C_2 \overset{o_2}{\Longrightarrow}_{M(O)} ((V', w), \sigma')$.

It follows directly from the rule ($\text{E}_{M(O)}$-SEQ) and lemma A.13.

(2) There exists a tag store ρ_1 such that "$\sigma; \rho; t^{\text{pc}} \vdash C_1 \Downarrow \sigma_1 : \rho_1$" and "$\forall x, (\rho_1(x) = \top) \Rightarrow x \in V_1$".
It follows from the inductive hypothesis, the global hypotheses $\star_2$ and $\star_3$, and the local conclusion (1).

(3) There exists a tag store ρ' such that "$\sigma_1; \rho_1; t^{\text{pc}} \vdash C_2 \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".
It follows from the inductive hypothesis, the global hypothesis $\star_3$, and the local conclusions (1) and (2).

(•) There exists a tag store ρ' such that "$\sigma; \rho; t^{\text{pc}} \vdash C \Downarrow \sigma' : \rho'$" and "$\forall x, (\rho'(x) = \top) \Rightarrow x \in V'$".

It follows from the rule ($E_{\mathcal{M}}$-SEQUENCE) and the local conclusions (1), (2), and (3).

References

Ellis S. Cohen. Information transmission in computational systems. *ACM SIGOPS Operating Systems Review*, 11(5):133–139, 1977. 7

Gurvan Le Guernic. Automaton-based Confidentiality Monitoring of Concurrent Programs. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium (CSFS20)*. IEEE Computer Society, July 6–8 2007a. ISBN 0-7695-2819-8. 4, 7

Gurvan Le Guernic. Automaton-based Non-interference Monitoring of Concurrent Programs. Technical Report 2007-1, Department of Computing and Information Sciences, College of Engineering, Kansas State University, 234 Nichols Hall, Manhattan, KS 66506, USA, February 2007b. URL http://www.cis.ksu.edu/~schmidt/techreport/2007.list.html. 14

Gurvan Le Guernic and Thomas Jensen. Monitoring Information Flow. In Andrei Sabelfeld, editor, *Proceedings of the Workshop on Foundations of Computer Security*, pages 19–30. DePaul University, June 2005. 8, 19

Gurvan Le Guernic, Anindya Banerjee, Thomas Jensen, and David Schmidt. Automata-based Confidentiality Monitoring. In *Proceedings of the Annual Asian Computing Science Conference*, Lecture Notes in Computer Science, December 6–8 2006a. To appear. 2, 4, 5, 7, 18, 19, 20, 21

Gurvan Le Guernic, Anindya Banerjee, and David Schmidt. Automaton-based Non-interference Monitoring. Technical Report 2006-1, Department of Computing and Information Sciences, College of Engineering, Kansas State University, 234 Nichols Hall, Manhattan, KS 66506, USA, April 2006b. URL http://www.cis.ksu.edu/~schmidt/techreport/2006.list.html. 14, 51, 52

D. Volpano, G. Smith, and C. Irvine. A sound type system for secure flow analysis. *J. Computer Security*, 4(3):167–187, 1996. 21