# Information Flow for Algol-like Languages

David Clark[1], Chris Hankin[2⋆], and Sebastian Hunt[3]

[1] Department of Computer Science, King's College, London WC2R 2LS
e-mail: david@dcs.kcl.ac.uk
[2] Department of Computing, Imperial College, London SW7 2BZ
e-mail: clh@doc.ic.ac.uk
[3] Department of Computing, City University, London EC1V 0HB
e-mail: seb@soi.city.ac.uk



| **SecSafe Document** | |
|---|---|
| **Number:** | SECSAFE-ICSTM-002 |
| **Version:** | 1.0 |
| **Classification:** | Public |
| **Status:** | Submitted |



**Abstract.** In this paper we present an approach to information flow analysis for a family of languages. We start with a simple imperative language. We present an information flow analysis using a *flow logic*. The paper contains detailed correctness proofs for this analysis. We next extend the analysis to Idealised Algol, a higher-order extension of the simple imperative language. The paper concludes with a discussion of further extensions, including a probabilistic extension of Idealised Algol.

## 1 Introduction

We are concerned here with Trojan horse confidentiality attacks. Suppose that $H$ and $L$ are two sets of variable names. $H$ is the set of high-security inputs, which we wish to keep secret from the attacker (who supplies the program), and $L$ is the set of low-security outputs, which all users (including the attacker) can read when the program terminates. We need to verify that the final values of variables in $L$ give no information about the initial values of variables in $H$.

The particular security property we are concerned with is a simple extensional non-interference property (our programs do not perform IO while executing, so observations of intermediate stores are not possible).

Following Denning [2] we divide information flows into two classes: *direct* and *indirect*. Indirect flows are just the transitive flows (a flow from $x$ to $y$ followed by a flow from $y$ to $z$ implies a flow from $x$ to $z$). The direct flows are further divided:

1. *Explicit* flows arise from assignments and function application; for example, $x := y + z$ causes explicit information flows from both $y$ and $z$ to $x$. Similarly, $(\lambda x.t)y$ causes an explicit flow from $y$ to the lambda-bound variable $x$.

⋆ This author partially funded by the EU FET Open project SecSafe.

2. *Implicit* flows arise when one variable's value influences the value assigned to another by determining the flow of control. There are two types of implicit flow (though Denning only considered the first type):
   (a) *Local flows* arise from guards in conditionals:

   $$\texttt{if}\quad x\quad \texttt{then}\quad y := z\quad \texttt{else}\quad y := w.$$

   Here there is local implicit information flow from $x$ to $y$, in addition to the explicit flows from $z$ and $w$.
   (b) *Global flows* arise from guards in while loops

   $$x := y; (\texttt{while}\quad w\quad \texttt{do}\quad x := z); \cdots.$$

   Here there is a global implicit flow from $w$ to all subsequent program points, since reaching those points carries the information that the loop terminated and hence, in this example, that $w$ was **true**.

The paper is essentially in two parts. The first part concerns the While language (Sections 2 to 4), the second part concerns Idealised Algol (Sections 5 and 6) and we conclude in Section 7. Section 2 presents the While language and a flow logic specification of information flow analysis. The following section gives a detailed treatment of the correctness of the analysis and Section 4 presents an example. Section 5 introduces Idealised Algol and extends the analysis of Section 2; the extension is non-trivial because aliasing issues arise and the control flow is no longer explicit from the program. This latter observation motivates a two phase analysis which involves a control flow analysis as the first phase. Section 6 presents a sketch of the correctness of the analysis; we concentrate on the correctness of the abstraction, the well-formedness and the existence of solutions results follow in a similar way to the earlier development.

## 2 Information Flow for While

The syntax of the language is as follows:

$$\begin{aligned} S &\in \textbf{Statement}\\ C &\in \textbf{Command}\\ \ell &\in \textbf{Lab}\\ x &\in \textbf{Ide}\\ a &\in \textbf{Arith-exp}\\ b &\in \textbf{Bool-exp}\end{aligned}$$

$$\begin{aligned} S &::= C^{\ell}\\ C &::= \texttt{skip} \;\mid\; x := a \;\mid\; S_1; S_2 \;\mid\\ &\qquad \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \;\mid\; \texttt{while}\ b\ \texttt{do}\ S \;\mid\\ &\qquad \texttt{new}\ x.\ S\end{aligned}$$

This is the standard While language, used in many programming language textbooks, extended with a block structuring construct (`new`); the reader should see [3] for a detailed treatment of program analysis techniques for this language. In contrast to the latter, a statement is a labelled command – in *op cit.* only the primitive statements are labelled. This difference arises because we want to analyse structure in programs whereas the analyses of *op cit.* analyse flowcharts. We make no assumption about the uniqueness of labels or variables (bound by `new`) but the results of the analysis will be more accurate if these constraints are enforced.

The semantics of the language is shown in table 1. This is a big step semantics; configurations consist of a statement and a state, $\sigma$, which maps **Ide** to integers. We use 0 to represent the semantics of false and 1 for true. The statement `void` is the null, or empty, statement.

We do not specify the syntax or semantics of arithmetic and boolean expressions but we assume that they are *total* and *purely functional*: for all expressions $e$, stores $\sigma$, there exists some $n$ such that $e, \sigma \Downarrow n, \sigma$. Totality is necessary for the proof of Termination Independence (theorem 3).

$$\frac{}{\texttt{skip}^\ell, \sigma \Downarrow \texttt{void}, \sigma}$$

$$\frac{a, \sigma \Downarrow v, \sigma}{(x := a)^\ell, \sigma \Downarrow \texttt{void}, \sigma[x \mapsto v]}$$

$$\frac{S_1, \sigma \Downarrow \texttt{void}, \sigma' \quad S_2, \sigma' \Downarrow \texttt{void}, \sigma''}{(S_1; S_2)^\ell, \sigma \Downarrow \texttt{void}, \sigma''}$$

$$\frac{b, \sigma \Downarrow 1, \sigma \quad S_1, \sigma \Downarrow \texttt{void}, \sigma'}{(\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2)^\ell, \sigma \Downarrow \texttt{void}, \sigma'}$$

$$\frac{b, \sigma \Downarrow 0, \sigma \quad S_2, \sigma \Downarrow \texttt{void}, \sigma'}{(\texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2)^\ell, \sigma \Downarrow \texttt{void}, \sigma'}$$

$$\frac{b, \sigma \Downarrow 0, \sigma}{(\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma \Downarrow \texttt{void}, \sigma}$$

$$\frac{b, \sigma \Downarrow 1, \sigma \quad S, \sigma \Downarrow \texttt{void}, \sigma' \quad (\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma' \Downarrow \texttt{void}, \sigma''}{(\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma \Downarrow \texttt{void}, \sigma''}$$

$$\frac{S, \sigma[x \mapsto 0] \Downarrow \texttt{void}, \sigma'}{(\texttt{new}\ x.\ S)^\ell, \sigma \Downarrow \texttt{void}, \sigma'[x \mapsto \sigma x]}$$

**Table 1.** Natural Semantics for While

The Information Flow Analysis is presented in the style of a *flow logic* as pioneered by Flemming Nielson and Hanne Riis Nielson and their group; see [3] for an introduction to this style. For this language the control flow is explicit and we just use the flow logic to specify information flow; in the next section this will change. We contrast our approach with others in the conclusions. We assume that **Ide** and **Lab** are restricted to those variables and labels appearing in the program. We write

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models S$$

when $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}})$ is an acceptable Information Flow Analysis of the statement $S$.

We use the following functions on labels:

$$\widehat{\mathsf{X}} \in \mathbf{Assign} = \mathbf{Lab} \to \mathcal{P}(\mathbf{Ide})$$

Given a label, $\widehat{\mathsf{X}}$ returns a superset of those variables which may be assigned to during evaluation of the statement with that label. It is necessary to collect this information in order to handle implicit flows correctly.

$$\widehat{\mathsf{G}} \in \mathbf{Global} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Ide}})$$

where $\widehat{\mathbf{Ide}} = \mathbf{Ide} \cup \{\bullet\}$. Given a label, $\widehat{\mathsf{G}}$ returns a superset of those variables whose values may affect termination of the statement with that label. The special variable $\bullet$ is used to indicate that a while loop has been encountered (and thus non-termination is possible). The variables in $\widehat{\mathsf{G}}(\ell)$ are involved in implicit global flows to any variables assigned to after the execution of the statement at $\ell$.

$$\widehat{\mathsf{D}} \in \mathbf{Dep} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Ide}} \times \widehat{\mathbf{Ide}})$$

Given a label, $\widehat{\mathsf{D}}$ returns a superset of the dependencies for the statement with that label; a pair $(x, y)$ is a dependency for $S$ if different values for $y$ in the state prior to execution of $S$ can result in different values for $x$ in the state after execution of $S$.

We also use $\mathrm{FV}(a)$ to denote the set of variables occurring in $a$ (and similarly for $b$).

Note that $D(\ell)$ is a binary relation on $\widehat{\mathbf{Ide}}$. We denote the identity relation on $\widehat{\mathbf{Ide}}$ by Id.

We use $\,;\,$ for relational composition, thus: $x\ R\,;S\ z$ iff $\exists y.\ x\ R\ y\ S\ z$. We also overload this notation to allow the 'composition' of a set with a relation, thus: $Y\,;R \stackrel{\text{def}}{=} \{z \mid \exists y \in Y.\ y\ R\ z\}$.

We use the notation $f\backslash x$ to restrict the range of a partial function, thus: $(f\backslash x)(y)$ is undefined if $x = y$ and is $f(y)$ otherwise. We apply the same notation to binary relations: $R\backslash x \stackrel{\text{def}}{=} \{(y, z) \in R \mid y \neq x\}$.

Where convenient, we treat $D(\ell)$ as a function of type $\widehat{\mathbf{Ide}} \to \mathcal{P}(\widehat{\mathbf{Ide}})$. In particular, we use a 'function update' notation on relations thus: $R[x \mapsto Y] \stackrel{\text{def}}{=} R\backslash x \cup \{x\} \times Y$.

Throughout this paper we assume that sets, for example **Ide**, are restricted to those elements, for example identifiers, that appear in the program. This ensures that the constraints have finite solutions.

The Information Flow Analysis is shown in table 2.

If a variable is not assigned at a label, its post-execution value will be the same as its pre-execution value; hence the clause for `skip`. When an assignment is executed, the variable on the left hand side is assigned to and there are direct information flows from the variables on the right hand side to the variable on the left hand side. In a sequential composition, the variables assigned are those assigned in the sub-statements; the variables that might affect termination are those that might affect the termination of the first sub-statement or those that might affect termination of the second sub-statement after taking account of dependencies created by the first sub-statement (hence the relational composition); the dependencies are generated by the relational composition of the dependencies for the sub-statements. The conditional clause mainly concerns propagation of information from the then and else branches; if either branch contains a while loop, the variables appearing in the predicate are added to $\widehat{\mathsf{G}}(\ell)$ – this ensures that there is an implicit global flow to statements following the conditional; the variables assigned in the branches are forced to depend on the variables in the predicate – this establishes the implicit local information flows. In the clause for the while loop, the first component of $\widehat{\mathsf{G}}(\ell)$ records the fact that a while loop has been encountered and that its termination may depend on the free variables of its condition; all the remaining parts of this clause are derived from those for sequential composition and conditionals by a single unwinding of the loop. Finally the rule for `new` just propagates information taking account of the scoping.

## 3 Correctness of the analysis for While

We are concerned with three aspects of correctness:

- First, that the analysis is well-defined.
- Second, that the analysis results are a proper abstraction of the semantics.
- Third, that every program has an acceptable information flow analysis and that the constraints have solutions.

Having analysed a program, $C^\ell$, we determine that there is a breach of security if either

- $H \cap \widehat{\mathsf{G}}(\ell) \neq \emptyset$, or
- $\exists x \in L. \exists y \in H. x \widehat{\mathsf{D}} y$

### 3.1 Well-definedness

The specification of the analysis is essentially defining the relation:

$$\models \;\; : (\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\}$$

$$
\begin{aligned}
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models \texttt{skip}^{\ell} \text{ iff } \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}\\[1ex]
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models (x := a)^{\ell}\\
&\qquad \text{iff } \widehat{\mathsf{X}}(\ell) \supseteq \{x\} \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}[x \mapsto \mathrm{FV}(a)]\\[1ex]
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models (C_1^{\ell_1}; C_2^{\ell_2})^{\ell}\\
&\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C_1^{\ell_1} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C_2^{\ell_2} \wedge\\
&\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge\\
&\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1) \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1)\\[1ex]
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models (\texttt{if } b \texttt{ then } C_1^{\ell_1} \texttt{ else } C_2^{\ell_2})^{\ell}\\
&\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C_1^{\ell_1} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C_2^{\ell_2} \wedge\\
&\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge\\
&\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2) \wedge\\
&\qquad\quad (\bullet \in \widehat{\mathsf{G}}(\ell) \Rightarrow \widehat{\mathsf{G}}(\ell) \supseteq \mathrm{FV}(b)) \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_1) \cup \widehat{\mathsf{D}}(\ell_2) \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \mathrm{FV}(b)\\[1ex]
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models (\texttt{while } b \texttt{ do } C^{\ell_1})^{\ell}\\
&\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C^{\ell_1} \wedge\\
&\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \wedge\\
&\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \{\bullet\} \cup \mathrm{FV}(b) \cup \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1) \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id} \cup \widehat{\mathsf{D}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1) \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \mathrm{FV}(b)\\[1ex]
&(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models (\texttt{new } x.\ C^{\ell_1})^{\ell}\\
&\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}}) \models C^{\ell_1} \wedge\\
&\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \backslash \{x\} \wedge\\
&\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \backslash \{x\} \wedge\\
&\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_1) \backslash \{x\} \cup \{(x,x)\}
\end{aligned}
$$

**Table 2.** Information Flow Analysis for While

and it is that we want to show is well-defined. Following [3] we view table 2 as defining a functional:

$$\begin{aligned}\mathcal{Q} : ((\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\}) \\ \to ((\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\})\end{aligned}$$

and take $\models$ to be a fixed point of this functional. For well-formedness we have to show that $\mathcal{Q}$ has fixed points. Defining the following ordering:

$$Q_1 \sqsubseteq Q_2 \Leftrightarrow \forall(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) : \ (Q_1(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) = \mathtt{true}) \Rightarrow (Q_2(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, S) = \mathtt{true})$$

it is easy to verify, by inspection of table 2, that $\mathcal{Q}$ is a monotone function on the complete lattice:

$$((\mathbf{Assign} \times \mathbf{Global} \times \mathbf{Dep} \times \mathbf{Statement}) \to \{\mathtt{true}, \mathtt{false}\}, \sqsubseteq)$$

This guarantees that $\mathcal{Q}$ has fixed points. In fact, since the rules are structural, there is a unique fixed point and we take this to be the relation $\models$.

### 3.2 Proper abstraction

Given a set of variables $X$, we write $\sigma_1 \sim_X \sigma_2$ to mean that the two stores agree on all $x \in X$: $\sigma_1 \sim_X \sigma_2 \Leftrightarrow \forall x \in X.\ \sigma_1(x) = \sigma_2(x)$. Clearly, $\sim_X$ is an equivalence relation for any choice of $X$. We sometimes write $\sim_x$ to mean $\sim_{\{x\}}$.

First we consider the simplest part of the analysis, $\widehat{\mathsf{X}}$, the semantic significance of which is captured by the following theorem. $\widehat{\mathsf{X}}$ simply calculates an over approximation to the set of variables which may be assigned to when a command executes. This implies a relationship with $\widehat{\mathsf{D}}$, since the final value of any variable *not* assigned to must equal, and is therefore dependent on, its initial value: this is part 2 of the theorem.

**Theorem 1 (Assignment Freedom (While)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$ and let $X' = \{x \in \mathbf{Ide} \mid x \notin \widehat{\mathsf{X}}(\ell)\}$. Then:

1. if $C^\ell, \sigma \Downarrow \mathtt{void}, \sigma'$ then $\sigma' \sim_{X'} \sigma$
2. if $x \notin \widehat{\mathsf{X}}(\ell)$ then $x\ \widehat{\mathsf{D}}(\ell)\ x$

*Proof.* Part 1 by induction on the height of the derivation. Part 2 by structural induction. Both parts straightforward. □

Now we concentrate on the correctness of the dependency information.

**Theorem 2 (Store Independence (While)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$, then, for all $x$:

$$\begin{array}{l} \text{if } (\sigma_1 \sim_{D_x} \sigma_2 \\ \text{then} \\ \quad \text{if } (C^\ell, \sigma_1 \Downarrow \mathtt{void}, \sigma_1' \wedge C^\ell, \sigma_2 \Downarrow \mathtt{void}, \sigma_2') \\ \quad \text{then } \sigma_1' \sim_x \sigma_2' \end{array}$$

where $D_x = \widehat{\mathsf{D}}(\ell)(x)$.

*Proof.* Assume $\forall y.\ (x\ \widehat{\mathsf{D}}(\ell)\ y) \Rightarrow \sigma_1(y) = \sigma_2(y)$. Proof is by induction on the height of the first derivation (the choice is arbitrary). We proceed by cases on the form of $C$.

`skip`**:** Trivial, since $\widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}$.

$x := a$**:** Trivial, since $\widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}[x \mapsto \mathrm{FV}(a)]$.

$S_1; S_2$**:** Note that, since $\widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1)$, our initial assumption implies $\sigma_1(z) = \sigma_2(z)$ whenever $x\ \widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1)\ z$. Now suppose, by way of contradiction, that for some $x$:

$$S_1; S_2, \sigma_1 \Downarrow \mathtt{void}, \sigma_1'' \wedge S_1; S_2, \sigma_2 \Downarrow \mathtt{void}, \sigma_2'' \wedge \\ \sigma_1''(x) \neq \sigma_2''(x)$$

Now let $S_1 \equiv C_1^{\ell_1}$ and $S_2 \equiv C_2^{\ell_2}$. Then we have $\sigma_1', \sigma_2'$ such that

$$C_1^{\ell_1}, \sigma_i \Downarrow \mathtt{void}, \sigma_i' \text{ and } C_2^{\ell_2}, \sigma_i' \Downarrow \mathtt{void}, \sigma_i''$$

for $i = 1, 2$. By IH (derivation for $S_2, \sigma_1'$), for some $y$:

$$x\widehat{\mathsf{D}}(\ell_2)y \ \wedge\ \sigma_1'(y) \neq \sigma_2'(y)$$

then by IH (derivation for $S_1, \sigma_1$), for some $z$:

$$y\widehat{\mathsf{D}}(\ell_1)z \ \wedge\ \sigma_1(z) \neq \sigma_2(z)$$

Hence $x\widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1)z$ but $\sigma_1(z) \neq \sigma_2(z)$. Contradiction!

`if` $b$ `then` $S_1$ `else` $S_2$**:** If $x \notin \widehat{\mathsf{X}}(\ell)$ the result follows by Assignment Freedom (theorem 1). If $x \in \widehat{\mathsf{X}}(\ell)$, then $x\ \widehat{\mathsf{D}}(\ell)\ y$ for all $y \in \mathrm{FV}(b)$, hence the values of $b$ in $\sigma_1$ and $\sigma_2$ are equal: the result follows by IH (twice, derivations for $S_1, \sigma_1$ and $S_2, \sigma_1$).

`while` $b$ `do` $S$**:** Let $S$ be $C_1^{\ell_1}$. If $x \notin \widehat{\mathsf{X}}(\ell)$ the result follows by Assignment Freedom. If $x \in \widehat{\mathsf{X}}(\ell)$, then $x\ \widehat{\mathsf{D}}(\ell)\ y$ for all $y \in \mathrm{FV}(b)$, hence the values of $b$ in $\sigma_1$ and $\sigma_2$ are equal. When $b$ evaluates to 0 the result follows because the stores are unchanged and $\widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}$. When $b$ evaluates to 1, the result follows by IH twice: the argument uses the fact that $\widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1)$ and is essentially identical to that for the case of sequential composition.

`new` $x.\ S$**:** Straightforward from the IH.

□

Finally, we consider $\widehat{\mathsf{G}}$. The purpose of $\widehat{\mathsf{G}}$ is to collect variables which contribute to implicit global flows; these are essentially the variables that appear in the predicates of while loops and conditionals which contain nested while loops. The semantic significance of these variables is that their values may determine whether or not a program fragment terminates, as captured by the following theorem.

**Theorem 3 (Termination Independence (While)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^\ell$. Then:

1. if $\bullet \notin \widehat{\mathsf{G}}(\ell)$ then $C^\ell, \sigma \Downarrow$ for all $\sigma$.
2. if $\sigma_1 \sim_{\widehat{\mathsf{G}}(\ell)} \sigma_2$ then $C^\ell, \sigma_1 \Downarrow \;\Leftrightarrow C^\ell, \sigma_2 \Downarrow$

*Proof.* Part 1 by straightforward structural induction.

For part 2, assume $\forall y \in \widehat{\mathsf{G}}(\ell).\ \sigma_1(y) = \sigma_2(y)$. By symmetry it suffices to show $C^\ell, \sigma_1 \Downarrow \;\Rightarrow C^\ell, \sigma_2 \Downarrow$ and this is done by induction on the height of the derivation for $C^\ell, \sigma_1$. We show here just the two most informative cases for $C$.

`if` $b$ `then` $S_1$ `else` $S_2$**:** If $\bullet \notin \widehat{\mathsf{G}}(\ell)$ the result is immediate by part 1. If $\bullet \in \widehat{\mathsf{G}}(\ell)$ then $\widehat{\mathsf{G}}(\ell) \supseteq \mathrm{FV}(b)$, so $b$ evaluates to the same value in both stores: the result follows by IH twice (derivations for $S_1, \sigma_1$ and $S_2, \sigma_1$).

`while` $b$ `do` $S$**:** Let $S \equiv C_1^{\ell_1}$. Since $\widehat{\mathsf{G}}(\ell) \supseteq \mathrm{FV}(b)$, $b$ evaluates to the same value in both stores. When $b$ evaluates to 0 the result is immediate. When $b$ evaluates to 1, we have $S, \sigma_1 \Downarrow \sigma_1'$ and $(\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma_1' \Downarrow \sigma_1''$. IH (derivation for $S$) gives us $S, \sigma_2 \Downarrow \sigma_2'$. Suppose, towards a contradiction, that $(\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma_2' \not\Downarrow$. By IH (derivation for $(\texttt{while}\ b\ \texttt{do}\ S)^\ell, \sigma_1'$) there must be some $x \in \widehat{\mathsf{G}}(\ell)$ such that $\sigma_1'(x) \neq \sigma_2'(x)$. Then, by Store Independence (theorem 2), there must be some $y$ such that $x\ \widehat{\mathsf{D}}(\ell_1)\ y \wedge \sigma_1(y) \neq \sigma_2(y)$. But then we have $x \in \widehat{\mathsf{G}}(\ell)$ and $x\ \widehat{\mathsf{D}}(\ell_1)\ y$, hence $y \in \widehat{\mathsf{G}}(\ell)\,;\widehat{\mathsf{D}}(\ell_1) \subseteq \widehat{\mathsf{G}}(\ell)$, hence (by initial assumption) $\sigma_1(y) = \sigma_2(y)$. Contradiction! □

### 3.3 Existence of solutions

Finally, again following the approach of [3], we show that for all statements, $S$, the set:

$$\{(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \mid (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models S\}$$

is a Moore family. Recall that a subset $Y$ of a complete lattice $L = (L, \sqsubseteq)$ is a Moore family if and only if $\sqcap Y' \in Y$ for all $Y' \subseteq Y$. An immediate corollary of our result is that there is always an acceptable information flow analysis for a statement and that, moreover, there is a least analysis.

We define the following top elements for the components of the analysis:

$$\begin{aligned}
\widehat{\mathsf{X}}_\top &= \lambda x.\mathbf{Ide} \\
\widehat{\mathsf{G}}_\top &= \lambda x.\widehat{\mathbf{Ide}} \\
\widehat{\mathsf{D}}_\top &= \lambda x.\widehat{\mathbf{Ide}} \times \widehat{\mathbf{Ide}}
\end{aligned}$$

**Theorem 4.** For all $S \in$ **Statement** the set $\{(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \mid (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models S\}$ is a Moore family.

*Proof.* First, we show

$$(\widehat{\mathsf{X}}_\top, \widehat{\mathsf{G}}_\top, \widehat{\mathsf{D}}_\top) \models C^\ell \qquad (*)$$

which follows by a straightforward structural induction.

Next, we show that

$$(\widehat{\mathsf{X}}_1, \widehat{\mathsf{G}}_1, \widehat{\mathsf{D}}_1) \models C^\ell \ \wedge \ (\widehat{\mathsf{X}}_2, \widehat{\mathsf{G}}_2, \widehat{\mathsf{D}}_2) \models C^\ell \\ \Rightarrow \ (\widehat{\mathsf{X}}_1 \sqcap \widehat{\mathsf{X}}_2, \widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2, \widehat{\mathsf{D}}_1 \sqcap \widehat{\mathsf{D}}_2) \models C^\ell \qquad (**)$$

which also follows by structural induction. We just consider two of the cases:

$(C_1^{\ell_1}; C_2^{\ell_2})^\ell$**:** By the IH:

$$(\widehat{\mathsf{X}}_1 \sqcap \widehat{\mathsf{X}}_2, \widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2, \widehat{\mathsf{D}}_1 \sqcap \widehat{\mathsf{D}}_2) \models C_i^{\ell_i}$$

for $i = 1, 2$. Then

$$\begin{aligned} \widehat{\mathsf{X}}_1(\ell) \cap \widehat{\mathsf{X}}_2(\ell) &\supseteq (\widehat{\mathsf{X}}_1(\ell_1) \cup \widehat{\mathsf{X}}_1(\ell_2)) \cap (\widehat{\mathsf{X}}_2(\ell_1) \cup \widehat{\mathsf{X}}_2(\ell_2)) \\ &\supseteq (\widehat{\mathsf{X}}_1(\ell_1) \cap \widehat{\mathsf{X}}_2(\ell_1)) \cup (\widehat{\mathsf{X}}_1(\ell_2) \cap \widehat{\mathsf{X}}_2(\ell_2)) \end{aligned}$$

and, using

$$A_1 ; B_1 \cap A_2 ; B_2 \supseteq (A_1 \cap A_2) ; (B_1 \cap B_2)$$

which follows from simple set-theoretic arguments, we have:

$$\begin{aligned} \widehat{\mathsf{G}}_1(\ell) \cap \widehat{\mathsf{G}}_2(\ell) &\supseteq (\widehat{\mathsf{G}}_1(\ell_1) \cup \widehat{\mathsf{G}}_1(\ell_2) ; \widehat{\mathsf{D}}_1(\ell_1)) \cap (\widehat{\mathsf{G}}_2(\ell_1) \cup \widehat{\mathsf{G}}_2(\ell_2) ; \widehat{\mathsf{D}}_2(\ell_1)) \\ &\supseteq (\widehat{\mathsf{G}}_1(\ell_1) \cap \widehat{\mathsf{G}}_2(\ell_1)) \cup (\widehat{\mathsf{G}}_1(\ell_2) ; \widehat{\mathsf{D}}_1(\ell_1) \cap \widehat{\mathsf{G}}_2(\ell_2) ; \widehat{\mathsf{D}}_2(\ell_1)) \\ &\supseteq (\widehat{\mathsf{G}}_1(\ell_1) \cap \widehat{\mathsf{G}}_2(\ell_1)) \cup ((\widehat{\mathsf{G}}_1(\ell_2) \cap \widehat{\mathsf{G}}_2(\ell_2)) ; (\widehat{\mathsf{D}}_1(\ell_1) \cap \widehat{\mathsf{D}}_2(\ell_1))) \end{aligned}$$

and

$$\begin{aligned} \widehat{\mathsf{D}}_1(\ell) \cap \widehat{\mathsf{D}}_2(\ell) &\supseteq \widehat{\mathsf{D}}_1(\ell_2) ; \widehat{\mathsf{D}}_1(\ell_1) \cap \widehat{\mathsf{D}}_2(\ell_2) ; \widehat{\mathsf{D}}_2(\ell_1) \\ &\supseteq (\widehat{\mathsf{D}}_1(\ell_2) \cap \widehat{\mathsf{D}}_2(\ell_2)) ; (\widehat{\mathsf{D}}_1(\ell_1) \cap \widehat{\mathsf{D}}_2(\ell_1)) \end{aligned}$$

which completes this case.

$(\texttt{if}\ b\ \texttt{then}\ C_1^{\ell_1}\ \texttt{else}\ C_2^{\ell_2})^\ell$**:** Again by the IH:

$$(\widehat{\mathsf{X}}_1 \sqcap \widehat{\mathsf{X}}_2, \widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2, \widehat{\mathsf{D}}_1 \sqcap \widehat{\mathsf{D}}_2) \models C_i^{\ell_i}$$

for $i = 1, 2$. For the $\widehat{\mathsf{X}}$ and $\widehat{\mathsf{D}}$ components, the proof is similar to the case for sequential composition. This is also the case for the first constraint on $\widehat{\mathsf{G}}$. Suppose that

$$(\widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2)(\ell_1) \supseteq \{\bullet\} \ \vee \ (\widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2)(\ell_2) \supseteq \{\bullet\}$$

then either:

$$\widehat{\mathsf{G}}_1(\ell_1) \supseteq \{\bullet\} \ \wedge \ \widehat{\mathsf{G}}_2(\ell_1) \supseteq \{\bullet\}$$

or

$$\widehat{\mathsf{G}}_1(\ell_2) \supseteq \{\bullet\} \ \wedge \ \widehat{\mathsf{G}}_2(\ell_2) \supseteq \{\bullet\}$$

or both. In any case

$$(\widehat{\mathsf{G}}_1 \sqcap \widehat{\mathsf{G}}_2)(\ell) \supseteq \mathrm{FV}(b)$$

Now let

$$Y = \{(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \mid (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^{\ell}\}$$

Since **Assign** $\times$ **Global** $\times$ **Dep** is finite (we have restricted **Ide** and **Lab** to those identifiers and labels appearing in the program), we can rewrite $Y$ as $\{(\widehat{\mathsf{X}}_i, \widehat{\mathsf{G}}_i, \widehat{\mathsf{D}}_i) \mid i \in \{1..n\}\}$ for some $n$. Now

$$\sqcap Y \in \{(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \mid (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}) \models C^{\ell}\}$$

follows from $(*)$ and $(**)$ because

$$\sqcap Y = (\widehat{\mathsf{X}}_\top, \widehat{\mathsf{G}}_\top, \widehat{\mathsf{D}}_\top) \sqcap (\widehat{\mathsf{X}}_1, \widehat{\mathsf{G}}_1, \widehat{\mathsf{D}}_1) \sqcap \ldots \sqcap (\widehat{\mathsf{X}}_n, \widehat{\mathsf{G}}_n, \widehat{\mathsf{D}}_n)$$

□

## 4 An example: information flow analysis for the *While* language

The following example offers a program fragment in which $H = \{p\}$ and $L = \{x, g, f\}$. The fragment is evidently not a password checking program (which by its nature can never be entirely secure). Rather it provides some aspects (in a very crude form) of a such a program: it merely suggests a low security user having a number of attempts at guessing a password stored in $p$. The "guesses" and their number are controlled by the variables $g$ and $x$ respectively. Depending on the success of the last guess a flag, $f$, is set. We can imagine that in a larger program there would then be further processing depending on the value of this flag. Variables into which information about the password may have leaked are then "discarded" by overwriting them at the end of the program. A Volpano and Smith style type system would not type this fragment as secure although it evidently is.

Here is the program:

```
while ( x < 3 )
do ( if ( p = g )
     then ( f := 1 )
     else ( f := 0 );
     x := x + 1;
     g := g + 10);
f := 2;
x := 0
```

A labelled version of the program is as follows.

```
( ( ( while ( x < 3 )
     do ( ( ( if ( p = g )
              then ( f:= 1 )^l10
              else ( f := 0 )^l11
```

```
        )^l8;
        ( x := x + 1 )^l9
      )^l6;
      ( g := g + 10 )^l7
    )^l5
  )^l3;
  ( f := 2 )^l4
 )^l1;
 ( x := 0 )^l2
)^l0
```

Our analysis then provides the following set of constraints:

$\widehat{\mathsf{X}}(\ell_0) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \ \wedge\ \widehat{\mathsf{G}}(\ell_0) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1) \ \wedge\ \widehat{\mathsf{D}}(\ell_0) \supseteq \widehat{\mathsf{D}}(\ell_2)\,;\widehat{\mathsf{D}}(\ell_1)$

$\widehat{\mathsf{X}}(\ell_1) \supseteq \widehat{\mathsf{X}}(\ell_3) \cup \widehat{\mathsf{X}}(\ell_4) \ \wedge\ \widehat{\mathsf{G}}(\ell_1) \supseteq \widehat{\mathsf{G}}(\ell_3) \cup \widehat{\mathsf{G}}(\ell_4)\,;\widehat{\mathsf{D}}(\ell_3) \ \wedge\ \widehat{\mathsf{D}}(\ell_1) \supseteq \widehat{\mathsf{D}}(\ell_4)\,;\widehat{\mathsf{D}}(\ell_3)$

$\widehat{\mathsf{X}}(\ell_2) \supseteq \{x\} \ \wedge\ \widehat{\mathsf{D}}(\ell_2) \supseteq \mathrm{Id}[x \mapsto \emptyset]$

$\widehat{\mathsf{X}}(\ell_3) \supseteq \widehat{\mathsf{X}}(\ell_5) \ \wedge\ \widehat{\mathsf{G}}(\ell_3) \supseteq \{\bullet\} \cup \{x\} \cup \widehat{\mathsf{G}}(\ell_5) \cup \widehat{\mathsf{G}}(\ell_3)\,;\widehat{\mathsf{D}}(\ell_5) \wedge$
$\widehat{\mathsf{D}}(\ell_3) \supseteq \mathrm{Id} \cup \widehat{\mathsf{D}}(\ell_3)\,;\widehat{\mathsf{D}}(\ell_5) \ \wedge\ \widehat{\mathsf{D}}(\ell_3) \supseteq \widehat{\mathsf{X}}(\ell_3) \times \{x\}$

$\widehat{\mathsf{X}}(\ell_4) \supseteq \{f\} \ \wedge\ \widehat{\mathsf{D}}(\ell_4) \supseteq \mathrm{Id}[f \mapsto \emptyset]$

$\widehat{\mathsf{X}}(\ell_5) \supseteq \widehat{\mathsf{X}}(\ell_6) \cup \widehat{\mathsf{X}}(\ell_7) \ \wedge\ \widehat{\mathsf{G}}(\ell_5) \supseteq \widehat{\mathsf{G}}(\ell_6) \cup \widehat{\mathsf{G}}(\ell_7)\,;\widehat{\mathsf{D}}(\ell_6) \ \wedge\ \widehat{\mathsf{D}}(\ell_5) \supseteq \widehat{\mathsf{D}}(\ell_7)\,;\widehat{\mathsf{D}}(\ell_6)$

$\widehat{\mathsf{X}}(\ell_6) \supseteq \widehat{\mathsf{X}}(\ell_8) \cup \widehat{\mathsf{X}}(\ell_9) \ \wedge\ \widehat{\mathsf{G}}(\ell_6) \supseteq \widehat{\mathsf{G}}(\ell_8) \cup \widehat{\mathsf{G}}(\ell_9)\,;\widehat{\mathsf{D}}(\ell_8) \ \wedge\ \widehat{\mathsf{D}}(\ell_6) \supseteq \widehat{\mathsf{D}}(\ell_9)\,;\widehat{\mathsf{D}}(\ell_8)$

$\widehat{\mathsf{X}}(\ell_7) \supseteq \{g\} \ \wedge\ \widehat{\mathsf{D}}(\ell_7) \supseteq \mathrm{Id}[g \mapsto \emptyset]$

$\widehat{\mathsf{X}}(\ell_8) \supseteq \widehat{\mathsf{X}}(\ell_{10}) \cup \widehat{\mathsf{X}}(\ell_{11}) \wedge \widehat{\mathsf{G}}(\ell_8) \supseteq \widehat{\mathsf{G}}(\ell_{10}) \cup \widehat{\mathsf{G}}(\ell_{11}) \wedge \bullet \in \widehat{\mathsf{G}}(\ell_8) \Rightarrow \widehat{\mathsf{G}}(\ell_8) \supseteq \{p, g\} \wedge$
$\widehat{\mathsf{D}}(\ell_8) \supseteq \widehat{\mathsf{D}}(\ell_{10}) \cup \widehat{\mathsf{D}}(\ell_{11}) \ \wedge\ \widehat{\mathsf{D}}(\ell_8) \supseteq \widehat{\mathsf{X}}(\ell_8) \times \{p, g\}$

$\widehat{\mathsf{X}}(\ell_9) \supseteq \{x\} \ \wedge\ \widehat{\mathsf{D}}(\ell_9) \supseteq \mathrm{Id}[x \mapsto \emptyset]$

$\widehat{\mathsf{X}}(\ell_{10}) \supseteq \{f\} \ \wedge\ \widehat{\mathsf{D}}(\ell_{10}) \supseteq \mathrm{Id}[f \mapsto \emptyset]$

$\widehat{\mathsf{X}}(\ell_{11}) \supseteq \{f\} \ \wedge\ \widehat{\mathsf{D}}(\ell_{11}) \supseteq \mathrm{Id}[f \mapsto \emptyset]$

Iterating over these constraints beginning from $\widehat{\mathsf{X}} = \lambda x.\emptyset$, $\widehat{\mathsf{G}} = \lambda x.\emptyset$, and $\widehat{\mathsf{D}} = \lambda x.\emptyset$ to a fixed point giving the least solution yields:

$$\begin{aligned}
\widehat{\mathsf{X}}(\ell_0) &= \{f, x, g\} \\
\widehat{\mathsf{G}}(\ell_0) &= \{\bullet, x\} \\
\widehat{\mathsf{D}}(\ell_0) &= \{(p,p), (g,g), (g,x)\}
\end{aligned}$$

which satisfy the security criteria for the while language.

## 5 Information Flow for Idealised Algol

In this section, we adapt and extend the analysis of the previous section to a much richer language. We consider a Call-by-Value Idealised Algol (IA) [1]. Idealised Algol is a language developed by Reynolds [4] which extends the simply-typed $\lambda$-calculus with imperative features. The result is a prototype language combining block structured imperative programming and elements of higher-order functional programming.

The basic phrase types of IA are

$$B ::= \mathtt{Exp} \mid \mathtt{Var} \mid \mathtt{com}$$

Terms of type `Exp` evaluate to values of primitive type (integers, in our case) and terms of type `Var` evaluate to *locations*, which are mapped to values of primitive type in a store. We let $\iota$ range over locations. We assume that expressions are *passive* - only terms of type `com` may have subexpressions of type `com`. General types are constructed in the usual way:

$$T ::= B \mid T \to T$$

The syntax of the language is as follows:

$$\begin{array}{c} x \in \mathbf{Ide} \\ c \in \mathbf{Const} \\ t \in \mathbf{Term} \\ e \in \mathbf{Exp} \\ \ell \in \mathbf{Lab} \end{array}$$

$$\begin{array}{l} e ::= t^{\ell} \\ t ::= x \mid c \mid e_1\ e_2 \mid \lambda x.e \mid e_1 + e_2 \mid e_1 * e_2 \mid \cdots \mid \\ \qquad \mathtt{if}\ e_0\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \mid e_1 ; e_2 \mid \\ \qquad \mathtt{skip} \mid e_1 := e_2 \mid \mathtt{deref}\ e \mid \mathtt{while}\ e_0\ \mathtt{do}\ e_1 \mid \mathtt{new}\ x.\ e \end{array}$$

where $c$ are constants of basic type and the other constants are of type:

$$\begin{array}{l} \mathtt{if}\ :\ \mathtt{Exp} \to T \to T \to T \\ ;\ :\ \mathtt{com} \to \mathtt{com} \to \mathtt{com} \\ :=\ :\ \mathtt{Var} \to \mathtt{Exp} \to \mathtt{com} \\ \mathtt{deref}\ :\ \mathtt{Var} \to \mathtt{Exp} \\ \mathtt{while}\ :\ \mathtt{Exp} \to \mathtt{com} \to \mathtt{com} \\ \mathtt{new}\ :\ (\mathtt{Var} \to \mathtt{com}) \to \mathtt{com} \end{array}$$

We now present a call-by-value big step semantics for IA (see table 3). Here $\sigma$ is a partial map from locations (not identifiers) to integers and $\rho$ is an environment - a partial map from **Ide** to *values*, where a value may be either an integer, a location or a closure. Variables which occur bound by a $\lambda$ in the program under consideration are termed *lambda-bound* and all other variables (including the program's free variables) are termed *assignable*. Note that these sets need not be disjoint.

A lambda abstraction evaluates to a closure containing the syntax of the lambda abstraction together with an environment binding its free variables.

To ease the statement and proof of correctness for the analysis, we assume that the set of locations is partitioned into a family of sets $\{Loc_x\}$ indexed by the assignable variable names. Given a location $\iota$, we write $Var(\iota)$ to mean the (necessarily unique) $x$ such that $\iota \in Loc_x$. In the semantics of `new`, the functions $\mathrm{alloc}_x(\sigma)$ deliver a location $\iota$ such that $\iota \in Loc_x$ and $\iota \notin \mathrm{dom}(\sigma)$. Furthermore, $\mathrm{alloc}_x(\sigma) = \mathrm{alloc}_x(\sigma \mid Loc_x)$. These assumptions amount to a benign instrumentation of the semantics which serves to make the connection between variable names and locations explicit in $\sigma$ (benign because none of the rules actually rely on the assumptions). In a less abstract semantics this might not be necessary as the global execution state would be more explicit (in a big-step semantics only $\sigma$ is explicitly global, with the rest of the execution state being 'distributed' in the derivation).

We do not specify the semantics of the arithmetic operators except to require that they are total.

The analysis consists of two stages: Control Flow Analysis and Information Flow Analysis. We have based the Control Flow Analysis on a flow logic for 0-CFA presented in [3].

### 5.1 Control Flow

The Control Flow Analysis is shown in table 4. This extends the control flow analysis of *op. cit.* by handling the imperative constructs. In most cases, the imperative constructs just require analysis of the subcomponents. The rule for `new` includes information about locations to enable us to handle properly the aliasing that arises from function application. With each assignable variable $x$ we associate a unique *abstract location*, $loc_x$; the set of all such abstract locations is denoted $\widehat{\mathbf{Loc}}$. In this analysis, $\widehat{\mathsf{C}} : \mathbf{Lab} \to \mathcal{P}(\mathbf{Statement} \cup \widehat{\mathbf{Loc}})$ and $\widehat{\rho} : \mathbf{Ide} \to \mathcal{P}(\mathbf{Statement} \cup \widehat{\mathbf{Loc}})$.

We write

$$(\widehat{\mathsf{C}}, \widehat{\rho}) \models e$$

for when $(\widehat{\mathsf{C}}, \widehat{\rho})$ is an acceptable Control Flow Analysis of the expression $e$.

We defer discussion of the correctness of the CFA to section 6.

### 5.2 Information Flow

We adapt and redefine the functions on labels used for While, as follows. As the semantics makes clear, in IA there is a distinction between variable *names*

$$\frac{\rho \vdash e_1, \sigma \Downarrow \langle (\lambda x.e_0)^\ell, \rho' \rangle, \sigma' \quad \rho \vdash e_2, \sigma' \Downarrow v_2, \sigma'' \quad \rho'[x \mapsto v_2] \vdash e_0, \sigma'' \Downarrow v, \sigma'''}{\rho \vdash (e_1\ e_2)^\ell, \sigma \Downarrow v, \sigma'''}$$

$$\frac{}{\rho \vdash (\lambda x.e)^\ell, \sigma \Downarrow \langle \lambda x.e, \rho \rangle, \sigma}$$

$$\frac{}{\rho \vdash \mathtt{skip}^\ell, \sigma \Downarrow \mathtt{void}, \sigma}$$

$$\frac{}{\rho \vdash x^\ell, \sigma \Downarrow \rho(x), \sigma}$$

$$\frac{\rho[x \mapsto \iota] \vdash e, \sigma[\iota \mapsto 0] \Downarrow \mathtt{void}, \sigma'}{\rho \vdash (\mathtt{new}\ \ x.\ e)^\ell, \sigma \Downarrow \mathtt{void}, \sigma' \backslash \iota}\ \iota = \mathrm{alloc}_x(\sigma)$$

$$\frac{\rho \vdash e, \sigma \Downarrow \iota, \sigma}{\rho \vdash (\mathtt{deref}\ e)^\ell, \sigma \Downarrow \sigma(\iota), \sigma}$$

$$\frac{}{\rho \vdash c^\ell, \sigma \Downarrow c, \sigma}$$

$$\frac{\rho \vdash e_0, \sigma \Downarrow \mathtt{1}, \sigma \quad \rho \vdash e_1, \sigma \Downarrow v, \sigma'}{\rho \vdash (\mathtt{if}\ \ e_0\ \ \mathtt{then}\ \ e_1\ \ \mathtt{else}\ \ e_2)^\ell, \sigma \Downarrow v, \sigma'}$$

$$\frac{\rho \vdash e_0, \sigma \Downarrow 0, \sigma \quad \rho \vdash e_2, \sigma \Downarrow v, \sigma'}{\rho \vdash (\mathtt{if}\ \ e_0\ \ \mathtt{then}\ \ e_1\ \ \mathtt{else}\ \ e_2)^\ell, \sigma \Downarrow v, \sigma'}$$

$$\frac{\rho \vdash e_1, \sigma \Downarrow \mathtt{void}, \sigma' \quad \rho \vdash e_2, \sigma' \Downarrow \mathtt{void}, \sigma''}{\rho \vdash (e_1; e_2)^\ell, \sigma \Downarrow \mathtt{void}, \sigma''}$$

$$\frac{\rho \vdash e_0, \sigma \Downarrow 0, \sigma}{\rho \vdash (\mathtt{while}\ \ e_0\ \ \mathtt{do}\ \ e_1)^\ell, \sigma \Downarrow \mathtt{void}, \sigma}$$

$$\frac{\rho \vdash e_0, \sigma \Downarrow \mathtt{1}, \sigma \quad \rho \vdash e_1, \sigma \Downarrow \mathtt{void}, \sigma' \quad \rho \vdash (\mathtt{while}\ \ e_0\ \ \mathtt{do}\ \ e_1)^\ell, \sigma' \Downarrow \mathtt{void}, \sigma''}{\rho \vdash (\mathtt{while}\ \ e_0\ \ \mathtt{do}\ \ e_1)^\ell, \sigma \Downarrow \mathtt{void}, \sigma''}$$

$$\frac{\rho \vdash e_1, \sigma \Downarrow \iota, \sigma' \quad \rho \vdash e_2, \sigma' \Downarrow v, \sigma'}{\rho \vdash (e_1 := e_2)^\ell, \sigma \Downarrow \mathtt{void}, \sigma'[\iota \mapsto v]}$$

**Table 3.** Natural Semantics for IA

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models \texttt{skip}^{\ell} \text{ always}$$

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models c^{\ell} \text{ always}$$

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models x^{\ell} \text{ iff } \widehat{\rho}(x) \subseteq \widehat{\mathsf{C}}(\ell)$$

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models (e1 + e2)^{\ell} \text{ iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models e_1 \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models e_2$$

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models (\lambda x.\ e_0)^{\ell} \text{ iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models e_0 \;\wedge\; \{\lambda x.\ e_0\} \subseteq \widehat{\mathsf{C}}(\ell)$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (\texttt{if } t_0^{\ell_0} \texttt{ then } t_1^{\ell_1} \texttt{ else } t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_0^{\ell_0} \;\wedge \\
\qquad\quad (\widehat{\mathsf{C}},\widehat{\rho}) \models t_1^{\ell_1} \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models t_2^{\ell_2} \;\wedge \\
\qquad\quad \widehat{\mathsf{C}}(\ell_1) \cup \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\mathsf{C}}(\ell)
\end{array}$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (t_1^{\ell_1}; t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_1^{\ell_1} \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models t_2^{\ell_2}
\end{array}$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (t_1^{\ell_1} := t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_1^{\ell_1} \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models t_2^{\ell_2}
\end{array}$$

$$(\widehat{\mathsf{C}},\widehat{\rho}) \models (\texttt{deref } x)^{\ell} \text{ always}$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (\texttt{while } t_0^{\ell_0} \texttt{ do } t_1^{\ell_1})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_0^{\ell_0} \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models t_1^{\ell_1}
\end{array}$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_1^{\ell_1} \;\wedge\; (\widehat{\mathsf{C}},\widehat{\rho}) \models t_2^{\ell_2} \;\wedge \\
\qquad\quad (\forall (\lambda x.\ t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1) : \\
\qquad\qquad\qquad \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \;\wedge\; \widehat{\mathsf{C}}(\ell_0) \subseteq \widehat{\mathsf{C}}(\ell))
\end{array}$$

$$\begin{array}{l}
(\widehat{\mathsf{C}},\widehat{\rho}) \models (\texttt{new } x.\ t_0^{\ell_0})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{C}},\widehat{\rho}) \models t_0^{\ell_0} \;\wedge\; \{loc_x\} \subseteq \widehat{\rho}(x)
\end{array}$$

**Table 4.** Syntax directed Control Flow Analysis

(identifiers) and variables (locations). Accordingly, we define the set $\widehat{\mathbf{Var}} = \widehat{\mathbf{Ide}} \cup \widehat{\mathbf{Loc}}$ and the maps $\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}$ now have types:

$$\begin{aligned} \widehat{\mathsf{X}} &\in \mathbf{Assign} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Loc}}) \\ \widehat{\mathsf{G}} &\in \mathbf{Global} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Var}}) \\ \widehat{\mathsf{D}} &\in \mathbf{Dep} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Loc}} \times \widehat{\mathbf{Var}}) \end{aligned}$$

Compared with the analysis for the simple While language, our information flow analysis requires one additional component:

$$\widehat{\mathsf{V}} \in \mathbf{Assign} = \mathbf{Lab} \to \mathcal{P}(\widehat{\mathbf{Var}})$$

Given a label, $\widehat{\mathsf{V}}$ returns a superset of those identifiers and variables whose values may affect the value of the term with that label. In the While language this component was unnecessary because the set of free variables of a term could be used: here, due to the presence of procedure calls, the free variables of a term do not suffice.

In the definition of the analysis, we often need to project out the separate parts of a set or relation which deal with identifiers and locations. Hence we define:

$$\begin{aligned} \widehat{\mathsf{D}}_I(\ell) &= \widehat{\mathsf{D}}(\ell) \cap (\widehat{\mathbf{Loc}} \times \widehat{\mathbf{Ide}}) & \widehat{\mathsf{D}}_L(\ell) &= \widehat{\mathsf{D}}(\ell) \cap (\widehat{\mathbf{Loc}} \times \widehat{\mathbf{Loc}}) \\ \widehat{\mathsf{V}}_I(\ell) &= \widehat{\mathsf{V}}(\ell) \cap \widehat{\mathbf{Ide}} & \widehat{\mathsf{V}}_L(\ell) &= \widehat{\mathsf{V}}(\ell) \cap \widehat{\mathbf{Loc}} \\ \widehat{\mathsf{G}}_I(\ell) &= \widehat{\mathsf{G}}(\ell) \cap \widehat{\mathbf{Ide}} & \widehat{\mathsf{G}}_L(\ell) &= \widehat{\mathsf{G}}(\ell) \cap \widehat{\mathbf{Loc}} \\ \mathrm{Id}_I &= \{(x, x) \mid x \in \widehat{\mathbf{Ide}}\} & \mathrm{Id}_L &= \{(loc_x, loc_x) \mid loc_x \in \widehat{\mathbf{Loc}}\} \end{aligned}$$

We write

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models_{(\widehat{\mathsf{C}}, \widehat{\rho})} e$$

for when $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}})$ is an acceptable Information Flow Analysis of the expression $e$ with respect to the Control Flow Analysis $(\widehat{\mathsf{C}}, \widehat{\rho})$. We usually omit the $(\widehat{\mathsf{C}}, \widehat{\rho})$ subscript. The rules are presented in tables 5, 6 and 7. We note that the rules for constructs which can only evaluate to `void` do not need to constrain $\widehat{\mathsf{V}}$ since their value cannot vary.

**Table 5** As for the While language, for expressions which do not affect the store, the analysis constrains $\widehat{\mathsf{D}}$ to include the identity relation (here on $\widehat{\mathbf{Loc}}$ rather than $\mathbf{Ide}$).

The rule for $x$ constrains $\widehat{\mathsf{V}}$ to contain $x$ since the value of $x$ obviously depends on itself.

In the evaluation of (`deref` $e$), first $e$ is evaluated to give a location, $\iota$, and then $\iota$ is dereferenced. Since the final value obtained depends both on the choice of $\iota$ and the value of $\sigma(\iota)$, the rule constrains $\widehat{\mathsf{V}}$ to contain both the dependencies for $e$ and the locations to which $e$ may evaluate.

In the rule for (`new` $x$. $e$), dependencies of $e$ on $x$ are removed from dependencies for the whole command, since $x$ is local. However, dependencies on $loc_x$

are not removed since aliasing allows the possibility of dependencies on locations allocated to $x$ in an outer scope. Conversely, since the analysis does not distinguish between different occurrences of $x$, we also have to include the dependency $(loc_x, loc_x)$ to allow for the possibility that $x$ allocated in an outer scope is unaffected by any assignments to $x$ in $e$.

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models c^\ell \text{ iff } \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L$$

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models \mathtt{skip}^\ell \text{ iff } \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L$$

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (\lambda x.\ e_0)^\ell \text{ iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models e_0 \wedge \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L$$

$$(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models x^\ell \text{ iff } \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L \wedge x \in \widehat{\mathsf{V}}(\ell)$$

$$\begin{array}{l} (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (t_1^{\ell_1} + t_2^{\ell_2})^\ell \\ \quad\quad \text{iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\ \quad\quad\quad \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L \wedge \\ \quad\quad\quad \widehat{\mathsf{V}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_1) \cup \widehat{\mathsf{V}}(\ell_2) \end{array}$$

$$\begin{array}{l} (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (\mathtt{deref}\ t_0^{\ell_0})^\ell \\ \quad\quad \text{iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_0^{\ell_0} \wedge \\ \quad\quad\quad \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L \wedge \\ \quad\quad\quad \widehat{\mathsf{V}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_0) \cup \{loc_x \mid loc_x \in \widehat{\mathsf{C}}(\ell_0)\} \end{array}$$

$$\begin{array}{l} (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (\mathtt{new}\ x.\ t_0^{\ell_0})^\ell \\ \quad\quad \text{iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_0^{\ell_0} \wedge \\ \quad\quad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_0) \wedge \\ \quad\quad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_0) \wedge \\ \quad\quad\quad \widehat{\mathsf{D}}(\ell) \supseteq \{(loc_y, z) \in \widehat{\mathsf{D}}_I(\ell_0) \mid z \neq x\} \cup \widehat{\mathsf{D}}_L(\ell_0) \cup \{(loc_x, loc_x)\} \end{array}$$

**Table 5.** Syntax directed Information Flow Analysis – Part 1

**Table 6** In the rule for a sequential composition $(e_1; e_2)$, the dependencies for $e_2$ are split into dependencies on names ($\widehat{\mathsf{G}}_I$ and $\widehat{\mathsf{D}}_I$) and dependencies on locations ($\widehat{\mathsf{G}}_L$ and $\widehat{\mathsf{D}}_L$). The dependencies for the composition are required to contain the $e_2$ dependencies on names since the bindings for names are not altered by $e_1$. The $e_2$ dependencies on locations, however, must be composed with the $e_1$ dependencies. As an example, consider $x := 0; z := (\mathtt{deref}\ x) + y$, where we may assume that $x$ and $z$ are assignable while $y$ is lambda-bound. Our analysis for the second assignment would reveal that it makes $loc_z$ dependent on $loc_x$ and $y$. The dependency on $y$ is unaffected by the first assignment but, overall, there is no dependency of the final value stored for $z$ on the initial value stored for $x$, since the first assignment makes the value stored for $x$ independent of the initial state.

The rule for conditionals is essentially the same as for the While language except that here we use $\widehat{\mathsf{V}}(\ell_0)$ instead of $\mathrm{FV}(e_0)$ and we must constrain $\widehat{\mathsf{V}}(\ell)$ in the obvious way.

As for the While language, the rule for `while` is derived by unwinding the loop once and applying the rules for conditionals and sequential composition.

---

$$
\begin{array}{l}
(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models (t_1^{\ell_1} ; t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\
\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}_I(\ell_2) \cup \widehat{\mathsf{G}}_L(\ell_2) \,;\, \widehat{\mathsf{D}}(\ell_1) \wedge \\
\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}_I(\ell_2) \cup \widehat{\mathsf{D}}_L(\ell_2) \,;\, \widehat{\mathsf{D}}(\ell_1)
\end{array}
$$

$$
\begin{array}{l}
(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models (\mathtt{if}\ t_0^{\ell_0}\ \mathtt{then}\ t_1^{\ell_1}\ \mathtt{else}\ t_2^{\ell_2})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_0^{\ell_0} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\
\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2) \wedge \\
\qquad\quad (\bullet \in \widehat{\mathsf{G}}(\ell) \Rightarrow \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_0)) \wedge \\
\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}(\ell_1) \cup \widehat{\mathsf{D}}(\ell_2) \wedge \\
\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \widehat{\mathsf{V}}(\ell_0) \wedge \\
\qquad\quad \widehat{\mathsf{V}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_0) \cup \widehat{\mathsf{V}}(\ell_1) \cup \widehat{\mathsf{V}}(\ell_2)
\end{array}
$$

$$
\begin{array}{l}
(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models (\mathtt{while}\ t_0^{\ell_0}\ \mathtt{do}\ t_1^{\ell_1})^{\ell} \\
\qquad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_0^{\ell_0} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge \\
\qquad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \wedge \\
\qquad\quad \widehat{\mathsf{G}}(\ell) \supseteq \{\bullet\} \cup \widehat{\mathsf{V}}(\ell_0) \cup \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}_L(\ell)) \,;\, \widehat{\mathsf{D}}(\ell_1) \wedge \\
\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L \cup \widehat{\mathsf{D}}_L(\ell) \,;\, \widehat{\mathsf{D}}(\ell_1) \wedge \\
\qquad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell) \times \widehat{\mathsf{V}}(\ell_0)
\end{array}
$$

---

**Table 6.** Syntax directed Information Flow Analysis – Part 2

**Table 7** The rule for an assignment $(e_1 := e_2)$ reflects the fact that, in IA, the target for an assignment is computed rather than being explicit in the syntax: the set of possible target locations is given by the CFA for $e_1$. Each candidate target $loc_x$ then gives rise to a constraint equivalent to the single constraint for assignment in the While language.

The rule for an application $(e_1\ e_2)$ is the most complex rule in the analysis. The complexity is partly due to the need to compose dependencies on lambda-bound names with the dependencies for the actual parameters and partly due to the dependencies arising from possible side-effects (assignments) during the evaluation of $e_1$ and $e_2$. Unlike the case for sequential composition, we cannot assume that an application evaluates to `void` and so it is necessary to constrain $\widehat{\mathsf{V}}$.

The semantics of application entails that first $e_1$ and then $e_2$ are evaluated. This part of the behaviour can be treated just like sequential composition and it can be seen that the auxiliary relation $\mathcal{D}$ is derived from the dependencies for $e_1$ and $e_2$ just as in the rule for $(e_1; e_2)$.

The CFA is then used to determine those abstractions to which $e_1$ can evaluate and each possibility gives rise to a set of constraints. The location dependencies for each case are those for the body of the abstraction composed with $\mathcal{D}$. The name dependencies for the body are modified by replacing dependencies on the lambda-bound variable $x$ by the value dependencies of the actual parameter, given here by $\mathcal{V}$.

$$
\begin{array}{l}
(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (t_1^{\ell_1} := t_2^{\ell_2})^{\ell} \\
\quad \text{iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\
\quad\quad \widehat{\mathsf{X}}(\ell) \supseteq \{ loc_x \mid loc_x \in \widehat{\mathsf{C}}(\ell_1) \} \wedge \\
\quad\quad \forall loc_x \in \widehat{\mathsf{C}}(\ell_1).\ \widehat{\mathsf{D}}(\ell) \supseteq \mathrm{Id}_L[loc_x \mapsto \widehat{\mathsf{V}}(\ell_2)]
\end{array}
$$

$$
\begin{array}{l}
(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models (t_1^{\ell_1}\ t_2^{\ell_2})^{\ell} \\
\quad \text{iff } (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\
\quad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
\quad\quad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_1) \cup \widehat{\mathsf{G}}(\ell_2) \wedge \\
\quad\quad \widehat{\mathsf{V}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_1) \wedge \\
\quad\quad \forall (\lambda x.\ t_0^{\ell_0}) \in \widehat{\mathsf{C}}(\ell_1). \\
\quad\quad\quad\quad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_0) \wedge \\
\quad\quad\quad\quad \widehat{\mathsf{G}}(\ell) \supseteq (\widehat{\mathsf{G}}_I(\ell_0) \cup \mathcal{V} \cup (\widehat{\mathsf{G}}_L(\ell_0) ; \mathcal{D}) \wedge \\
\quad\quad\quad\quad \widehat{\mathsf{V}}(\ell) \supseteq (\widehat{\mathsf{V}}_I(\ell_0) \cup \mathcal{V} \cup (\widehat{\mathsf{V}}_L(\ell_0) ; \mathcal{D}) \wedge \\
\quad\quad\quad\quad \widehat{\mathsf{D}}(\ell) \supseteq (\widehat{\mathsf{D}}_I(\ell_0) ; \mathrm{Id}_I[x \mapsto \mathcal{V}]) \cup (\widehat{\mathsf{D}}_L(\ell_0) ; \mathcal{D}) \wedge \\
\quad\quad\quad\quad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_0) \times \widehat{\mathsf{V}}(\ell_1) \\
\\
\quad\quad \text{where } \mathcal{V} = \widehat{\mathsf{V}}_I(\ell_2) \cup \widehat{\mathsf{V}}_L(\ell_2) ; \widehat{\mathsf{D}}(\ell_1) \\
\quad\quad\quad\quad\quad\ \ \mathcal{D} = \widehat{\mathsf{D}}_I(\ell_2) \cup \widehat{\mathsf{D}}_L(\ell_2) ; \widehat{\mathsf{D}}(\ell_1)
\end{array}
$$

**Table 7.** Syntax directed Information Flow Analysis – Part 3

## 6 Correctness of the analysis for Idealised Algol

The statement of the correctness results requires that the environments in the semantics and the CFA analysis should be related by a correctness relation $\mathcal{R}$.

**Definition 1 (CFA Correctness Relation).** $\mathcal{R}$ is the smallest relation between environments and CFA environments such that $\rho\ \mathcal{R}\ \widehat{\rho}$ if, for all $x$:

1. if $\rho(x) = \langle t, \rho' \rangle$ then $t \in \widehat{\rho}(x)$ and $\rho'\ \mathcal{R}\ \widehat{\rho}$;
2. if $\rho(x) = \iota$ then $loc_y \in \widehat{\rho}(x)$, where $y = Var(\iota)$.

**Theorem 5 (CFA Correctness).** Let $\rho\ \mathcal{R}\ \widehat{\rho}$ and $(\widehat{\mathsf{C}}, \widehat{\rho}) \models t^\ell$ and $\rho \vdash t^\ell, \sigma \Downarrow v, \sigma'$. Then:

1. if $v = \langle t', \rho' \rangle$ then $t' \in \widehat{\mathsf{C}}(\ell) \wedge \rho'\ \mathcal{R}\ \widehat{\rho}$;
2. if $v = \iota$ then $loc_y \in \widehat{\mathsf{C}}(\ell)$, where $y = Var(\iota)$.

*Proof.* By induction on the height of the derivation. We proceed by cases on the form of $t$.

**$\lambda x.e$:** Immediate.

**$x$:** The result follows by the assumption that $\rho\ \mathcal{R}\ \widehat{\rho}$ and the constraint $\widehat{\rho}(x) \in \widehat{\mathsf{C}}(\ell)$.

**$t_1^{\ell_1} t_2^{\ell_2}$:** We have $\rho \vdash t_1^{\ell_1}, \sigma \Downarrow \langle \lambda x.e_0, \rho'' \rangle, \sigma''$ and $\rho \vdash t_2^{\ell_2}, \sigma'' \Downarrow v_2, \sigma'''$ and $\rho''[x \mapsto v_2] \vdash e_0, \sigma''' \Downarrow v, \sigma'$.

By the constraints on $\widehat{\mathsf{C}}(\ell)$ for applications we have $(\widehat{\mathsf{C}}, \widehat{\rho}) \models t_i^{\ell_i}, i = 1, 2$. By IH (derivation for $t_1$) we have $\rho''\ \mathcal{R}\ \widehat{\rho}$ and $\lambda x.e_0 \in whC(\ell_1)$. Then we must consider two cases for $v_2$:

1. If $v_2 = \langle t'', \rho''' \rangle$ then, by IH (derivation for $t_2$) we have $\rho'''\ \mathcal{R}\ \widehat{\mathsf{C}}$ and $t'' \in \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x)$, hence $t'' \in \widehat{\rho}(x)$, hence $\rho''[x \mapsto v_2]\ \mathcal{R}\ \widehat{\rho}$.
2. If $v_2 = \iota$ then let $y = Var(\iota)$. By IH (derivation for $t_2$) we have $loc_y \in \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x)$, hence $loc_y \in \widehat{\rho}(x)$, hence $\rho''[x \mapsto v_2]\ \mathcal{R}\ \widehat{\rho}$.

The result follows by IH (derivation for $e_0$).

**`if` $e_0$ `then` $e_1$ `else` $e_2$:** Immediate application of IH to sub-derivations.

The remaining cases hold trivially. □

For $X \subseteq \widehat{\mathbf{Var}}$ we define the family $\{\sim_X\}$ to be the smallest relations such that:

1. $n \sim_X n$ for all integers $n$
2. $\iota \sim_X \iota$ for all locations $\iota$
3. $\langle \lambda x.e, \rho_1 \rangle \sim_X \langle \lambda x.e, \rho_2 \rangle$ if $\rho_1 \sim_{X \setminus x} \rho_2$
4. $\rho_1 \sim_X \rho_2$ if $\forall x \in X \cap \mathbf{Ide}.\ \rho_1(x) \sim_X \rho_2(x)$
5. $\sigma_1 \sim_X \sigma_2$ if $\forall loc_x \in X.\ \forall \iota \in Loc_x.\ \sigma_1(\iota) \doteq \sigma_2(\iota)$

where $\doteq$ is Kleene equivalence ($\sigma_1(\iota) \doteq \sigma_2(\iota)$ if $\sigma_1$ and $\sigma_2$ are either both undefined on $\iota$ or they are both defined and equal on $\iota$).

**Theorem 6 (Assignment Freedom (IA)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t^\ell$ and $\rho\ \mathcal{R}\ \widehat{\rho}$ and $\rho \vdash t^\ell, \sigma \Downarrow \mathtt{void}, \sigma'$. Let $X' = \{loc_x \in \widehat{\mathbf{Loc}} \mid loc_x \notin \widehat{\mathsf{X}}(\ell)\}$. Then:

1. $\sigma' \sim_{X'} \sigma$
2. if $loc_x \notin \widehat{\mathsf{X}}(\ell)$ then $loc_x\ \widehat{\mathsf{D}}(\ell)\ loc_x$

*Proof.* Essentially as for the While language except that part 2 uses the assumption that $\rho \vdash t^\ell, \sigma \Downarrow \texttt{void}, \sigma'$. This is to rule out the possibility that $t^\ell$ occurs in the body of an abstraction which is never applied, in which case it may contain an assignment ($x := e$) for a lambda-bound $x$ which is not constrained by the CFA, resulting in an empty dependency relation. □

**Theorem 7 (Store Independence (IA)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t^\ell$, and $\rho_i \mathcal{R} \widehat{\rho}$, $i = 1, 2$. Then:

1. for all $X \supseteq \widehat{\mathsf{V}}(\ell)$

$$\begin{array}{l} \text{if } (\rho_1 \sim_X \rho_2) \wedge (\sigma_1 \sim_X \sigma_2) \text{then} \\ \quad \text{if } (\rho_1 \vdash t^\ell, \sigma_1 \Downarrow v_1, \sigma_1' \wedge \rho_2 \vdash t^\ell, \sigma_2 \Downarrow v_2, \sigma_2') \\ \quad \text{then } v_1 \sim_X v_2 \end{array}$$

2. for all assignable $x$:

$$\begin{array}{l} \text{if } (\rho_1 \sim_{\widehat{\mathsf{D}}(\ell)} \rho_2) \wedge (\sigma_1 \sim_{\widehat{\mathsf{D}}(\ell)} \sigma_2) \text{then} \\ \quad \text{if } (\rho_1 \vdash t^\ell, \sigma_1 \Downarrow v_1, \sigma_1' \wedge \rho_2 \vdash t^\ell, \sigma_2 \Downarrow v_2, \sigma_2') \\ \quad \text{then } \sigma_1' \sim_{loc_x} \sigma_2' \end{array}$$

**Theorem 8 (Termination Independence (IA)).** Suppose $(\widehat{\mathsf{X}}, \widehat{\mathsf{G}}, \widehat{\mathsf{D}}, \widehat{\mathsf{V}}) \models t^\ell$ and $\rho, \rho_1, \rho_2 \mathcal{R} \widehat{\rho}$. Then:

1. If $\bullet \notin \widehat{\mathsf{G}}(\ell)$, then, for all $\sigma$, $\rho \vdash t^\ell, \sigma \Downarrow$.
2. If $(\rho_1 \sim_{\widehat{\mathsf{G}}(\ell)} \rho_2) \wedge (\sigma_1 \sim_{\widehat{\mathsf{G}}(\ell)} \sigma_2)$ then

$$\rho_1 \vdash t^\ell, \sigma_1 \Downarrow \Leftrightarrow \rho_2 \vdash t^\ell, \sigma_2 \Downarrow$$

## 7 An example: information flow analysis for Idealised Algol

We take the if statement used in section 4 and embed it into a function application rather than the while loop of that example. As before, the variable $f$, into which information flows from the conditional expression of the if statement, is overwritten at the end of the program, and once again the the program is secure even though it breaks the kinds of rules one would expect to find in a type system approach. We allow a free variable, $p$, to store the secret and for simplicity's sake suggest the input of a guess by the user with a constant.

Here is the unlabelled program:

$$\texttt{new } f.((\lambda x.(x; f := 2))(\texttt{if } (p = 100)\texttt{then } (f := 0)\texttt{else } (f := 1)))$$

A labelled version of the same:

$$\begin{array}{l}(\texttt{new } f.((\lambda x.(x^{\ell_9}; (f^{\ell_{10}} := 2^{\ell_{11}})^{\ell_8})^{\ell_7})^{\ell_6} \\ (\texttt{if } (p^{\ell_{12}} = 100^{\ell_{13}})^{\ell_5}\texttt{then } (f^{\ell_{14}} := 0^{\ell_{15}})^{\ell_4}\texttt{else } (f^{\ell_{16}} := 1^{\ell_{17}})^{\ell_3})^{\ell_2})^{\ell_1})^{\ell_0}\end{array}$$

The control flow analysis then provides the following set of constraints:

$$\begin{array}{l}
\{loc_f\} \subseteq \widehat{\rho}(f) \\
\forall(\lambda x.t^{l_z}) \in \widehat{\mathsf{C}}(\ell_6) \;.\; \widehat{\mathsf{C}}(\ell_2) \subseteq \widehat{\rho}(x) \wedge \widehat{\mathsf{C}}(\ell_z) \subseteq \widehat{\mathsf{C}}(\ell_1) \\
\{\lambda x.(x^{\ell_9}; (f^{\ell_{10}} := 2^{\ell_{11}})^{\ell_8})^{\ell_7}\} \subseteq \widehat{\mathsf{C}}(\ell_6) \\
\widehat{\rho}(x) \subseteq \widehat{\mathsf{C}}(\ell_9) \\
\widehat{\rho}(f) \subseteq \widehat{\mathsf{C}}(\ell_{10}) \\
\widehat{\mathsf{C}}(\ell_4) \cup \widehat{\mathsf{C}}(\ell_3) \subseteq \widehat{\mathsf{C}}(\ell_2) \\
\widehat{\rho}(p) \subseteq \widehat{\mathsf{C}}(\ell_{13}) \\
\widehat{\rho}(f) \subseteq \widehat{\mathsf{C}}(\ell_{14}) \\
\widehat{\rho}(f) \subseteq \widehat{\mathsf{C}}(\ell_{16})
\end{array}$$

Since $p$ is a free variable, a solution obtained by iterating over the constraints from $\perp$ will not say anything useful about the environment for this variable. Since we assume that this is an assignable variable and that the larger program will associate a location with it (and since we wish the correctness relation between actual and abstract environments to hold) we iterate from the assumption of a location for $p$, i.e. from $\widehat{\rho} = (\lambda x.\emptyset)[p \mapsto \{loc_p\}]$, $\widehat{\mathsf{C}} = \lambda x.\emptyset$. This yields the following guess for the abstract environment and abstract cache:

$$\begin{array}{l}
\widehat{\rho} = (\lambda x.\emptyset)\ [f \mapsto \{loc_f\}, p \mapsto \{loc_p]\\
\widehat{\mathsf{C}} = (\lambda x.\emptyset)\ [l_6 \mapsto \{\lambda x.(x^{l_9}; (f^{l_{10}} := 2^{l_{11}})^{l_8})^{l_7}\}, \\
\qquad\qquad l_{10} \mapsto \{loc_f\}, l_{13} \mapsto \{loc_p\}, l_{14} \mapsto \{loc_f\}, l_{16} \mapsto \{loc_f\}]
\end{array}$$

The information flow analysis provides the following constraints:

$$\widehat{\mathsf{X}}(\ell_0) \supseteq \widehat{\mathsf{X}}(\ell_1) \wedge \widehat{\mathsf{G}}(\ell_0) \supseteq \widehat{\mathsf{G}}(\ell_1) \wedge \widehat{\mathsf{D}}(\ell_0) \supseteq \widehat{\mathsf{D}}(\ell_1) \cup \{(loc_f, loc_f)\}$$

$$\begin{array}{l}
\widehat{\mathsf{X}}(\ell_1) \supseteq \widehat{\mathsf{X}}(\ell_6) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \widehat{\mathsf{G}}(\ell_1) \supseteq \widehat{\mathsf{G}}(\ell_6) \cup \widehat{\mathsf{G}}(\ell_2) \wedge \widehat{\mathsf{V}}(\ell_1) \supseteq \widehat{\mathsf{V}}(\ell_6) \wedge \\
\forall(\lambda x.t^{\ell_z}) \in \widehat{\mathsf{C}}(\ell_6) \;.\; \begin{array}[t]{l}
\widehat{\mathsf{X}}(\ell_1) \supseteq \widehat{\mathsf{X}}(\ell_z) \wedge \\
\widehat{\mathsf{G}}(\ell_1) \supseteq \widehat{\mathsf{G}}_I(\ell_z) \cup \mathcal{V} \cup (\widehat{\mathsf{G}}_L(\ell_z); \mathcal{D}) \wedge \\
\widehat{\mathsf{V}}(\ell_1) \supseteq \widehat{\mathsf{V}}_I(\ell_z) \cup \mathcal{V} \cup (\widehat{\mathsf{V}}_L(\ell_z); \mathcal{D}) \wedge \\
\widehat{\mathsf{D}}(\ell_1) \supseteq (\widehat{\mathsf{D}}_I(\ell_z); \mathrm{Id}_I[x \mapsto \mathcal{V}]) \cup (\widehat{\mathsf{D}}(\ell_z); \mathcal{D}) \wedge \\
\widehat{\mathsf{D}}(\ell_1) \supseteq \widehat{\mathsf{X}}(\ell_z) \times \widehat{\mathsf{V}}(\ell_6)
\end{array} \\
\text{where } \begin{array}[t]{l}
\mathcal{V} = \widehat{\mathsf{V}}_I(\ell_2) \cup (\widehat{\mathsf{V}}_L(\ell_2); \widehat{\mathsf{D}}(\ell_6)) \\
\mathcal{D} = \widehat{\mathsf{D}}_I(\ell_2) \cup (\widehat{\mathsf{D}}_L(\ell_2); \widehat{\mathsf{D}}(\ell_6))
\end{array}
\end{array}$$

$$\widehat{\mathsf{D}}(\ell_6) \supseteq \mathrm{Id}_L$$

$$\begin{array}{l}
\widehat{\mathsf{X}}(\ell_7) \supseteq \widehat{\mathsf{X}}(\ell_9) \cup \widehat{\mathsf{X}}(\ell_8) \wedge \\
\widehat{\mathsf{G}}(\ell_7) \supseteq \widehat{\mathsf{G}}(\ell_9) \cup \widehat{\mathsf{G}}_I(\ell_8) \cup (\widehat{\mathsf{G}}_L(\ell_8); \widehat{\mathsf{D}}(\ell_9)) \wedge \\
\widehat{\mathsf{D}}(\ell_7) \supseteq \widehat{\mathsf{D}}_I(\ell_8) \cup (\widehat{\mathsf{D}}_L(\ell_8); \widehat{\mathsf{D}}(\ell_9))
\end{array}$$

$$\widehat{\mathsf{D}}(\ell_9) \supseteq \mathrm{Id}_L \wedge x \in \widehat{\mathsf{V}}(\ell_9)$$

$\widehat{\mathsf{X}}(\ell_8) \supseteq \{loc_x \mid loc_x \in \widehat{\mathsf{C}}(\ell_{10})\} \wedge$
$\forall loc_x \in \widehat{\mathsf{C}}(\ell_{10}) \;.\; \widehat{\mathsf{D}}(\ell_8) \supseteq \mathrm{Id}_L[loc_x \mapsto \widehat{\mathsf{V}}(\ell_{11})]$

$\widehat{\mathsf{D}}(\ell_{11}) \supseteq \mathrm{Id}_L$

$\widehat{\mathsf{X}}(\ell_2) \supseteq \widehat{\mathsf{X}}(\ell_4) \cup \widehat{\mathsf{X}}(\ell_3) \wedge$
$\widehat{\mathsf{G}}(\ell_2) \supseteq \widehat{\mathsf{G}}(\ell_4) \cup \widehat{\mathsf{G}}(\ell_3) \wedge$
$\bullet \in \widehat{\mathsf{G}}(\ell_2) \Rightarrow \widehat{\mathsf{G}}(\ell_2) \supseteq \widehat{\mathsf{V}}(\ell_5) \wedge$
$\widehat{\mathsf{D}}(\ell_2) \supseteq \widehat{\mathsf{D}}(\ell_4) \cup \widehat{\mathsf{D}}(\ell_3)$
$\widehat{\mathsf{D}}(\ell_2) \supseteq \widehat{\mathsf{X}}(\ell_2) \times \widehat{\mathsf{V}}(\ell_5) \wedge$
$\widehat{\mathsf{V}}(\ell_2) \supseteq \widehat{\mathsf{V}}(\ell_5) \cup \widehat{\mathsf{V}}(\ell_4) \cup \widehat{\mathsf{V}}(\ell_3)$

$\widehat{\mathsf{D}}(\ell_5) \supseteq \mathrm{Id}_L \wedge \widehat{\mathsf{V}}(\ell_5) \supseteq \widehat{\mathsf{V}}(\ell_{12}) \cup \widehat{\mathsf{V}}(\ell_{13})$

$\widehat{\mathsf{D}}(\ell_{12}) \supseteq \mathrm{Id}_L \wedge g \in \widehat{\mathsf{V}}(\ell_{12})$

$\widehat{\mathsf{D}}(\ell_{13}) \supseteq \mathrm{Id}_L$

$\widehat{\mathsf{X}}(\ell_4) \supseteq \{loc_x \mid loc_x \in \widehat{\mathsf{C}}(\ell_{14})\} \wedge$
$\forall loc_x \in \widehat{\mathsf{C}}(\ell_{14}) \;.\; \widehat{\mathsf{D}}(\ell_4) \supseteq \mathrm{Id}_L[loc_x \mapsto \widehat{\mathsf{V}}(\ell_{15})]$

$\widehat{\mathsf{X}}(\ell_3) \supseteq \{loc_x \mid loc_x \in \widehat{\mathsf{C}}(\ell_{16})\} \wedge$
$\forall loc_x \in \widehat{\mathsf{C}}(\ell_{16}) \;.\; \widehat{\mathsf{D}}(\ell_3) \supseteq \mathrm{Id}_L[loc_x \mapsto \widehat{\mathsf{V}}(\ell_{17})]$

$\widehat{\mathsf{D}}(\ell_{16}) \supseteq \mathrm{Id}_L \wedge f \in \widehat{\mathsf{V}}(\ell_{16})$

$\widehat{\mathsf{D}}(\ell_{17}) \supseteq \mathrm{Id}_L$

Iterating over these constraints beginning from $\widehat{\mathsf{X}} = \lambda x.\emptyset$, $\widehat{\mathsf{G}} = \lambda x.\emptyset$, $\widehat{\mathsf{D}} = \lambda x.\emptyset$ and $\widehat{\mathsf{V}} = \lambda x.\emptyset$ to a fixed point giving the least solution yields:

$$\begin{aligned}
\widehat{\mathsf{X}}(\ell_0) &= \{loc_f\} \\
\widehat{\mathsf{G}}(\ell_0) &= \emptyset \\
\widehat{\mathsf{D}}(\ell_0) &= \mathrm{Id}_L \\
\widehat{\mathsf{V}}(\ell_0) &= \emptyset
\end{aligned}$$

which is easily seen to have no flow of information from $p$, regarded as high, to $f$, regarded as low.

## 8 Conclusions

There have been a number of papers using type systems to analyse information flow; [6] is a good example. In contrast to these approaches our analyses are flow sensitive. This allows us to achieve a more accurate analysis of information flow under some circumstances. It is also difficult to give an elegant treatment of implicit global flows in the type-based setting.

In our treatment of IA we have assumed that terms of type `Exp` are passive. It is also possible to allow active expressions at the cost of some additional complexity in the analysis. With active expressions, the rule for conditional (for example) becomes:

$$
\begin{array}{l}
(\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models (\texttt{if } t_0^{\ell_0} \texttt{ then } t_1^{\ell_1} \texttt{ else } t_2^{\ell_2})^{\ell} \\
\quad \text{iff } (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_0^{\ell_0} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_1^{\ell_1} \wedge (\widehat{\mathsf{X}},\widehat{\mathsf{G}},\widehat{\mathsf{D}},\widehat{\mathsf{V}}) \models t_2^{\ell_2} \wedge \\
\qquad \widehat{\mathsf{X}}(\ell) \supseteq \widehat{\mathsf{X}}(\ell_0) \cup \widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2) \wedge \\
\qquad \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{G}}(\ell_0) \cup \widehat{\mathsf{G}}_I(\ell_1) \cup \widehat{\mathsf{G}}_I(\ell_2) \wedge \\
\qquad \widehat{\mathsf{G}}(\ell) \supseteq (\widehat{\mathsf{G}}_L(\ell_1) \cup \widehat{\mathsf{G}}_L(\ell_2)) \,;\, \widehat{\mathsf{D}}(\ell_0) \wedge \\
\qquad (\bullet \in \widehat{\mathsf{G}}(\ell) \Rightarrow \widehat{\mathsf{G}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_0)) \wedge \\
\qquad \widehat{\mathsf{D}}(\ell) \supseteq \widehat{\mathsf{D}}_I(\ell_1) \cup \widehat{\mathsf{D}}_I(\ell_2) \wedge \\
\qquad \widehat{\mathsf{D}}(\ell) \supseteq (\widehat{\mathsf{D}}_L(\ell_1) \cup \widehat{\mathsf{D}}_L(\ell_2)) \,;\, \widehat{\mathsf{D}}(\ell_0) \wedge \\
\qquad \widehat{\mathsf{D}}(\ell) \supseteq (\widehat{\mathsf{X}}(\ell_1) \cup \widehat{\mathsf{X}}(\ell_2)) \times \widehat{\mathsf{V}}(\ell_0) \wedge \\
\qquad \widehat{\mathsf{V}}(\ell) \supseteq \widehat{\mathsf{V}}(\ell_0) \cup \widehat{\mathsf{V}}_I(\ell_1) \cup \widehat{\mathsf{V}}_I(\ell_2) \wedge \\
\qquad \widehat{\mathsf{V}}(\ell) \supseteq (\widehat{\mathsf{V}}_L(\ell_1) \cup \widehat{\mathsf{V}}_L(\ell_2)) \,;\, \widehat{\mathsf{D}}(\ell_0)
\end{array}
$$

Much of the recent work on security has focused on probabilistic notions, see for example [7, 5]. Danos and Harmer, [1], have introduced a probabilistic version of IA. The language is essentially extended with a new constant, `coin`. The semantics is specified with probabilistic transitions and the new rules:

$$\frac{}{\rho \vdash \texttt{coin}^{\ell}, \sigma \Downarrow_{0.5} 0, \sigma}$$

and

$$\frac{}{\rho \vdash \texttt{coin}^{\ell}, \sigma \Downarrow_{0.5} 1, \sigma}$$

We conjecture that our possibilistic analysis gives safe information about probabilistic information flow. The analysis of IA requires no extension; we just treat `coin` as any other constant.

This paper concerns the specification of information flow analyses. Efficient implementations might require some safe transformations of the constraint systems but we have not yet studied this problem.

Our ultimate goal is to apply these techniques to realistic object-oriented languages with threads and other advanced features.

## References


1. V. Danos and R. Harmer. Probabilistic game semantics. In *Proc. the Thirteenth Annual IEEE Symposium on on Logic in Computer Science*. IEEE Computer Society Press, 2000. To appear.
2. D. Denning. A lattice model of secure information flow. In *Communications of the ACM*, pages 236–243. ACM, 1976.

3. F. Nielson, H. R. Neilson, and C. L. Hankin. *Principles of Program Analysis*. Springer, 1999.
4. J. C. Reynolds. The essence of Algol. In *Proceedings of the 1981 International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, 1981.
5. A. Sabelfeld and D. Sands. Probablistic noninterference for multi-threaded programs. In *Proc. the Thirteenth IEEE Computer Security Foundations Workshop*, Cambridge, England, 2000. IEEE Computer Society Press. To appear.
6. D. Volpano and G. Smith. Eliminating covert flows with minimum typings. In *Proc. the Tenth IEEE Computer Security Foundations Workshop*, pages 156–168, Rockport, MA, 1997. IEEE Computer Society Press.
7. D. Volpano and G. Smith. Probablistic noninterference in a concurrent language. In *Proc. the Eleventh IEEE Computer Security Foundations Workshop*, pages 34–43, Rockport, MA, 1998. IEEE Computer Society Press.