Flow-policy awareness for distributed mobile code (proofs)
Draft

Ana Almeida Matos
Instituto de Telecomunicações / Instituto Superior Técnico de Lisboa
ana.matos@ist.utl.pt

July 21, 2008

Contents

Erratum

List of typos in previous versions of the paper, corrected here:

- In Proposition 1.2.8.
- Subsection 1.3.2, in definition of (Meet) Pre-semilattices of flow policies.
- In Proposition 1.3.5.

Abstract

Several programming constructs have been recently proposed with the purpose of enabling the programmer to encode declassifying information flows within a program that complies with information flow security policies. Theses constructs may or may not incorporate some means for controlling when, where, what, or by whom the declassification can be set up. In the context of global computing, other forms of controlling declassification that transcend the power of a single declassification construct may turn out to be desirable. In this paper we point out potential unwanted behaviors that can arise in a context where programs that contain declassifying instructions can migrate to computation domains with different security policies. We propose programming language design techniques for tackling such unwanted behaviors and prove soundness of those techniques, both at the local and global computation level.

Chapter 1

Flow-policy awareness for distributed mobile code

1.1 Introduction

With the emergence of the new possibilities offered by global computing, new security issues follow from the fact that they can just as well be exploited by parties with hazardous intentions. Such attacks are ubiquitous enough to make security a crucial concern. Many attacks arise at the application level, and can be tackled by means of programming language design and analysis techniques, such as static analysis and proof carrying code. For instance, confidentiality can be violated by execution of programs that reveal secret information. This kind of program behavior can be controlled using *information flow* analyzes [Sabelfeld and Myers(2003); Barthe et al.(2006)Barthe, Rezk, and Naumann], by detecting dependencies in programs that could encode flows of information from private to publicly available resources.

In the field of security, control must often be balanced with flexibility for practical reasons, since models that are prohibitively restrictive do not suit the real world-needs. In information flow research, it has been a challenging problem to find an alternative to the classical non-interference property [Goguen and Meseguer(1982)] that is flexible enough to allow for *declassification* to take place in a controlled manner [Zdancewic(2004)]. So far, most solutions have been directed towards local computation scenarios, thus overlooking decentralization issues that are inherent to distributed settings. Indeed, enforcement of confidentiality in networks must deal with distributed security policies, since different *computation domains* (or *sites*) follow different security orientations. For example, migrating programs that were conceived to comply to certain flow policies don't necessarily respect those of the computational locations they might end up executing at. This problem seems to be beyond the grasp of single declassification constructs that can restrict by whom, when, what, or where in the program declassification can be performed [Sabelfeld and Sands(2005)], since now the question is: *in which context?*

In this paper we show that the issue of enabling and controlling flexible information flow policies in computations that can spread out over locations (governed by different flow policies) can be addressed at the programming language level. We propose to remove some of the burden of restricting declassification away from the declassification instruction itself, and transfer it to new program constructions that provide awareness about the flow policy of the context in which it is running. Programmers can then be given the power to predict alternatives to the pieces of code that contain the forbidden declassification operations they would wish to use. With this new tool, it becomes realistic to write programs that can safely run under any flow policy. Furthermore, it becomes acceptable to rely on mechanisms that reject the execution of programs that, being unaware of the flow policy of the context, blindly encode disallowed declassifications.

Some security minded distributed network models have been proposed with the purpose of controlling the migration of code in between computation sites, such as by means of programmable

domains [Boudol(2005)] and type systems [Martins and Vasconcelos(2005)]. These ideas can be applied to the proof-carrying code model [Necula(1997)], since it consists of a particular instance of boundary transposition control that performs type checks to incoming code[Gorla et al.(2005)Gorla, Hennessy, and Sassone]. In order to apply migration control techniques to the problem of controlling declassification by preventing programs from migrating to sites if they would potentially violate that site's flow policies. However, we fall short on technical mechanisms that would allow, on one hand, for a site to know what are the most flexible flow policies that a program sets up for its own executions; on another hand, for a program to know how flexible is the flow policy of the context in which it is running. The main contributions of this paper are the following:

- A new programming construct (allowed F then M else N) that tests the flexibility of the *allowed flow policy* imposed by the domain where it is currently located and can act accordingly. From a local perspective, this is a first step towards programming in contexts where domains can change their allowed flow policies dynamically. From a global perspective, in the presence of code mobility, the *allowed construct* provides useful expressibility for programming alternative behaviors that a program can have, should it end up in a site where certain declassification operations are not permitted.

- A new security property we call *confinement to a flow policy*, that ensures programs will respect certain flow policies, regardless of the declassification operations they might contain. This property is formulated both at the local level, with respect to single allowed flow policies, and at the network level, by considering distributed flow policies and the location of programs at runtime.

- A new form of security effect that can be associated to a program, containing information about the declassifying environments that can potentially be established by that program. We call it the *declassification effect*, and it is flexible enough to allow programs containing operations that are forbidden at certain sites to be considered secure nevertheless, as long as these operations are protected by an appropriate *allowed* construct. We show that this information is useful when setting up a migration control mechanism for deciding whether or not certain programs should be allowed to execute at each site.

- Two type systems that can be used to enforce flow policy confinement on an expressive language, according to two different approaches. First, a type system that is suitable for static analysis of local computations, that only accepts programs that do not perform disallowed declassifications, whatever the allowed flow policy of the site they are executing at. Second, a type system for a distributed language with code mobility that hands over the decision of accepting or rejecting programs, while providing the declassification effect of the analyzed program. The latter approach can be integrated into more dynamic security frameworks for code mobility.

- The identification of a new form of *migration leaks* that can be encoded in a distributed language with code mobility by means of the new allowed construct. These do not result from memory synchronization issues, but reflect instead the new possibilities of accessing information about the location of programs in the network. We show how these migration leaks can be controlled by means of the aforementioned type systems that also enforce a flexible information flow policy, namely the Non-disclosure policy [Almeida Matos and Boudol(2005); Almeida Matos(2006)], for both a local and a distributed language with code mobility containing the new allowed construct.

This paper is structured into two main sections, corresponding to the local (Section 1.2)and global perspective (Section 1.3) on the proposed problem. In both sections, we start by defining the language (Subsections 1.2.1 and 1.3.1) and the formal security properties of Non-disclosure and Confinement (Subsections 1.2.2 and 1.3.2); then, a type system is presented, and its soundness is proved (Subsections 1.2.3 and 1.3.3). Finally we discuss related work (Section 1.4) and conclude (Section 1.5).

Variables	x, y	$\in$	$\boldsymbol{Var}$		
Reference Names	a, b, c	$\in$	$\boldsymbol{Ref}$		
Values	V	$\in$	$\boldsymbol{Val}$	$::=$	$() \mid x \mid a_{l,\theta} \mid (\lambda x.M) \mid tt \mid ff$
Pseudo-values	W	$\in$	$\boldsymbol{Pse}$	$::=$	$V \mid (\varrho x.W)$
Expressions	M, N	$\in$	$\boldsymbol{Exp}$	$::=$	$W \mid (M\ N) \mid (M; N) \mid (\text{if } M \text{ then } N_t \text{ else } N_f)$
					$(\text{ref}_{l,\theta}\ M) \mid (!\ N) \mid (M := N) \mid$
					$\textbf{(flow } \boldsymbol{F} \textbf{ in } \boldsymbol{M}\textbf{)} \mid \textbf{(allowed } \boldsymbol{F} \textbf{ then } \boldsymbol{N_t} \textbf{ else } \boldsymbol{N_f}\textbf{)}$

Figure 1.1: Syntax of Expressions

1.2 Local perspective

Here we follow the non-disclosure point of view of information flow analysis presented in [Almeida Matos and Boudol(2005); Almeida Matos(2006)]. The non-disclosure property is a generalization of non-interference. It uses information provided by the program semantics describing which flow policies are valid at different points of the computation, to ensure that, at each step, all information flows comply to the valid flow policy. In order to enforce non-disclosure, the programming language must be enriched with a flow declaration construct, that simply declares the flow policy that is valid in its scope within the program, while the semantics of the language should convey information regarding the flow policy that rules at each step.

Once the language is enriched with flow declarations, some means for controlling the usage of this construct is desirable. For instance, a computation domain might want to impose a limit to the flexibility of the flow declarations that are used within its programs. We can then define the notion of *confinement with respect to a flow policy* as a property of programs that can only perform steps that comply to that allowed flow policy. This property can be formalized by making use of the information about the flow policy that is available at each step, as provided by the semantics.

At the moment that a program is written, it might be hard to anticipate which flow policies will be imposed at computation time. In order to provide programs with some awareness regarding the flow policy that is ruling in the current computation domain, we introduce the (allowed F then M else N), a condition that tests whether the flow policy F is allowed by the policy of the context and executes branches M or N accordingly.

In order to statically analyze the potential flow policies that a program might set up, we can verify a simple rule that combines the usage of flow declarations and of allowed-conditions: Every flow declaration must be performed within the "allowed" branch of an allowed-condition that tests the declared flow policy. In this way, programs become robust with respect to all possible allowed flow policies.

1.2.1 Language

The language that we consider for the study of local computations is a an imperative higher-order λ-calculus with reference creation, where we include a flow policy declaration construct (for directly manipulating flow policies [Almeida Matos and Boudol(2005); Almeida Matos(2006)]) and the new flow policy tester construct that branches according to whether a certain flow policy is allowed in the program's computing context. Although we will consider a deterministic sequential language, the results and techniques that are used make the extension to concurrent runtime threads straightforward.

Syntax

Security annotations and types are apparent in the syntax of the language, though they do not play any role in the operational semantics (they will be used at a latter stage of the analysis). Security levels l, j, k are sets of principals, which are ranged over by $p, q \in \boldsymbol{Pri}$. They are associated to references (and reference creators), representing the set of principals that are allowed to read the information contained in each reference. We also decorate references with the type of the values that they can hold. In the following we may omit security level subscripts whenever they are not relevant. The syntax of types τ, σ, θ is given later in Subsection 1.2.3. Flow policies A, F, G are binary relation over $\boldsymbol{Pri}$. A pair $(p, q) \in F$, most often written $p \prec q$, is to be understood as "information may flow from principal p to principal q", that is, more precisely, "*everything that principal p is allowed to read may also be read by principal q*". We denote, as usual, by F^* the reflexive and transitive closure of F.

The local language of *expressions* (defined in Figure 1.1) is based on a call-by-value λ-calculus extended with the imperative constructs of ML, conditional branching and boolean values (here the $(\varrho x.W)$ construct provides for recursive values). Variables x and references a, b, c are drawn from two disjoint countable sets $\boldsymbol{Var}$ and $\boldsymbol{Ref}$, respectively. Reference names can be created at runtime. The new features are the flow declaration and the allowed-condition. The flow declaration construct is written (flow F in M), where M is executed in the context of the current flow policy *extended with* F; after termination the current flow policy is restored, that is, the scope of F is M. The allowed-condition is similar to a standard boolean condition, with the difference that in (allowed F then N_t else N_f) the branches N_t or N_f are executed according to whether or not F is allowed by the site's allowed flow policy.

The (local) evaluation relation, defined next, operates over *configurations* of the form $\langle M, S\rangle$ where: M is an expression and the memory or store $S : (\boldsymbol{Ref} \times 2^{\boldsymbol{Pri}} \times \boldsymbol{Typ}) \rightharpoonup \boldsymbol{Val}$ is a mapping from a finite set of decorated reference names to values.

Operational semantics

We now define the semantics of the language as a small step operational semantics on configurations. In defining the evaluation order, it is convenient to write expressions using *evaluation contexts*. Intuitively, expressions that are placed in such contexts are to be executed first. We write $\mathrm{E}[M]$ to denote an expression where the subexpression M is placed in the evaluation context E, obtained by replacing the occurrence of $[]$ in E by M. The evaluation contexts of the language define a call-by-value evaluation order (see Figure 1.3).

The analysis of whether the information flows that occur in M are to be allowed depends on the flow policies that are declared in the evaluation context where M is executed. We denote by $\lceil \mathrm{E} \rceil$ the flow policy that is permitted by the evaluation context E. It collects all the flow policies that are declared using flow declaration constructs into one single flow policy, using set union:

Definition 1.2.1 (Flow Policy Declared by an Evaluation Context)**.** *The* flow policy declared by the evaluation context E *is given by* $\lceil \mathrm{E} \rceil$ *where:*

$$\lceil [] \rceil = \emptyset, \qquad \lceil (\text{flow } F \text{ in } \mathrm{E}) \rceil = F \cup \lceil \mathrm{E} \rceil,$$
$$\lceil \mathrm{E}'[\mathrm{E}] \rceil = \lceil \mathrm{E} \rceil, \ \textit{if } \mathrm{E}' \textit{ does not contain flow declarations}$$

Some *basic notations and conventions* are useful for defining transitions on configurations. Given a configuration $\langle M, S\rangle$, we define $\mathrm{dom}(S)$ as the set of decorated reference names that are mapped by S. We say a reference name a is fresh in S if it does not occur, with any subscript, in $\mathrm{dom}(S)$, that is if $b_{l,\theta} \in \mathrm{dom}(S)$ implies $b \neq a$. We denote by $\mathrm{rn}(M)$ the set of decorated reference names that occur in the expression M. We let $\mathrm{fv}(M)$ be the set of variables occurring free in M. We restrict our attention to well formed configurations $\langle M, S\rangle$ satisfying the following conditions on memories: $\mathrm{rn}(M) \subseteq \mathrm{dom}(S)$; for any $a_{l,\theta} \in \mathrm{dom}(S)$ we have $\mathrm{rn}(S(a_{l,\theta})) \subseteq \mathrm{dom}(S)$, and also that all occurrences of a name in a configuration are decorated in the same way. We denote by $\{x \mapsto W\}M$ the capture-avoiding substitution of W for the free occurrences of x in M.

$$
\begin{aligned}
A \vdash\ \langle \mathrm{E}[((\lambda x.M)\ V)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[\{x \mapsto V\}M], S\rangle \\
A \vdash\ \langle \mathrm{E}[(\text{if } \mathit{tt} \text{ then } N_t \text{ else } N_f)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[N_t], S\rangle \\
A \vdash\ \langle \mathrm{E}[(\text{if } \mathit{ff} \text{ then } N_t \text{ else } N_f)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[N_f], S\rangle \\
\boldsymbol{A \vdash\ \langle \mathrm{E}[(\text{allowed } F \text{ then } N_t \text{ else } N_f)], S\rangle} &\boldsymbol{\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[N_t], S\rangle,\ \textit{where } F \subseteq A^*} \\
\boldsymbol{A \vdash\ \langle \mathrm{E}[(\text{allowed } F \text{ then } N_t \text{ else } N_f)], S\rangle} &\boldsymbol{\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[N_f], S\rangle,\ \textit{where } F \not\subseteq A^*} \\
A \vdash\ \langle \mathrm{E}[(V; N)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[N], S\rangle \\
A \vdash\ \langle \mathrm{E}[(\varrho x.W)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[(\{x \mapsto (\varrho x.W)\}\ W)], S\rangle \\
A \vdash\ \langle \mathrm{E}[(\text{flow } F \text{ in } V)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[V], S\rangle \\
\\
A \vdash\ \langle \mathrm{E}[(!\ a_{l,\theta})], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[V], S\rangle,\ \textit{where } S(a_{l,\theta}) = V \\
A \vdash\ \langle \mathrm{E}[(a_{l,\theta} := V)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[()], [a_{l,\theta} := V]S\rangle \\
A \vdash\ \langle \mathrm{E}[(\mathrm{ref}_{l,\theta}\ V)], S\rangle &\xrightarrow[\lceil \mathrm{E} \rceil]{} \langle \mathrm{E}[a_{l,\theta}], [a_{l,\theta} := V]S\rangle,\ \textit{a fresh in } S
\end{aligned}
$$

Figure 1.2: Operational Semantics

The operation of adding or updating the image of an object z to z' in a mapping Z is denoted $[z := z']Z$.

The transitions of our *small step semantics* are defined in Figure 1.2. The '$A \vdash$' turnstile makes explicit the allowed flow policy A of the site where the computations are taking place. The labeled transition rules of our semantics are decorated with the flow policy declared by the evaluation context where they are performed. Most of the transitions do not depend on the flow label F that decorates them. In particular, the evaluation of (flow F in M) simply consists in the evaluation of M, annotated with a flow policy that comprises (in the sense of set inclusion) F. The lifespan of the flow declaration terminates when the expression M that is being evaluated terminates (that is, M becomes a value). The flow policy that decorates the transition steps is used only by the rules for (allowed F then N_t else N_f), whose semantics is similar to the conditional branching, but where the choice of the branch depends on whether F is allowed to be declared or not.

One can prove that the semantics preserves the conditions for well-formedness of configurations. Furthermore, a configuration has at most one transition, up to the choice of new names.

Example The allowed flow policy A of a site restricts on the flow policies that can be set up by programs running in that site. Then, the (allowed F then M else N) construct tests whether F is allowed by A^*, and can safely set up a flow declaration for F in its "allowed" branch. A typical usage of the construct could then be:

$$
\begin{array}{lll}
(\text{allowed } \{H \prec L\} & \text{then} & (\text{flow } \{H \prec L\} \text{ in } (x_L := (!\ y_H))) \\
& \text{else} & \mathit{plan\_B})
\end{array}
\tag{1.1}
$$

1.2.2 Properties

In this section we formally define the security properties of non-disclosure, the underlying information flow policy that this work is based on, and confinement, a new security property that restricts the usage of declassification instructions. The study of confidentiality traditionally relies on a lattice of security levels [Denning(1976)], corresponding to security clearances that can be associated to information containers in the programming language. Here we will use a more general structure, that of a pre-lattice, that is sufficient and convenient for defining a dynamic flow relation that can account for runtime changes in the flow policy of a program.

$$
\begin{array}{lll}
\textit{Evaluation Contexts} & \mathrm{E} & ::= \\
 & & [] \mid (\mathrm{E}\ N) \mid (V\ \mathrm{E}) \mid (\mathrm{E}; N) \mid (\mathrm{ref}_{l,\theta}\ \mathrm{E}) \mid (!\ \mathrm{E}) \\
 & & (\mathrm{E} := N) \mid (V := \mathrm{E}) \mid (\text{if E then } N_t \text{ else } N_f) \\
 & & \textbf{(flow } \boldsymbol{F} \textbf{ in E)}
\end{array}
$$

Figure 1.3: Evaluation Contexts

Pre-lattices of security levels A pre-lattices is a preordered set $(\mathcal{L}, \preceq)$ (reflexive, transitive but not necessarily anti-symmetric) such that any two elements of $\mathcal{L}$ have a least upper-bound and a greatest lower-bound with respect to $\preceq$. We now present the concrete pre-lattices our properties are based on, and show how the derived flow relation is parameterized by the current flow policies.

Consider the concrete lattice of security levels j, k, l, partially ordered by reverse inclusion. We have seen that these levels are sets of principals $p, q \in \boldsymbol{Pri}$ representing read-access rights. We can interpret the reverse inclusion of security levels as indicating allowed flows of information: if $l_1 \supseteq l_2$ then information in a a_{l_1} may be transferred to b_{l_2}, since the principals allowed to read this value from b were already allowed to read it from a. These lattices can be customized by means of a binary relation on principals, specifically by flow polices A, F, G representing additional directions in which information is allowed to flow: a pair $(p, q) \in F$, most often be written $p \prec q$, is to be understood as "information may flow from principal p to principal q".

We now introduce the *preorder on security levels* $\preceq_F$ that is determined by the flow policy F. We use the notion of *F-upward closure* of a security level l, defined as $l \uparrow_F = \{q \mid \exists p \in l.\ p\ F^*\ q\}$. The F-upward closure of l contains all the principals that are allowed by the policy F to read the contents of a reference labeled l. We can now derive a more permissive flow relation [Myers and Liskov(1998); Almeida Matos and Boudol(2005)] and use it to define the pre-lattice that is determined by a flow policy.

$$l_1 \preceq_F l_2 \ \stackrel{\text{def}}{\Leftrightarrow}\ \forall q \in l_2\ .\ \exists p \in l_1\ :\ p\ F^*\ q\ \Leftrightarrow\ (l_1 \uparrow_F) \supseteq (l_2 \uparrow_F)$$

Since in a pre-lattice the meet and join operations are not unique, here we chose $l_1 \curlywedge_F l_2 = l_1 \cup l_2$ and $l_1 \curlyvee_F l_2 = (l_1 \uparrow_F) \cap (l_2 \uparrow_F)$. Consequently, we have $\top = \emptyset$ and $\bot = \boldsymbol{Pri}$. Notice that $\preceq_F$ extends $\supseteq$ in the sense that $\preceq_F$ is larger than $\supseteq$ and that $\preceq_\emptyset = \supseteq$.

Non-disclosure

So far we are considering a sequential setting, but we wish our results to be easily scalable to concurrent (and distributive) compositionality. We will then formulate our information flow property in terms of a bisimulation, which is convenient for non-deterministic settings since bisimulations provide a natural way of relating programs according to their behavior [Smith(2001); Boudol and Castellani(2002)]. We briefly present the definition of the non-disclosure property, and refer the reader to [Almeida Matos and Boudol(2005)] for further explanations.

We recall the definitions of low part of a memory and of low-equality of memories with respect to a flow policy F and security level l. Intuitively, two memories are said to be "low-equal" if they have the same "low-domain", and if they give the same values to all references that are labeled with "low" security levels.

Definition 1.2.2 (Low-Equality)**.** *Low-equality between memories S_1 and S_2 with respect to a flow policy F and a security level l is given by*

$$S_1 =^{F,l} S_2 \ \stackrel{def}{\Leftrightarrow}\ S_1 \restriction^{F,l} = S_2 \restriction^{F,l} \text{ , where:}$$
$$S \restriction^{F,l} \ \stackrel{def}{=}\ \{(a_{k,\theta}, V) \mid (a_{k,\theta}, V) \in S\ \&\ k \preceq_F l$$

This relation is transitive, reflexive and symmetric. We will use the notation $=^l$ whenever the flow policy is $\emptyset$.

We now present a bisimulation that relates two programs if they show the same behavior on the low part of two memories. The bisimulation is based on the small-step semantics defined in Section 1.2.1. It is defined on expressions M, rather than between configurations $\langle M, S\rangle$, in order to account for memory changes that are external to the program M when running concurrently with other threads (again, this feature is not required here, but is useful for compositionality). In the following we denote by $\twoheadrightarrow$ the reflexive closure of the union of the transitions $\underset{F}{\longrightarrow}$, for all F.

Definition 1.2.3 ($\approx_l$)**.** *An l-bisimulation is a symmetric relation $\mathcal{R}$ on expressions such that:*

$$M_1 \ \mathcal{R} \ M_2 \text{ and } A \vdash \langle M_1, S_1\rangle \underset{F}{\longrightarrow} \langle M_1', S_1'\rangle \text{ and } S_1 =^{F,l} S_2$$
$$\text{implies that } \exists M_2', S_2' \ :$$
$$A \vdash \langle M_2, S_2\rangle \twoheadrightarrow \langle M_2', S_2'\rangle \text{ and } S_1' =^{l} S_2' \text{ and } M_1' \ \mathcal{R} \ M_2'$$
$$\text{when:}$$
$$\mathrm{dom}(S_1{}') - \mathrm{dom}(S_1) \cap \mathrm{dom}(S_2) = \emptyset$$

The largest l-bisimulation[1] *is denoted by $\approx_l$.*

It is important to see that the relation $\approx_{G,l}$ is not reflexive since, as we can see from the following definition, it should only be "reflexive" with respect to secure programs. For instance, the insecure expression $(v_B := (!\ u_A))$ is not bisimilar to itself if $A \not\preceq_F B$.

The reason why the above bisimulation potentially relates more programs than one for Non-interference is the stronger premise $S_1 =^{F,l} S_2$. By starting with pairs of memories that are low-equal "to a greater extent", i.e. that coincide in a larger portion of the memory, the condition on the behavior of the program P_2 becomes weaker. The intuition is that when P_1 performs a transition within the scope of the local flow policy F, it is allowed to read "low"-references from the input memory (S_1 and S_2) according to the current flow policy F.

The bisimulation relation could have been parameterized by an additional flow policy G, which would represent the global flow policy that is assumed to hold everywhere by default. In this paper we chose to omit this parameter by fixing $G = \emptyset$, for notational clarity, though all the results can be easily extended accordingly. For simplicity of the bisimulation definition, we are also not concerned with the fact that this definition can be considered somewhat restrictive in what respects changes in references that are created at run-time.

If a program is shown to be bisimilar to itself, one can conclude that the high part of the memory has not interfered with the low part, i.e., no security leak has occurred. A secure program can then be defined as one that is related to itself by the above bisimulation.

Definition 1.2.4 (Non-disclosure)**.** *An expression M satisfies the Non-disclosure policy if it satisfies $M \approx_l M$ for all security levels l.*

Confinement to a flow policy

We now define operational confinement with respect to an allowed flow policy, and justify the chosen formalization. In light of the semantics of our flow declarations, we can predict which flow policies are declared by a program at runtime by observing the flow policy that decorates each of the possible steps it might take. Then, confinement to an allowed policy A means that every step is decorated with a flow policy F that is stricter than A:

Definition 1.2.5 (Operationally A-Confined Expressions)**.** *A set $\mathcal{C}$ of expressions is said to be a set of operationally A-confined expressions if the following holds for any $M \in \mathcal{C}$:*

$$A \vdash \langle M, S\rangle \underset{F}{\longrightarrow} \langle M', S'\rangle \text{ implies } F \subseteq A^* \text{ and } M' \in \mathcal{C}.$$

The largest set of operationally A-confined expressions[2] *is denoted by $\mathcal{C}_A$. We then say that an expression M is operationally A-confined, if $M \in \mathcal{C}_A$.*

[1] Note that for any l there is an l-bisimulation, like for instance the set $\boldsymbol{Val} \times \boldsymbol{Val}$ of pairs of values. Furthermore, the union of a family of l-bisimulations is an l-bisimulation, which is the largest l-bisimulation.

[2] The largest of these sets exists, for analogous reasons to Footnote 1.

Using notions of information flow, one could formulate a stricter property that interprets the allowed flow policy of a domain as an imposition on the information flows the the program actually sets up. Then, to say that a program computing in a domain should respect a flow policy A would mean that all information flows that are performed at every computation step should comply to the flow relation $\preceq_A$. This property can be stated using the following bisimulation:

Definition 1.2.6 ($\approx_{A,l}$)**.** *An (A,l)-bisimulation is a symmetric relation $\mathcal{R}$ on expressions such that:*

$$M_1 \ \mathcal{R} \ M_2 \text{ and } A \vdash \langle M_1, S_1\rangle \underset{F}{\longrightarrow} \langle M_1', S_1'\rangle \text{ and } S_1 =^{A,l} S_2$$
implies that $\exists M_2', S_2' \ :$
$$A \vdash \langle M_2, S_2\rangle \twoheadrightarrow \langle M_2', S_2'\rangle \text{ and } S_1' =^{l} S_2' \text{ and } M_1' \ \mathcal{R} \ M_2'$$
when:
$$\mathrm{dom}(S_1{}') - \mathrm{dom}(S_1) \cap \mathrm{dom}(S_2) = \emptyset$$

The largest (A,l)-bisimulation[3] *is denoted by $\approx_{A,l}$.*

Notice the difference between this bisimulation, and the one in Definition 1.2.3: Here the flow policies that decorate the steps are irrelevant; the low part of the memories is taken with respect to the flow policy we want flows to comply to. We can now prove that, for programs that satisfy non-disclosure, operational A-confinement implies the stronger property that is obtained from the above (A,l)-bisimulation:

Proposition 1.2.7. *If M satisfies non-disclosure, then M is operationally A-confined implies that $M \approx_{A,l} M$.*

Proof. Easy, by showing that the set

$$C_A = \{(M_1, M_2) \mid M_1, M_2 \in \mathcal{C}_A \text{ and } M_1 \approx_l M_2\}$$

is an (A,l)-bisimulation. □

To see that the converse result is not true, consider the example:

$$(\text{flow } \boldsymbol{Pri} \times \boldsymbol{Pri} \text{ in } (x_H := 0)) \tag{1.2}$$

This program satisfies non-disclosure, and is (A,l)-bisimilar to itself, for all A. However, it is not operationally confined to any flow policies B such that $B^* \neq \boldsymbol{Pri} \times \boldsymbol{Pri}$.

1.2.3 Type System

We now present a type and effect system that only accepts programs that satisfy the Non-disclosure and Confinement properties defined in Subsection 1.2.2.

As defined in Figure 1.4, the judgments of the type and effect system have the form

$$\Gamma \vdash_{A,F} M : s, \tau$$

meaning that the expression M is typable with type τ and security effect s in the typing context $\Gamma : \boldsymbol{Var} \to \boldsymbol{Typ}$, which assigns types to variables. The turnstile has two flow policies as parameters:

- the flow policy *declared by the context* F, represents the one that is valid in the evaluation context in which the expression M is typed, and contributes to the meaning of operations and relations on security levels;
- the flow policy *tested by the context* A, represents the one that has been positively tested as being allowed by the computation domain where the expression M is running.

[3]The largest (A,l)-bisimulation exists, for analogous reasons to Footnote 1.

$$[\textsc{Nil}]\ \Gamma \vdash () : \mathsf{unit} \qquad [\textsc{BoolT}]\ \Gamma \vdash tt : \mathsf{bool} \qquad [\textsc{BoolF}]\ \Gamma \vdash ff : \mathsf{bool} \qquad [\textsc{Loc}]\ \Gamma \vdash a_{l,\theta} : \theta\ \mathrm{ref}_l$$

$$[\textsc{Var}]\ \Gamma, x : \tau \vdash x : \tau \qquad [\textsc{Abs}]\ \frac{\Gamma, x : \tau \vdash_{A,F} M : s, \sigma}{\Gamma \vdash (\lambda x.M) : \tau \xrightarrow[A,F]{s} \sigma} \qquad [\textsc{Rec}]\ \frac{\Gamma, x : \tau \vdash_{A,F} W : s, \tau}{\Gamma \vdash (\varrho x.W) : \tau}$$

$$[\textsc{Flow}]\ \frac{\boldsymbol{\Gamma \vdash_{A,F \cup F'} N : s, \tau} \quad \boldsymbol{F' \subseteq A^*}}{\boldsymbol{\Gamma \vdash_{A,F} (\mathbf{flow}\ F'\ \mathbf{in}\ N) : s, \tau}}$$

$$[\textsc{Allow}]\ \frac{\begin{array}{c}\boldsymbol{\Gamma \vdash_{A \cup F',F} N_t : s_t, \tau} \\ \boldsymbol{\Gamma \vdash_{A,F} N_f : s_f, \tau}\end{array}}{\boldsymbol{\Gamma \vdash_{A,F} (\mathbf{allowed}\ F'\ \mathbf{then}\ N_t\ \mathbf{else}\ N_f) : s_t \curlyvee s_f, \tau}}$$

$$[\textsc{Ref}]\ \frac{\Gamma \vdash_{A,F} M : s, \theta \quad s.r \preceq_F l}{\Gamma \vdash_{A,F} (\mathrm{ref}_{l,\theta}\ M) : s \curlyvee \langle \bot, l, \bot \rangle, \theta\ \mathrm{ref}_l} \qquad [\textsc{Der}]\ \frac{\Gamma \vdash_{A,F} M : s, \theta\ \mathrm{ref}_l}{\Gamma \vdash_{A,F} (!\ M) : s \curlyvee \langle l, \top, \bot \rangle, \theta}$$

$$[\textsc{Ass}]\ \frac{\Gamma \vdash_{A,F} M : s, \theta\ \mathrm{ref}_l \quad \Gamma \vdash_{A,F} N : s', \theta \qquad \begin{array}{r} s.t \preceq_F s'.w \\ s.r, s'.r \preceq_F l \end{array}}{\Gamma \vdash_{A,F} (M := N) : s \curlyvee s' \curlyvee \langle \bot, l, \bot \rangle, \mathsf{unit}}$$

$$[\textsc{Cond}]\ \frac{\Gamma \vdash_{A,F} M : s, \mathsf{bool} \qquad \begin{array}{c} \Gamma \vdash_{A,F} N_t : s_t, \tau \\ \Gamma \vdash_{A,F} N_f : s_f, \tau \end{array} \qquad s.r \preceq_F s_t.w, s_f.w}{\Gamma \vdash_{A,F} (\mathrm{if}\ M\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f) : s \curlyvee s_t \curlyvee s_f \curlyvee \langle \bot, \top, s.r \rangle, \tau}$$

$$[\textsc{Seq}]\ \frac{\Gamma \vdash_{A,F} M : s, \tau \quad \Gamma \vdash_{A,F} N : s', \sigma \quad s.t \preceq_F s'.w}{\Gamma \vdash_{A,F} (M; N) : s \curlyvee s', \sigma}$$

$$[\textsc{App}]\ \frac{\Gamma \vdash_{A,F} M : s, \tau \xrightarrow[F]{s'} \sigma \quad \Gamma \vdash_{A,F} N : s'', \tau \qquad \begin{array}{r} s.t \preceq_F s''.w \\ s.r, s''.r \preceq_F s'.w \end{array}}{\Gamma \vdash_{A,F} (M\ N) : s \curlyvee s' \curlyvee s'' \curlyvee \langle \bot, \top, s.r \curlyvee s''.r \rangle, \sigma}$$

Figure 1.4: Type and Effect System

The security effect s is composed of three security levels that are referred to by $s.r$, $s.w$ and $s.t$, and can be understood as follows: $s.r$ is the *reading effect*, an upper-bound on the security levels of the references that are read by M; $s.w$ is the *writing effect*, a lower bound on the references that are written by M; $s.t$ is the *termination effect*, an upper bound on the level of the references on which the termination of expression M might depend. According to these intuitions, in the type system the reading and termination levels are composed in a covariant way, whereas the writing level is contravariant.

Types have the following syntax (t is a type variable):

$$\tau, \sigma, \theta \ \in \ \boldsymbol{Typ} \ ::= \ t \mid \mathsf{unit} \mid \mathsf{bool} \mid \theta \ \mathrm{ref}_l \mid \tau \xrightarrow[A,F]{s} \sigma$$

Typable expressions that reduce to $()$ have type unit, and those that reduce to booleans have type bool. Typable expressions that reduce to a reference which points to values of type θ and has security level l have the reference type $\theta \ \mathrm{ref}_l$. The security level l is used to determine the effects of expressions that handle references. Expressions that reduce to a function that takes a parameter of type τ, that returns an expression of type σ, and with a *latent effect* s [Lucassen and Gifford(1988)] have the function type $\tau \xrightarrow[A,F]{s} \sigma$. The latent effect is the security effect of the body of the function, while the latent flow policies are those that are assumed to hold when the function is applied to an argument.

We use a (join) pre-semilattice on security effects, that is obtained from the pointwise composition of the pre-lattices of the security effects. More precisely:

$$s \preceq_F s' \ \stackrel{\text{def}}{\Leftrightarrow} \ s.r \preceq_F s'.r \ \& \ s'.w \preceq_F s.w \ \& s.t \preceq_F s'.t$$
$$s \curlyvee_F s' \ \stackrel{\text{def}}{\Leftrightarrow} \ \langle s.r \curlyvee_F s'.r, s.w \curlywedge_F s'.w, s.t \curlyvee_F s'.t \rangle$$
$$\top = \langle \emptyset, \boldsymbol{Pri}, \emptyset \rangle \qquad \bot = \langle \boldsymbol{Pri}, \emptyset, \boldsymbol{Pri} \rangle$$

We use some abbreviations to alleviate the notation of the typing judgments and operations, namely we write $\Gamma \vdash M : \tau$ when $\Gamma \vdash_{A,F} M : \langle \bot, \top, \bot \rangle, \tau$, which is mainly used when typing (pseudo-)values, and we write $\boldsymbol{s} \curlyvee \boldsymbol{s'}$ when $\boldsymbol{s} \curlyvee_{\emptyset} \boldsymbol{s'}$, which is used when constructing the security effects of the typed expressions.

Our type and effect system applies restrictions to programs in order to enforce two main properties:

- Compliance of all information flows to the flow relation that is parameterized with the current flow policy. This is achieved by conditions of the kind "$\preceq_F$" in the premises of the typing rules, and by the update of the security effects in the conclusions. Apart from the parameterization of the flow relation with the current flow policy, these are fairly standard in information flow type system and enforce syntactic rules of the kind "no low writes should depend on high reads", both with respect to the values that are read, and to termination behaviors that might be derived. Notice that the FLOW rule types the body of the flow declaration under a more permissive flow policy. We refer the reader to [Almeida Matos(2006)] for explanations on all of these conditions.

- Confinement of all flow declarations of a policy F to be performed only once F has been tested to be positively allowed by the domain's allowed flow policy. This is achieved by means of the tested allowed flow policy A that parameterizes the typing judgments, and by the condition $F' \subseteq A^*$ in the FLOW rule. Notice that the ALLOW rule types the "allowed" branch of the condition under an extended allowed flow policy.

Soundness

We start by stating that the type of an expression is preserved by reduction, while its effects may "weaken". When an expression performs a computation step, some of its effects may be performed by reading, updating or creating a reference, and some may be discarded when a branch in a conditional expression is taken.

Proposition 1.2.8 (Subject Reduction). *Consider an expression M for which there exist Γ, F, s and τ such that $\Gamma \vdash_{A,F} M : s, \tau$. Then, if $\hat{A} \vdash \langle M, S\rangle \xrightarrow[F']{} \langle M', S'\rangle$, there exists s' such that $\Gamma \vdash_{A \cup A', F} M' : s', \tau$, and where $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $A' \subseteq \hat{A}$.*

Proof. See Appendix A.1. □

We will now verify the soundness of the proposed static analysis techniques, with respect to the properties we are interested here: the non-disclosure property, where we must check that our type system appropriately constrains the usage of the new constructs introduced in this language in order to prevent them from encoding information leaks, and the confinement property, where we must check that all flow declarations are "protected" by an appropriate allowed construct.

Non-disclosure Security of expressions with respect to Non-disclosure is guaranteed by the type system, as is formalized by the following result:

Theorem 1.2.9 (Soundness of Typing Non-disclosure). *If M is an expression for which there exist Γ, s and τ such that $\Gamma \vdash_{\emptyset,\emptyset} M : s, \tau$, then M satisfies the Non-disclosure policy.*

Proof. See Appendix A.2. □

Confinement to a flow policy Our type system also guarantees security of expressions with respect to Confinement, as is formalized by the following result:

Theorem 1.2.10 (Soundness of Typing Confinement). *If M is an expression for which there exist Γ, s and τ such that $\Gamma \vdash_{\emptyset,\emptyset} M : s, \tau$, then M is operationally A-confined, for all A.*

Proof. See Appendix A.3. □

1.3 Global perspective

As was observed in the beginning of the previous section, it is not possible to statically know what is the allowed flow policy of the computation site at execution time. In a distributed context with code mobility, the problem becomes more acute, since the computation site might change *during* computation, along with the allowed flow policy that the program must comply to. A type system for static analysis as the one presented in the previous section turns out not to be enough to ensure that global computations always comply to the currently valid allowed flow policy. Therefore, we now propose a new type system that is to be used at runtime to compute information about the declassification behaviors of programs. This information will be used by the semantics of the language in order to control migration of programs according to the following rule: programs can only migrate to a site if their declassification behaviors comply to that site's flow policy.

The idea of the type system is simple: it gathers information about all the flow policy declarations that are encoded in the program into a new *declassification effect*. This information can be used to decided whether or not a certain program should be allowed to execute at a certain site. Since our language contains the new allowed construct, the programmer can increase the chances that the program will be allowed to migrate to other sites by protecting flow declarations with the new construct, in the same way that as in the local setting. Our type system does take that effort into account when building the declassification effect, by hiding the tested flow policy from the declassification effect of the "allowed" branch. As a result, programs containing flow declarations that are too permissive might still be authorized to execute in a certain domain, as long as they occur in the "not allowed" branch of our new construct, since as we know (by its semantics), it will never be executed.

In order to illustrate the usage of the techniques presented in the previous section, we will now lift the main results to the network level. To this end, we define a distributed language with code mobility, based on the one of Section 1.2.1, and formulate the network level version of the

Thread Names	m, n	$\in$	$\boldsymbol{Nam}$
Domain Names	d	$\in$	$\boldsymbol{Dom}$
Expressions	M, N	$\in$	$\boldsymbol{Exp}$
		$::=$	$\cdots \mid (\text{thread}_l\ M) \mid (\text{goto}\ d)$
Threads		$::=$	$M^{m_j} \quad (\in \boldsymbol{Exp} \times \boldsymbol{Nam} \times 2^{\boldsymbol{Pri}})$
Networks	X, Y	$::=$	$d[\ A, P, S\ ] \mid X \parallel Y$

Figure 1.5: Syntax of Expressions and Configurations

information flow and confinement properties. We will see that the new local-level type system, together with a simple membrane control mechanism at the level of each computation domain, is enough to ensure the desired properties.

1.3.1 Language

The distributed language that we define now is a straightforward generalization of the language in Section 1.2.1, obtained by adding a notion of computing domain, to which we associate an allowed flow policy, and a code migration primitive. Programs computing in different domains are subjected to different allowed flow policies – this is what distinguishes local computations from global computations, and is the main novelty in this language. We opt for a rather simplistic memory model, assuming memory to be shared by all programs and every computation domain, in a transparent form. As we will see in Section 1.3.2, this will allow us to avoid synchronization issues that are not central to this work (and that are already handled in previous work [Almeida Matos(2005)]).

Syntax

The syntax of the language of global computations is an extension of the one defined in Figure 1.1, an imperative higher-order λ-calculus with reference creation that includes a flow policy declaration construct and the new allowed-condition construct, to which we now add the notion of computation domain, its allowed flow policy, thread creation and a migration primitive. The additional syntax is defined in Figure 1.5.

There are two new kinds of names, given to threads (m, n) and to domains (d), each drawn from two new disjoint countable sets $\boldsymbol{Dom} \neq \emptyset$ and $\boldsymbol{Nam}$. A security level is associated to each thread, and appears as a subscript of thread names. This level can be understood as the set of principals that are allowed to know about the location of the thread in the network. As for references, in the following we may omit subscripts whenever they are not relevant.

The language of expressions now includes a thread creator and a migration primitive. The thread creator $(\text{thread}_l\ M)$ spawns the thread M, to which the security level l is given, and returns $()$; the new thread is to be executed concurrently. The migration construct $(\text{goto}\ d)$, where d is a domain name is also added to the language. The meaning is that the thread that executes the migration operation should migrate to the domain d.

Networks are flat juxtapositions of domains, each containing a store and a pool of threads, which are subjected to the flow policy of the domain. Threads run concurrently in pools $P : (\boldsymbol{Nam} \times 2^{\boldsymbol{Pri}}) \rightarrow \boldsymbol{Exp}$, which are mappings from decorated thread names to expressions (they can also be seen as sets of threads). Stores $S : (\boldsymbol{Ref} \times 2^{\boldsymbol{Pri}} \times \boldsymbol{Typ}) \rightarrow \boldsymbol{Val}$, as in previous section, map decorated reference names to values. Notice that networks are in fact just a collection of references, threads that are running in parallel, and the flow policies allowed by each domain. To keep track of the locations of threads it suffices to maintain a mapping from thread names to domain names. This is the purpose of the *position-tracker* $T : (\boldsymbol{Nam} \times 2^{\boldsymbol{Pri}}) \rightarrow \boldsymbol{Dom}$, which is a mapping from a finite set of decorated thread names to domain names. In turn, the flow policies that are allowed

$$\vdots$$

$$\langle W, \{(\text{allowed } F \text{ then } N_t \text{ else } N_f)^{m_j}\}, T, S\rangle \xrightarrow[F]{\emptyset} \langle W, \{{N_t}^{m_j}\}, T, S'\rangle, \quad \textit{where } F \subseteq W(T(m_j))$$

$$\langle W, \{(\text{allowed } F \text{ then } N_t \text{ else } N_f)^{m_j}\}, T, S\rangle \xrightarrow[F]{\emptyset} \langle W, \{{N_f}^{m_j}\}, T, S'\rangle, \quad \textit{where } F \not\subseteq W(T(m_j))$$

$$\frac{\boldsymbol{n} \textit{ fresh in } \mathbf{T}}{\langle \boldsymbol{W}, \{\mathbf{E}[(\mathbf{thread}_{\boldsymbol{l}}\ \boldsymbol{N})]^{\boldsymbol{m_j}}\}, \boldsymbol{T}, \boldsymbol{S}\rangle \xrightarrow[\lceil \mathbf{E} \rceil]{\boldsymbol{N^{n_k}}} \langle \boldsymbol{W}, \{\mathbf{E}[()]^{\boldsymbol{m_j}}\}, [\boldsymbol{n_k} := \boldsymbol{T}(\boldsymbol{m_j})]\boldsymbol{T}, \boldsymbol{S}\rangle}$$

$$\frac{\boldsymbol{\Gamma} \vdash^{\boldsymbol{j}}_{\emptyset} \mathbf{E}[()] : \boldsymbol{s}, \boldsymbol{\tau} \quad \boldsymbol{s.d} \subseteq \boldsymbol{W(d)^*}}{\langle \boldsymbol{W}, \{\mathbf{E}[(\mathbf{goto}\ \boldsymbol{d})]^{\boldsymbol{m_j}}\}, \boldsymbol{T}, \boldsymbol{S}\rangle \xrightarrow[\lceil \mathbf{E} \rceil]{\emptyset} \langle \boldsymbol{W}, \{\mathbf{E}[()]^{\boldsymbol{m_j}}\}, [\boldsymbol{m_j} := \boldsymbol{d}]\boldsymbol{T}, \boldsymbol{S}\rangle}$$

$$\frac{\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{\emptyset} \langle W, \{M'^{m_j}\}, T', S'\rangle}{\langle W, \{M^{m_j}\}, T, S\rangle \underset{F}{\longrightarrow} \langle W, \{M'^{m_j}\}, T', S'\rangle} \quad \frac{\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle}{\langle W, \{M^{m_j}\}, T, S\rangle \underset{F}{\longrightarrow} \langle W, \{M'^{m_j}, N^{n_k}\}, T', S'\rangle}$$

$$\frac{\langle W, P, T, S\rangle \underset{F}{\longrightarrow} \langle W, P', T', S'\rangle \quad \langle W, P \cup Q, T, S\rangle \textit{ is well formed}}{\langle W, P \cup Q, T, S\rangle \underset{F}{\longrightarrow} \langle W, P' \cup Q, T', S'\rangle}$$

Figure 1.6: Operational Semantics

by each domain can be kept using a mapping $W : \boldsymbol{Dom} \to \boldsymbol{Pri} \times \boldsymbol{Pri}$ from domain names to flow policies. These two sets, together with the pool P containing all the threads in the network, and the store S containing all the references in the network, form *configurations* $\langle W, P, T, S\rangle$, over which the evaluation relation is defined in the next subsection. We then obtain a configuration of the form $\langle W, P, T, S\rangle$ from a network $d_1[\ A_1, P_1, S_1\ ] \parallel \cdots \parallel d_n[\ A_n, P_n, S_n\ ]$, where:

$W = \{d_1 \mapsto A_1, \ldots, d_n \mapsto A_n\}$,

$T = \{m_j \mapsto d_1 | M^{m_j} \in P_1\} \cup \cdots \cup \{m_j \mapsto d_n | M^{m_j} \in P_n\}$,

$P = P_1 \cup \cdots \cup P_n$, and $S = S_1 \cup \cdots \cup S_n$.

Operational semantics

The (small step) semantics of the language that we consider for the study of global computations is an extension of the one defined in Subsection 1.2.1. Its most distinctive ingredient is the rule for the migration instruction, which depends on the type system. We postpone explanations on the meaning of the conditions in this rule to Subsection 1.3.3, where the type system is explained.

We add to the basic notations and conventions of the previous section the sets $\mathrm{dom}(W)$, $\mathrm{dom}(P)$ and $\mathrm{dom}(T)$, the sets of decorated names of threads and references that are mapped by W, P and T, respectively. We say that a thread or reference name is fresh in T or S if it does not occur, with any subscript, in $\mathrm{dom}(T)$ or $\mathrm{dom}(S)$, respectively. We denote by $\mathrm{tn}(P)$ and $\mathrm{rn}(P)$ the set of decorated thread and reference names, respectively, that occur in the expressions of P (this notation is extended in the obvious way to expressions). As for the local language, we restrict our attention to *well formed configurations* $\langle W, P, T, S\rangle$ satisfying the following additional conditions for memories, values stored in memories, and thread names: $\mathrm{rn}(P) \subseteq \mathrm{dom}(S)$; $\mathrm{dom}(P) \subseteq \mathrm{dom}(W) \cap \mathrm{dom}(T)$;

$\mathrm{tn}(\mathrm{dom}(S)) \subseteq \mathrm{dom}(T)$, all threads in a configuration have distinct names, and also all occurrences of a name in a configuration are decorated in the same way. We denote by $\{x \mapsto W\}M$ the capture avoiding substitution of W for the free occurrences of x in M.

We start by defining the transitions of a single thread. These are decorated with the thread N^{n_k} that is possibly spawned during that transition, where $N^{n_k} = ()$ if no thread is created. The evaluation of single local expressions remains unchanged with respect to the ones in Figures 1.3 and 1.2, where rules of the form

$$A \vdash \langle M, S\rangle \underset{F}{\longrightarrow} \langle M', S'\rangle$$

are rewritten as

$$\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{()} \langle W, \{M'^{m_j}\}, T, S'\rangle$$

assuming that $W(m) = A$ and that the new network configuration is well formed. The additional semantics is defined in Figure 1.6, including the rules for the allowed construct. Given a configuration $\langle W, P, T, S\rangle$, we call the pair (T, S) the *state* of the configurations. Notice that the W component of configurations never changes.

Thread names are used in three situations: When a new thread is created, its new fresh name is added to the position-tracker, associated to the parent domain. When the (goto d) statement is executed by a thread m, the position of m in the position-tracker is updated to d. The execution of the statement depends however on the typing of the thread that wishes to migrate, and on the allowed flow policy of the destination site. Thread names are also used when an allowed-condition is performed: the tested flow policy is compared to the allowed flow policy of the site where that particular thread is executing.

The last three rules use the information contained in the label to add any spawned threads to the pool of threads. By the last rule we can see that the execution of a pool of threads is compositional. The last three rules use the information contained in the label to add any spawned threads to the pool of threads. By the last rule we can see that the execution of a pool of threads is compositional.

One can prove that the semantics preserves for well-formedness of configurations, and that a configuration with a single thread has at most one transition, up to the choice of new names.

1.3.2 Properties

We will now generalize the security properties of non-disclosure and confinement that were presented in Subsection 1.2.2 by adapting them to our network setting. The security analysis in this section is also based on security levels, so we will use a pre-lattice of security levels similar to the one in the previous section; furthermore, we will deal with relations between flow policies, which leads us to define a pre-lattice of flow policies.

(Meet) Pre-semilattices of flow policies We introduce the *preorder on flow policies* $\preceq$, thus overloading the notation for the flow relations on security levels. The meaning of relating two flow policies as in $F_1 \preceq F_2$ is that F_1 is more permissive than F_2. A flow policy F_1 is considered "more permissive" than F_2 if F_1 encodes all the information flows that are enabled by F_2. Formally, we have that

$$F_1 \preceq F_2 \overset{\text{def}}{\Leftrightarrow} F_2 \subseteq F_1^*$$

Where the meet operation is given by $F_1 \curlywedge F_2 = F_1 \cup F_2$. Consequently, we have $\top = \emptyset$.

Non-disclosure

Now that we have defined our concurrent language, we can generalize the non-disclosure definition of Subsection 1.2.2 in order to reflect the fact that we are dealing with pools of threads instead of single expressions. The notion of low-equality is similar to the one in Subsection 1.2.2. However, as we will see towards the end of this section, the position of a thread in the network can reveal

information about the values in the memory. For this reason, we must use a notion of low-equality that is extended to states.

Definition 1.3.1 (Low-Equality)**.** *The low-equality between states $\langle T_1, S_1\rangle$ and $\langle T_2, S_2\rangle$ with respect to a flow policy F and a security level l is given by*

$$\langle T_1, S_1\rangle =^{F,l} \langle T_2, S_2\rangle \stackrel{def}{\Leftrightarrow} T_1 \restriction^{F,l} = T_2 \restriction^{F,l} \text{ and } S_1 \restriction^{F,l} = S_2 \restriction^{F,l}$$

where:

$$T \restriction^{F,l} \stackrel{def}{=} \{(n_k, d) \mid (n_k, d) \in T \ \&\ k \preceq_F l\}$$
$$S \restriction^{F,l} \stackrel{def}{=} \{(a_{k,\theta}, V) \mid (a_{k,\theta}, V) \in S \ \&\ k \preceq_F l\}$$

This relation is also transitive, reflexive and symmetric.

Now we define a bisimulation for networks, which can be used to relate networks with the same behavior over low parts of the states. In the following we denote by $\twoheadrightarrow$ the reflexive closure of the union of the transitions $\underset{F}{\longrightarrow}$, for all F.

Definition 1.3.2 ($\approx_l$)**.** *An l-bisimulation is a symmetric relation $\mathcal{R}$ on sets of threads such that:*

$$\begin{array}{l} P_1 \ \mathcal{R}\ P_2 \text{ and } \langle W, P_1, T_1, S_1\rangle \underset{F}{\longrightarrow} \langle W, P_1', T_1', S_1'\rangle \text{ and} \\ \langle T_1, S_1\rangle =^{F,l} \langle T_2, S_2\rangle \\ \text{implies} \\ \exists P_2', T_2', S_2' \,.\, \langle W, P_2, T_2, S_2\rangle \twoheadrightarrow \langle W, P_2', T_2', S_2'\rangle \text{ and} \\ \langle T_1', S_1'\rangle =^{l} \langle T_2', S_2'\rangle \text{ and } P_1' \ \mathcal{R}\ P_2' \\ \text{when} \\ \mathrm{dom}(S_1{}') - \mathrm{dom}(S_1) \cap \mathrm{dom}(S_2) = \emptyset \text{ and:} \\ \mathrm{dom}(T_1{}') - \mathrm{dom}(T_1) \cap \mathrm{dom}(T_2) = \emptyset \end{array}$$

The largest l-bisimulation[4] *is denoted by $\approx_l$.*

Intuitively, our security property must state that, at each computation step performed by some thread in a network, the information flow that occurs respects the flow policy (F) that is declared by the context where the command is executed.

Definition 1.3.3 (Non-disclosure for Networks)**.** *A pool of threads P satisfies the Non-disclosure for Networks policy if it satisfies $P \approx_l P$ for all security levels l.*

Non-disclosure for Networks differs from that of Definition 1.2.4 in that the position of the "low threads" is also treated as "low-information".

Examples We are considering a simplistic memory model where all of the network's memory is accessible at all times by every process in the network. With this assumption we avoid *migration leaks* that derive from synchronization behaviors [Almeida Matos(2005)]. In our setting, we avoid these issues, but migration leaks can be encoded nonetheless. The idea is that now a program can reveal information about the position of a thread in a network by performing tests on the flow policy that is allowed by that site:

$$\begin{array}{l} (\text{if } (!\ x_H) \text{ then } (\text{goto } d_1) \text{ else } (\text{goto } d_2))\ ; \\ (\text{allowed } F \text{ then } (y_L := 1) \text{ else } (y_L := 2)) \end{array} \tag{1.3}$$

In this example, the thread will migrate to domains d_1 or d_2 depending on the tested high value; then, if these domains have different allowed flow policies, different low-assignments are performed, thus revealing high level information. Therefore, the program is insecure with respect to Non-disclosure for Networks.

[4]The largest l-bisimulation exists, for analogous reasons to Footnote 1.

Confinement

In Subsection 1.2.2 we motivated our notion of an operationally A-confined expression, as one that, when running at a computation domain with an allowed flow policy A, only sets up flow policies that comply to A, i.e., it's information flows never violate A. We also saw that our type system ensures confinement with respect to any allowed flow policy. In a setting where code can migrate between domains with different allowed security policies, enforcement of a confinement property becomes more subtle. Consider for example the following program:

$$(\text{allowed } F \text{ then } ((\text{goto } d_1); (\text{flow } F \text{ in } M_1)) \text{ else } M_2) \tag{1.4}$$

In this program, the flow declaration of the policy F is executed only if F has been tested as being allowed by domain the program is located at. In the sense of the type system of the previous section, we can say that the flow declaration is "protected" by an appropriate allowed construct. However, by the time the flow declaration is performed, the thread is already located at another site, where that flow policy might not hold. This is why, to start with, the semantics of the code migration rule involves information about the allowed flow policy of the destination site. More significantly, as we will see in the next section, the type system we will use to enforce confinement for networks will give more refined information about the potential declassifications performed by a program.

Since the allowed flow policy that programs should comply to during global computations is not fixed, here the notion of operational confinement is not parameterized by a single flow policy. Instead, the relation is set up on triples that carry information about the location of the site and its allowed flow policy, which is used to place a restriction on the flow policies that decorate the transitions: at any step, they should comply to the flow policy of the site that the program who performed that step is located at.

Definition 1.3.4 (Operationally Confined Threads)**.** *A set $\mathcal{C}$ of triples $\langle A, d, M^{m_j}\rangle$ is said o be a set of* operationally confined threads *if the following holds for any $\langle A, d, M^{m_j}\rangle \in \mathcal{C}$:*

$$\begin{array}{l}\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle \text{ implies}\\ F \subseteq A^* \text{ and } \langle A', T'(m_j), M'^{m_j}\rangle, \langle A, T'(n_k), N^{n_k}\rangle \in \mathcal{C}\\ \text{when:}\\ T(m_j) = d \text{ and } W(d) = A \text{ and } W(T'(m_j)) = A'\end{array}$$

The largest set of operationally confined threads[5] *is denoted by $\mathcal{C}$. We then say that a thread $\langle A, d, M^{m_j}\rangle$ is operationally confined, if $\langle A, d, M^{m_j}\rangle \in \mathcal{C}$.*

From the above definition of operational confinement of individual threads, we can derive a notion of network confinement by obtaining from a well formed network $\langle W, P, T\rangle$ the set:

$$\begin{array}{l}\text{trp}(\langle W, P, T\rangle) =\\ \quad \{\langle A, d, M^{m_j}\rangle \mid M^{m_j} \in P \text{ and } T(m_j) = d \text{ and } W(d) = A\}\end{array}$$

A network is said to be operationally confined if all of the triples in its corresponding set are operationally confined in the above sense.

1.3.3 Type system

We now present a type and effect system that accepts programs that satisfy Non-disclosure for Networks, and that constructs a declassification effect that can be used to enforce Confinement (these properties are defined in Subsection 1.3.2).

The typing judgments used in Figure 1.7 have the same meaning as in Subsection 1.2.3, with one extra parameter:

$$\Gamma \vdash^{j}_{F} M : s, \tau$$

[5]The largest of these sets exists, for analogous reasons to Footnote 1.

$$[\textsc{Nil}]\ \Gamma \vdash () : \mathsf{unit} \qquad [\textsc{BoolT}]\ \Gamma \vdash tt : \mathsf{bool} \qquad [\textsc{BoolF}]\ \Gamma \vdash ff : \mathsf{bool} \qquad [\textsc{Loc}]\ \Gamma \vdash a_{l,\theta} : \theta\ \mathrm{ref}_l$$

$$[\textsc{Var}]\ \Gamma, x : \tau \vdash x : \tau \qquad [\textsc{Abs}]\ \frac{\Gamma, x : \tau \vdash^j_F M : s, \sigma}{\Gamma \vdash (\lambda x.M) : \tau \xrightarrow[F,j]{s} \sigma} \qquad [\textsc{Rec}]\ \frac{\Gamma, x : \tau \vdash^j_{A,F} W : s, \tau}{\Gamma \vdash (\varrho x.W) : \tau}$$

$$[\textsc{Flow}]\ \frac{\boldsymbol{\Gamma \vdash^j_{F \cup F'} N : s, \tau}}{\boldsymbol{\Gamma \vdash^j_F (\mathbf{flow}\ F'\ \mathbf{in}\ N) : s + F', \tau}}$$

$$[\textsc{Allow}]\ \frac{\begin{array}{c}\boldsymbol{\Gamma \vdash^j_F N_t : s_t, \tau} \\ \boldsymbol{\Gamma \vdash^j_F N_f : s_f, \tau}\end{array} \quad \boldsymbol{j \preceq_F s_t.w, s_f.w}}{\boldsymbol{\Gamma \vdash^j_F (\mathbf{allowed}\ F'\ \mathbf{then}\ N_t\ \mathbf{else}\ N_f) : s_t - F' \curlyvee s_f \curlyvee \langle \bot, \top, j, \top \rangle, \tau}}$$

$$[\textsc{Ref}]\ \frac{\Gamma \vdash^j_F M : s, \theta \quad s.r, s.t \preceq_F l}{\Gamma \vdash^j_F (\mathrm{ref}_{l,\theta}\ M) : s \curlyvee \langle \bot, l, \bot, \top \rangle, \theta\ \mathrm{ref}_l} \qquad [\textsc{Der}]\ \frac{\Gamma \vdash^j_F M : s, \theta\ \mathrm{ref}_l}{\Gamma \vdash^j_F (!\ M) : s \curlyvee \langle l, \top, \bot, \top \rangle, \theta}$$

$$[\textsc{Ass}]\ \frac{\Gamma \vdash^j_F M : s, \theta\ \mathrm{ref}_l \quad \Gamma \vdash^j_F N : s', \theta \qquad \begin{array}{r} s.t \preceq_F s'.w \\ s.r, s'.r, s.t, s'.t \preceq_F l \end{array}}{\Gamma \vdash^j_F (M := N) : s \curlyvee s' \curlyvee \langle \bot, l, \bot, \top \rangle, \mathsf{unit}}$$

$$[\textsc{Cond}]\ \frac{\Gamma \vdash^j_F M : s, \mathsf{bool} \quad \begin{array}{c}\Gamma \vdash^j_F N_t : s_t, \tau \\ \Gamma \vdash^j_F N_f : s_f, \tau\end{array} \quad s.r, s.t \preceq_F s_t.w, s_f.w}{\Gamma \vdash^j_F (\mathrm{if}\ M\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f) : s \curlyvee s_t \curlyvee s_f \curlyvee \langle \bot, \top, s.r, \top \rangle, \tau}$$

$$[\textsc{Seq}]\ \frac{\Gamma \vdash^j_F M : s, \tau \quad \Gamma \vdash^j_F N : s', \sigma \quad s.t \preceq_F s'.w}{\Gamma \vdash^j_F (M; N) : s \curlyvee s', \sigma}$$

$$[\textsc{App}]\ \frac{\Gamma \vdash^j_F M : s, \tau \xrightarrow[F,j]{s'} \sigma \quad \Gamma \vdash^j_F N : s'', \tau \qquad \begin{array}{r} s.t \preceq_F s''.w \\ s.r, s''.r, s.t, s''.t \preceq_F s'.w \end{array}}{\Gamma \vdash^j_F (M\ N) : s \curlyvee s' \curlyvee s'' \curlyvee \langle \bot, \top, s.r \curlyvee s''.r, \top \rangle, \sigma}$$

$$[\textsc{Thr}]\ \frac{\boldsymbol{j \preceq_F l} \qquad \boldsymbol{\Gamma \vdash^l_\emptyset M : s, \mathsf{unit}}}{\boldsymbol{\Gamma \vdash^j_F (\mathbf{thread}_l\ M) : \langle \bot, j \curlyvee s.w, \bot, s.d \rangle, \mathsf{unit}}} \qquad [\textsc{Mig}]\ \boldsymbol{\Gamma \vdash^j_F (\mathbf{goto}\ d) : \langle \bot, j, \bot, \top \rangle, \mathsf{unit}}$$

Figure 1.7: Type and Effect System

The security level j represents the confidentiality level associated to the thread that the expression M is part of; more precisely, it is the confidentiality level of the location of that thread in the network. The security effect s now includes a fourth component, a flow policy that is denoted $s.d$, which corresponds to the *declassification effect*: a lower bound to the flow policies that are declared in the typed expression. The declassification effect is composed in a contravariant in the type system. The syntax of types is similar to the ones used in the previous section, the only difference being the omission of the latent allowed flow policy from the type of expressions that reduce to functions; the latent security level j of the thread containing the expression appears instead. The (join) pre-semilattice on security effects is extended in the obvious with with the declassification effect. As in the previous section we abbreviate $\Gamma \vdash M : \tau$ by $\Gamma \vdash^{j}_{F} M : \langle \bot, \top, \bot, \top \rangle, \tau$, and $\boldsymbol{s} \curlyvee \boldsymbol{s'}$ by $\boldsymbol{s} \curlyvee_{\emptyset} \boldsymbol{s'}$. Furthermore, we write '$s - F$' for $\langle s.r, s.w, w.t, s.d - F \rangle$ and '$s + F$' for $\langle s.r, s.w, w.t, s.d \cup F \rangle$.

As we have mentioned, the declassification effect should give information about the potential flow policy environments that are set up by the program. The effect is constructed by aggregating (by union) all the flow policies that are declared within the program. However, when a part of the program is "protected" by an (allowed F then M else N) construct, some of the information in the declassification effect can be discarded.

Similarly to the type and effect system of Figure 1.4, the one in Figure 1.7 analyzes programs according to two main properties.

- Compliance of all information flows to the flow relation that is parameterized with the current flow policy. This is achieved in a similar manner to the type system of the previous section by conditions of the kind "$\preceq_F$". There are, however, extra conditions that are introduced in order to deal with new forms of migration leaks that appear in our distributed setting (such as Example 1.3), and that deserve further attention: the security level j that is associated to each thread, and represents the confidentiality level of the position of the thread in the network is used to update the writing effect in the migration rule, and the termination effect in the allowed-condition rule, while it is constrained not to "precede low writes" in rule Allow. The extra condition regarding the termination effect (such as in rules Ass, Cond and App) are in fact, they are implicit in the previous type system as well. Here they must appear explicitly since it is no longer true that $s.t \preceq_F s.r$ for any F, due to the update of the termination effect with the level j in rule Allow.

- Accurate construction of the declassification effect of the program, which should allow to test confinement of all flow declarations with respect to any allowed flow policy. The new effect is updated in rule Flow, each time a flow declaration is performed, and "grows" as the declassification effects of subterms are met (by union) in order to form that of the parent command. An exception appears in rule Allow, where the declassification effect of the "allowed" branch is not used entirely: the part that has been tested by the allowed-condition is omitted. The intuition is that the part that is removed (say, F) is of no concern since in practice the allowed branch will only be executed if F is allowed.

Soundness

We check that the type of a thread is preserved by reduction, while its effects "weaken". This result generalizes the one Proposition 1.3.5, with an additional statement on type and effects of the potentially created thread. The declassification effect may become less permissive as computations within flow declarations are completed, or when branches in conditional expressions are taken.

Proposition 1.3.5 (Subject Reduction)**.** *Consider a thread M^{m_j} for which there exist Γ, F, s and τ such that $\Gamma \vdash^{j}_{F} M : s, \tau$. Then, if $\langle W, \{M^{m_j}\}, T, S \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S' \rangle$, there exists s' such that $\Gamma \vdash^{j}_{F} M' : s', \tau$, and where $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq s'.d$. Furthermore, $\exists s''$ such that $\Gamma \vdash^{k}_{\emptyset} N : s'', \mathsf{unit}$ and $s.w \preceq s''.w$ and $s.d \cup W(T(n_k)) \preceq s''.d$.*

Proof. See Appendix B.1. □

Non-disclosure for networks Our soundness result for non-disclosure is compositional, in the sense that it is enough to verify the typability of each thread separately in order to ensure non-disclosure for the whole network:

Theorem 1.3.6 (Soundness for Non-disclosure for Networks.)**.**
Consider a pool of threads P. If for all $M^{m_j} \in P$ there exist Γ, s and τ such that $\Gamma \vdash^{j}_{\emptyset,\emptyset} M : s, \tau$, then P satisfies the Non-disclosure for Networks policy.

Proof. See Appendix B.2. □

Confinement We will now see that the declassification effect can be used for enforcing operational confinement: if a program is typable with a declassification effect $s.d$, and if it performs an execution step that is decorated with a flow policy F, then F is contained in $s.d$, meaning that the actual flow policy that was ruling for that step is stricter than the declassification effect.

Proposition 1.3.7 (Meaning of the declassification effect)**.**
Consider a thread M^{m_j} for which there exist Γ, G, s and τ such that $\Gamma \vdash^{j}_{G} M : s, \tau$. If we have $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$, then $F \subseteq s.d$.

Proof. See PropositionB.3.1 in Appendix B.3. □

If a program M is typable with a declassification effect G, then its steps are decorated with flows that are allowed by G. This means that a site can trust that declassifications performed by an incoming thread are not more permissive than what is declared in the type.

Theorem 1.3.8 (Soundness of Typing Confinement for Networks)**.** *Consider a well formed network $\langle W, P, T\rangle$ such that for all $M^{m_j} \in P$ there exist Γ, s and τ such that $\Gamma \vdash^{j}_{G} M : s, \tau$ and $s.d \subseteq W(T(m_j))^*$. Then the set of triples $\mathrm{trp}(\langle W, P, T\rangle)$ is a set of operationally confined threads.*

Proof. See Appendix B.3. □

The above result might seem somewhat surprising at first, given that in the typing rule of the allowed construct, the flow policy that is being tested is subtracted from the declassification effect of the allowed branch. Notice however that the allowed branch will only be taken by the operational semantics, if the tested flow policy is allowed (by G) in the first place. This means that the part of the declassification effect of the allowed branch that is omitted is known to be allowed by G. To illustrate this idea, consider again Example 1.1, which has an empty declassification effect (and is therefore syntactically G-confined, for all G; it's transitions can however be decorated with the flow policy F in case the first branch is taken, which in turn can only happen if $F \subseteq G^*$.

1.3.4 Comparison of the type systems

When using the type system to construct a declassification effect, we provide a way to build a certificate for the program, that can be analyzed at any time and place to conclude about whether a program should be allowed to execute or not. In case the certificate is not trusted, programs could also be statically checked to be "flow declaration safe". The relation between the "rejecting" type system of the previous section, and the "declassification effect" type system of the current section is clarified by the following result.

Proposition 1.3.9. *Consider an expression M in the local language of the previous section. Then, if there exist Γ, F, τ such that $\Gamma \vdash_{A,F} M : s, \tau$, then we have $\Gamma \vdash^{j}_{F} M : s', \tau$, with $s'.d \subseteq A^*$.*

Proof. By induction on the inference of $\Gamma \vdash_{A,F} M : s, \sigma$, and by case analysis on the last rule used in this typing proof. The only rule in which the declassification effect is updated with a flow policy F in the second type system is Flow, in which cases F is always allowed by A in the first type system. □

1.4 Related work

Information flow and distribution Mantel and Sabelfeld [Mantel and Sabelfeld(2003)] provide a type system for preserving confidentiality for different kinds of channels established over a publicly observable medium in a distributed setting, but where interaction between domains is restricted to the exchange of values (no code mobility). Sharing our underlying aim of studying the distribution of code under decentralized security policies, Zdancewic *et al.* [Zdancewic et al.(2002)Zdancewic, Zheng, Nystrom, and Myers] study the secure partitioning of programs into a distributed system of potentially corrupted hosts and of principals. Castagna, Bugliesi and Craffa study non-interference for a purely functional distributed and mobile calculus [Crafa et al.(2002)Crafa, Bugliesia, and Castagna]; where no declassification mechanisms are contemplated. The work that is closest to this one is [Almeida Matos(2005); Almeida Matos(2006)], which studies insecure information flows that are introduced by mobility in the context of a distributed language with states. Declassification is present in the language by means of flow declarations. In the computation model that is considered, threads own references that move along with them during migration; this gives rise to *migration leaks*.

Controlling declassification In a recent survey [Sabelfeld and Sands(2005)] examine the literature regarding the subject of declassification, and observe that declassification can be controlled according to four main orthogonal goals as to: *what* information should be released [Sabelfeld and Myers(2004); Li and Zdancewic(2005)], *when* it should be allowed to happen [Chong and Myers(2004)], *who* should be authorized to use it [Myers et al.(2006)Myers, Sabelfeld, and Zdancewic], and *where* in the program it can be stated [Broberg and Sands(2006); Almeida Matos and Boudol(2005)]; these dimensions can also be combined [Barthe et al.(2008)Barthe, Cavadini, and Rezk]. Most of the overviewed approaches implicitly assume local settings, where the computation platform enforces fixed policies. Furthermore, the tools that are given to the programmer for controlling the usage of declassification operations are restricted to the declassifying operations themselves. In this sense, we can say that they provide for ways of controlling declassification *from within*, as opposed to the techniques that are proposed in this paper: both the allowed-construct and restricted versions of migration instructions are external to the flow declaration construct. In this sense, this paper advocates for technique for restricting declassification *from without*.

Combining access control and declassification could be seen as another form of controlling declassification from without. The idea is to make sure that a program can only declassify information that it has the right to read. In [Boudol and Kolundzija(2007)], standard access control primitives are used to control the access level of programs that perform declassifications. In particular, the test l then M else$N$ instruction, that tests whether the access level l is granted by the context, bears resemblance with our allowed-condition construct.

1.5 Conclusions and Future work

The issue of controlling information flow in global computing is attracting increased attention (e.g. the Mobius IST-15905 Project). Here we have motivated the need for controlling the usage of declassifying instructions in a global computing context by pointing out that programs that were conceived to comply to certain flow policies don't necessarily respect those of the computational locations they might end up executing at. We have addressed this issue, by providing techniques for ensuring that a thread can only migrate to a site if it complies to its allowed flow policy.

One could also mention the dual problem, that information that is carried by programs into sites with more permissive flow policies becomes vulnerable. In order to tackle this problem, one could consider a model where references can move along with threads [Almeida Matos(2005); Almeida Matos(2006)]. We leave this research direction for future work. Nevertheless, we believe that the allowed-condition construct that was introduced here can play an important role in the solution, since it enables threads to inspect the allowed flow policy of a site, according to which they can decide whether to remain there or to migrate away.

When considering a network model with code mobility and restricted accesses to remote references, memory synchronization issues can lead to migration leaks as was shown in [Almeida Matos(2005)]. When considering a strictly distributed memory model (where accesses to remote references are restricted), memory synchronization issues can lead to migration leaks as was shown in [Almeida Matos(2005)]. However, this paper shows that migration leaks do not exclusively depend on the memory model. In fact, even while assuming transparent remote accesses to references, a new form of migration leaks appear as a result of introducing our new program construction for inspecting the site's flow policies. This motivates a better understanding of migration leaks in global computations.

Appendix A

Local Perspective (proofs)

A.1 Subject Reduction

In order to establish the soundness of the type system of Figure 1.4 we need a Subject Reduction result, stating that types that are given to expressions are preserved by computation. To prove it we follow the usual steps [Wright and Felleisen(1994)].

Remark A.1.1. *If $W \in \boldsymbol{Pse}$ and $\Gamma \vdash_{A,F} W : s, \tau$, then for all flow policies A', F', we have that $\Gamma \vdash_{A',F'} W : \langle \bot, \top, \bot \rangle, \tau$.*

Lemma A.1.2.

1. *If $\Gamma \vdash_{A,F} M : s, \tau$ and $x \notin \mathrm{dom}(\Gamma)$ then $\Gamma, x : \sigma \vdash_{A,F} M : s, \tau$.*
2. *If $\Gamma, x : \sigma \vdash_{A,F} M : s, \tau$ and $x \notin \mathrm{fv}(M)$ then $\Gamma \vdash_{A,F} M : s, \tau$.*

Proof. By induction on the inference of the type judgment. □

Lemma A.1.3 (Substitution)**.**
If $\Gamma, x : \sigma \vdash_{A,F} M : s, \tau$ and $\Gamma \vdash W : \sigma$ then $\Gamma \vdash_{A,F} \{x \mapsto W\}M : s, \tau$.

Proof. By induction on the inference of $\Gamma, x : \tau \vdash_{A,F} M : s, \sigma$, and by case analysis on the last rule used in this typing proof, using the previous lemma. □

Lemma A.1.4 (Replacement)**.**
If $\Gamma \vdash_{A,F} \mathrm{E}[M] : s, \tau$ is a valid judgment, then the proof gives M a typing $\Gamma \vdash_{A \cup A', F \cup \lceil \mathrm{E} \rceil} M : \bar{s}, \bar{\tau}$ for some $\bar{s}$ and $\bar{\tau}$ such that $\bar{s}.r \preceq s.r$, $s.w \preceq \bar{s}.w$ and $\bar{s}.t \preceq s.t$.In this case, if $\Gamma \vdash_{A \cup A' \cup A'', F \cup \lceil \mathrm{E} \rceil} N : \bar{s}', \bar{\tau}$ with $\bar{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \bar{s}'.w$ and $\bar{s}'.t \preceq \bar{s}.t$, then $\Gamma \vdash_{A \cup A'', F} \mathrm{E}[N] : s', \tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

Proof. By induction on the structure of E. □

Proposition A.1.5 (Subject Reduction)**.** *Consider an expression M for which there exist Γ, F, s and τ such that $\Gamma \vdash_{A,F} M : s, \tau$. Then, if $\hat{A} \vdash \langle M, S \rangle \underset{F'}{\longrightarrow} \langle M', S' \rangle$, there exists s' such that $\Gamma \vdash_{A \cup A', F} M' : s', \tau$, and where $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $A' \subseteq \hat{A}$.*

Proof. Suppose that $M = \bar{\mathrm{E}}[\bar{M}]$ and that $\langle \bar{M}, S \rangle \underset{\bar{F}}{\longrightarrow} \langle \bar{M}', \bar{S}' \rangle$. We start by observing that this implies $F' = \bar{F} \cup \lceil \bar{\mathrm{E}} \rceil$, $M' = \bar{\mathrm{E}}[\bar{M}']$ and $\bar{S}' = S'$. We can assume, without loss of generality, that $\bar{M}$ is the smallest in the sense that there is no $\hat{\mathrm{E}}, \hat{M}$ such that $\hat{\mathrm{E}} \neq []$ and $\hat{\mathrm{E}}[\hat{M}] = \bar{M}$ for which we can write $\langle \hat{M}, S \rangle \underset{\hat{F}}{\longrightarrow} \langle \hat{M}', S' \rangle$.

By Lemma A.1.4, we have $\Gamma \vdash_{A \cup \bar{A}, F \cup \lceil \bar{\mathrm{E}} \rceil} \bar{M} : \bar{s}, \bar{\tau}$ in the proof of $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}] : s, \tau$, for some $\bar{s}$ and $\bar{\tau}$. We proceed by case analysis on the transition $\langle \bar{M}, S \rangle \underset{\bar{F}}{\longrightarrow} \langle \bar{M}', S' \rangle$, and prove that $\Gamma \vdash_{A \cup \bar{A}, F \cup \lceil \bar{\mathrm{E}} \rceil} \bar{M}' : \bar{s}', \bar{\tau}$, for some $\bar{s}'$ such that $\bar{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \bar{s}'.w$ and $\bar{s}'.t \preceq \bar{s}.t$.

$\bar{M} = ((\lambda x.\hat{M})\ V)$. Here we have $\bar{M}' = \{x \mapsto V\}\hat{M}$. By rule App, we have $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} (\lambda x.\hat{M}) : \hat{s},\hat{\tau} \xrightarrow[F\cup\lceil\bar{\mathrm{E}}\rceil]{\hat{s}'} \hat{\sigma}$ and $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} V : \hat{s}'',\hat{\tau}$, where $\hat{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \hat{s}'.w$ and $\hat{s}'.t \preceq \bar{s}.t$. and $\bar{s}.d \preceq_A \hat{s}'.d$. By Abs, then $\Gamma, x : \hat{\tau} \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} \hat{M} : \hat{s}',\hat{\sigma}$, and by Remark A.1.1 we have $\Gamma \vdash V : \hat{\tau}$. Therefore, by Lemma A.1.3, we get $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} \{x \mapsto V\}\hat{M} : \hat{s}',\hat{\sigma}$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (\mathbf{if}\ tt\ \mathbf{then}\ N_t\ \mathbf{else}\ N_f)$. Here we have $\bar{M}' = N_t$. By Cond, we have $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} N_t : s_t,\bar{\tau}$, where $s_t.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_t.w$ and $s_t.t \preceq \bar{s}.t$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (\mathbf{ref}_{l,\theta}\ V)$. Here we have $\bar{M}' = a_{l,\theta}$. By Loc, we have $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} a : \langle\bot,\top,\bot\rangle, \theta\ \mathrm{ref}_l$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$ and $\bot \preceq s.t$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (!\ a_{l,\theta})$. Here we have $\bar{M}' = S(a_{l,\theta})$. By assumption, we have that $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} S(a_{l,\theta}) : \langle\bot,\top,\bot\rangle,\theta$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$ and $\bot \preceq s.t$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (\mathbf{flow}\ F'\ \mathbf{in}\ V)$. Here we have $\bar{M}' = V$. By rule Flow, we have that $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil\cup F'} V : \hat{s}',\tau$ and by Remark A.1.1, we have $\Gamma \vdash_{A\cup\bar{A},F\cup\lceil\bar{\mathrm{E}}\rceil} V : \langle\bot,\top,\bot\rangle,\bar{\tau}$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$ and $\bot \preceq s.t$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (\mathbf{allowed}\ F'\ \mathbf{then}\ N_t\ \mathbf{else}\ N_f)$ **and** $F' \subseteq \hat{A}^*$. Here we have $\bar{M}' = N_t$. By Allow, we have that $\Gamma \vdash_{A\cup\bar{A}\cup F',F\cup\lceil\bar{\mathrm{E}}\rceil} N_t : s_t,\bar{\tau}$, where $s_t.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_t.w$ and $s_t.t \preceq \bar{s}.t$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A\cup F',F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

$\bar{M} = (\mathbf{allowed}\ F'\ \mathbf{then}\ N_t\ \mathbf{else}\ N_f)$ **and** $F' \not\subseteq \hat{A}^*$. Here we have $\bar{M}' = N_f$. By Allow, we have that $\Gamma \vdash_{A,F\cup\lceil\bar{\mathrm{E}}\rceil} N_f : s_f,\bar{\tau}$, where $s_f.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_f.w$, $s_f.t \preceq \bar{s}.t$ and $A \subseteq A\cup\bar{A}$. By Lemma A.1.4, we can conclude that $\Gamma \vdash_{A,F} \bar{\mathrm{E}}[\bar{M}'] : s',\tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$ and $s'.t \preceq s.t$.

The proof for the case $\bar{M} = (\varrho x.W)$ is analogous to the one for $\bar{M} = ((\lambda x.\hat{M})\ V)$, while the proofs for the cases $\bar{M} = (\mathrm{if}\ ff\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f)$ and $\bar{M} = (V;\hat{M})$ are analogous to the one for $\bar{M} = (\mathrm{if}\ tt\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f)$, and the ones for $\bar{M} = (a_{l,\theta} := V)$ is analogous to the one for $\bar{M} = (\mathrm{ref}_{l,\theta}\ V)$. □

A.2 Non-disclosure

We now present the main steps for proving soundness of the type system of Figure 1.4 with respect to the notion of security of Definition 1.2.4.

A.2.1 Basic Properties

Properties of the Semantics One can prove that the semantics preserves the conditions for well-formedness, and that a configuration with a single expression has at most one transition, up to the choice of new names.

The following result states that, if the evaluation of an expression M differs in the context of two distinct states while not creating two distinct reference names, this is because either M is performing a dereferencing operation.

Lemma A.2.1 (Splitting Computations)**.**

If we have $A \vdash \langle M, S_1\rangle \underset{F}{\longrightarrow} \langle {M_1}', S_1'\rangle$ *and also* $A \vdash \langle M, S_2\rangle \xrightarrow[F']{{N'}^{n_k}} \langle {M_2}', S_2'\rangle$ *with* ${M_1}' \neq {M_2}'$ *and* $\mathrm{dom}(S_2' - S_2) = \mathrm{dom}(S_1' - S_1)$*, then there exist* E *and* $a_{l,\theta}$ *such that* $F = \lceil \mathrm{E} \rceil = F'$*,* $M = \mathrm{E}[(!\ a_{l,\theta})]$*, and* $M' = \mathrm{E}[S_1(a_{l,\theta})]$*,* $M'' = \mathrm{E}[S_2(a_{l,\theta})]$ *with* $S_1' = S_1$ *and* $S_2' = S_2$*.*

Proof. Note that the only rule where the state is involved that for $\mathrm{E}[(!\ a_{l,\theta})]$. By case analysis on the transition $\langle M, S_1\rangle \underset{F}{\longrightarrow} \langle {M_1}', S_1'\rangle$. □

Effects

Lemma A.2.2 (Update of Effects)**.**

1. *If* $\Gamma \vdash_{A,F} \mathrm{E}[(!\ a_{l,\theta})] : s, \tau$ *then* $l \preceq s.r$*.*
2. *If* $\Gamma \vdash_{A,F} \mathrm{E}[(a_{l,\theta} := V)] : s, \tau$*, then* $s.w \preceq l$*.*
3. *If* $\Gamma \vdash_{A,F} \mathrm{E}[(\mathrm{ref}_{l,\theta}\ V)] : s, \tau$*, then* $s.w \preceq l$*.*

Proof. By induction on the structure of E. □

High Expressions We can identify a class of expressions that have the property of never performing any change in the "low" part of the memory. These are classified as being "high" according to their behavior:

Definition A.2.3 (Operationally High Expressions)**.** *A set* $\mathcal{H}$ *of threads is said to be a set of operationally* (F, l)*-high threads if the following holds for any* $M \in \mathcal{H}$*:*

$$\langle M, S\rangle \underset{F'}{\longrightarrow} \langle M', S'\rangle \text{ implies}$$

$$\langle T, S\rangle =^{F,l} \langle T', S'\rangle \text{ and both } M', N \in \mathcal{H}$$

The largest set of operationally (F, l)*-high expressions is denoted by* $\mathcal{H}_{F,l}$*. We then say that an expressions* M *is operationally* (F, l)*-high, if* $M \in \mathcal{H}_{F,l}$*.*

Note that for any F and l there exists a set of operationally (F, l)-high expressions, like for instance $\{V \mid V \in \boldsymbol{Val}\}$. Furthermore, the union of a family of sets of operationally (F, l)-high expressions is a set of operationally (F, l)-high expressions. Notice that if $F' \subseteq F$, then any operationally (F, l)-high thread is also operationally (F', l)-high.

Some expressions can be easily classified as "high" by the type system, which only considers their syntax. These cannot perform changes to the "low" memory simply because their code does not contain any instruction that could perform them. Since the writing effect is intended to be a lower bound to the level of the references that the expression can create or assign to, expressions with a high writing effect can be said to be *syntactically high*:

Definition A.2.4 (Syntactically High Expressions)**.** *An expression* M *is* syntactically (F, l)-high *if there exists* Γ, A, s, τ *such that* $\Gamma \vdash_{A,F} M : s, \tau$ *with* $s.w \not\preceq_F l$*. The expression* M *is a* syntactically (F, l)-high function *if there exists* Γ, s, τ *such that* $\Gamma \vdash M : \tau \xrightarrow[A,F]{s} \sigma$ *with* $s.w \not\preceq_F l$*.*

We can now state that syntactically high expressions have an operationally high behavior.

Lemma A.2.5 (High Expressions)**.** *If* M *is a syntactically* (F, l)*-high expression, then* M *is an operationally* (F, l)*-high expression.*

Proof. We show that if M is syntactically (F, l)-high, that is if there exists Γ, s, τ such that $\Gamma \vdash_{A,F} M : s, \tau$ with $s.w \not\preceq_F l$, and if $\langle M, S\rangle \underset{F'}{\longrightarrow} \langle M', S'\rangle$ then $S' =^{F,l} S$. This is enough since, by Subject Reduction (Theorem A.1.5), M' is syntactically (F, l)-high. We proceed by cases on the proof of the transition $\langle M, S\rangle \underset{F'}{\longrightarrow} \langle M', S'\rangle$. The lemma is trivial in all the cases where $S = S'$.

$\boldsymbol{M = \mathrm{E}[(a_{\bar{l},\bar{\theta}} := V)]}$. Here $S' = [a_{\bar{l},\bar{\theta}} := V]S$ and also $s.w \preceq \bar{l}$ by Lemma A.2.2. This implies $\bar{l} \not\preceq_F l$, hence $S' =^{F,l} S$.

The proof of the case $M = \mathrm{E}[(\mathrm{ref}_{l,\theta}\ V)]$ is analogous to the proof for $M = \mathrm{E}[(a_{l,\theta} := V)]$. □

A.2.2 Behavior of "Low"-Terminating Expressions

Recall that, according to the intended meaning of the termination effect, the termination or non-termination of expressions with low termination effect should only depend on the low part of the state. In other words, two computations of a same thread running under two "low"-equal states should either both terminate or both diverge. In particular, this implies that termination-behavior of these expressions cannot be used to leak "high" information when composed with other expressions (via termination leaks).

Lemma A.2.6 (Guaranteed Transitions)**.** *If $\langle M, S_1\rangle \underset{F}{\longrightarrow} \langle M_1', S_1'\rangle$ such that a is fresh for S_2 if $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$ and for some F' we have $S_1 =^{F\cup F',low} S_2$, then there exist M_2' and S_2' such that $\langle M, S_2\rangle \underset{F}{\longrightarrow} \langle M_2', S_2'\rangle$ with $S_1' =^{F\cup F',low} S_2'$.*

Proof. By case analysis on the proof of $\langle M, S_1\rangle \underset{F}{\longrightarrow} \langle M_1', S_1'\rangle$. In most cases, this transition does not modify or depend on the state S_1, and we may let $M_2' = M_1'$ and $S_2' = S_2$.

$\boldsymbol{M = \mathrm{E}[(\mathbf{ref}_{l,\theta}\ V)]}$. Here $M' = \mathrm{E}[a_{l,\theta}]$, $F = \lceil \mathrm{E} \rceil$ and $S_1' = S_1 \cup \{a_{l,\theta} \mapsto V\}$. Since a is fresh for S_2, we also have that $\langle M, S_2\rangle \underset{F}{\longrightarrow} \langle M_1', S_2' \cup \{a_{l,\theta} \mapsto V\}\rangle$.

$\boldsymbol{M = \mathrm{E}[(!\ a_{l,\theta})]}$. Here $M' = \mathrm{E}[S_1(a_{l,\theta})]$, $F = \lceil \mathrm{E} \rceil$ and $S_1' = S_1$. We have $\langle M, S_2\rangle \underset{F}{\longrightarrow} \langle \mathrm{E}[S_2(a_{l,\theta})], S_2\rangle$.

$\boldsymbol{M = \mathrm{E}[(a_{l,\theta} := V)]}$. then $M' = \mathrm{E}[()]$, $F = \lceil \mathrm{E} \rceil$ and $S_1' = [a_{l,\theta} := V]S_1$. We have $\langle M, S_2\rangle \xrightarrow[F]{N^{\bar{n}}\bar{k}} \langle \mathrm{E}[()], [a_{l,\theta} := V]S_2\rangle$.

□

We aim at proving that any typable expression M that has a low-termination effect always presents the same behavior according to a *strong* bisimulation on low-equal states: if two continuations M_1 and M_2 of M are related, and if M_1 can perform an execution step over a certain state, then M_2 can perform the same low changes to any low-equal state in precisely one step, while the two resulting continuations are still related. This implies that any two computations of M under low-equal states should have the same "length", and in particular they are either both finite or both infinite. To this end, we design a reflexive binary relation on expressions with low-termination effects that is closed under the transitions of Guaranteed Transitions (Lemma A.2.6).

The definition of $\mathcal{T}_{F,low}$ is given in Figure A.1. Notice that it is a symmetric relation. In order to ensure that expressions that are related by $\mathcal{T}_{F,low}$ perform the same changes to the low memory, its definition requires that the references that are created or written using (potentially) different values are high.

Remark A.2.8. *If for some F and low we have $M_1\ \mathcal{T}_{F,low}\ M_2$ and $M_1 \in \boldsymbol{Val}$, then $M_2 \in \boldsymbol{Val}$.*

From the following lemma one can conclude that the relation $\mathcal{T}_{F,low}$ relates the possible outcomes of expressions that are typable with a low termination effect, and that perform a high read over low-equal memories.

Lemma A.2.9. *If there exist Γ, s, τ such that $\Gamma \vdash_{A,F} \mathrm{E}[(!\ a_{l,\theta})] : s, \tau$ with $s.t \preceq_F low$ and $l \not\preceq_{F\cup\lceil \mathrm{E} \rceil} low$, then for any values $V_0, V_1 \in \boldsymbol{Val}$ such that $\Gamma \vdash V_i : \theta$ we have $\mathrm{E}[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}[V_1]$.*

Proof. By induction on the structure of E.

Definition A.2.7 ($\mathcal{T}_{F,low}$)**.** *We have that* $M_1\ \mathcal{T}_{F,low}\ M_2$ *if* $\Gamma \vdash_{A,F} M_1 : s_1, \tau$ *and* $\Gamma \vdash_{A,F} M_2 : s_2, \tau$ *for some* Γ*,* A*,* s_1*,* s_2 *and* τ *with* $s_1.t \preceq_F low$ *and* $s_2.t \preceq_F low$ *and one of the following holds:*

Clause 1. M_1 *and* M_2 *are both values, or*

Clause 2. $M_1 = M_2$*, or*

Clause 3. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *where* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$*, or*

Clause 4. $M_1 = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ *and* $M_2 = (\mathrm{ref}_{l,\theta}\ \bar{M}_2)$ *where* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$*, and* $l \not\preceq_F low$*, or*

Clause 5. $M_1 = (!\ \bar{M}_1)$ *and* $M_2 = (!\ \bar{M}_2)$ *where* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$*, or*

Clause 6. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$*, and* $\bar{N}_1\ \mathcal{T}_{F,low}\ \bar{N}_2$*, and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \mathrm{ref}_l$ *for some* θ *and* l *such that* $l \not\preceq_F low$*, or*

Clause 7. $M_1 = (\text{flow } F' \text{ in } \bar{M}_1)$ *and* $M_2 = (\text{flow } F' \text{ in } \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{T}_{F\cup F',low}\ \bar{M}_2$*.*

Figure A.1: The relation $\mathcal{T}_{F,low}$

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ a_{l,\theta})$. We have $V_0\ \mathcal{T}_{F,low}\ V_1$ by Clause 1.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]\ \boldsymbol{M})$. By App we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau} \xrightarrow[A,F]{s'} \bar{\sigma}$ with $\bar{s}.r \preceq s.t.$ By Lemma A.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\boldsymbol{V}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule App we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}'', \bar{\tau}$ with $\bar{s}''.r \preceq s.t.$ By Lemma A.2.2, we have $l \preceq \bar{s}''.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{if}\ \mathbf{E_1}[(!\ a_{l,\theta})]\ \mathbf{then}\ \boldsymbol{M_t}\ \mathbf{else}\ \boldsymbol{M_f})$. By Cond we have that $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \mathsf{bool}$ with $\bar{s}.r \preceq s.t.$ By Lemma A.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]; \boldsymbol{M})$. By Seq we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.t \preceq_F s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma A.1.4 and Clause 3 we can conclude.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{ref}_{l',\theta'}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Ref we have that $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.r = s.r \preceq_F l'$ and $\bar{s}.t = s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma A.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F low$. Therefore, $l' \not\preceq_F low$, and we conclude by Lemma A.1.4 and Clause 4.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ \mathbf{E_1}[a_{l,\theta}])$. By rule Der we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.t \preceq_F s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. We conclude by Lemma A.1.4 and Clause 5.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})] := \boldsymbol{M})$. By rule Ass we have that $\Gamma \vdash_{A,F} \mathrm{E}_1[a_{l,\theta}] : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l}}$ with $\bar{s}.t \preceq_F s.t$ and $\bar{s}.r \preceq_F \bar{l}$. Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. On the other hand, by Clause 2 we have $M\ \mathcal{T}_{F,low}\ M$. By Lemma A.2.2 we have $l \preceq \bar{s}.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F low$, since otherwise $l \preceq_{F\cup \mathrm{E}} low$. Therefore, we conclude by Lemma A.1.4 and Clause 6.

$\mathbf{E}[(!\ a_{l,\theta})] = (V := \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Ass we have that $\Gamma \vdash_{A,F} V : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l}}$, and $\Gamma \vdash_{A,F} \mathrm{E}_1[a_{l,\theta}] : \bar{s}', \theta$ with $\bar{s}'.t \preceq_F s.t$ and $\bar{s}'.r \preceq_F \bar{l}$. Therefore $\bar{s}'.t \preceq_F \mathit{low}$, and since $l \not\preceq_{F \cup \mathrm{E}} \mathit{low}$ implies $l \not\preceq_{F \cup \mathrm{E}_1} \mathit{low}$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,\mathit{low}}\ \mathrm{E}_1[V_1]$. On the other hand, by Clause 2 we have $V\ \mathcal{T}_{F,\mathit{low}}\ V$. By Lemma A.2.2 we have $l \preceq \bar{s}'.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F \mathit{low}$, since otherwise $l \preceq_{F \cup \mathrm{E}} \mathit{low}$. We then conclude by Lemma A.1.4 and Clause 6.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{flow}\ F'\ \mathbf{in}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Flow we have $\Gamma \vdash_{A,F \cup F'} V : s, \tau$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F \cup F',\mathit{low}}\ \mathrm{E}_1[V_1]$, so we conclude by Lemma A.1.4 and Clause 7.

□

We can now prove that $\mathcal{T}_{F,\mathit{low}}$ behaves as a kind of "strong bisimulation":

Proposition A.2.10 (Strong Bisimulation for Low-Termination)**.**
If we have $M_1\ \mathcal{T}_{F,\mathit{low}}\ M_2$ and also $\langle M_1, S_1\rangle \xrightarrow[F']{} \langle M_1', S_1'\rangle$, with $S_1 =^{F \cup F', \mathit{low}} S_2$ such that a is fresh for S_2 if $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$, then there exist S_2' such that $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle M_2', S_2'\rangle$ with $M_1'\ \mathcal{T}_{F,\mathit{low}}\ M_2'$ and $S_1' =^{F,\mathit{low}} S_2'$.

Proof. In the following, we use Subject Reduction (Theorem A.1.5) to guarantee that the termination effect of the expressions resulting from M_1 and M_2 is still low with respect to *low* and F. This, as well as typability (with the same type) for *low* and F, is a requirement for being in the $\mathcal{T}_{F,\mathit{low}}$ relation.

Clause 1. This case is not possible.

Clause 2. Here $M_1 = M_2$. By Guaranteed Transitions (Lemma A.2.6) there exist M_2' and S_2' such that $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle M_2', S_2'\rangle$ with $S_1' =^{F \cup F', \mathit{low}} S_2'$.

- **$M_2' = M_1'$.** Then we have $M_1'\ \mathcal{T}_{F,\mathit{low}}\ M_2'$, by Clause 2 and Subject Reduction (Theorem A.1.5).
- **$M_2' \neq M_1'$.** Then by Splitting Computations (Lemma A.2.1) there exist E and $a_{l,\theta}$ such that $M_1' = \mathrm{E}[S_1(a_{l,\theta})]$, $F' = \lceil \mathrm{E} \rceil$, $M_2' = \mathrm{E}[S_2(a_{l,\theta})]$, $S_1' = S_1$ and $S_2' = S_2$. Since $S_1(a_{l,\theta}) \neq S_2(a_{l,\theta})$, we have $l \not\preceq_{F \cup F'} \mathit{low}$. Therefore, $M_1'\ \mathcal{T}_{F,\mathit{low}}\ M_2'$, by Lemma A.2.9 above.

Clause 3. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ where $\bar{M}_1\ \mathcal{T}_{F,\mathit{low}}\ \bar{M}_2$. Then either:

- **$\bar{M}_1$ can compute.** In this case $M_1' = (\bar{M}_1'; \bar{N})$ with $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use the induction hypothesis, Clause 3 and Subject Reduction (Theorem A.1.5) to conclude.
- **$\bar{M}_1$ is a value.** In this case $M_1' = \bar{N}$ and $F' = \emptyset$ and $S_1' = S_1$. We have $\bar{M}_2 \in \boldsymbol{Val}$ by Remark A.2.8, hence $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle \bar{N}, S_2\rangle$, and we conclude using Clause 2 and Subject Reduction (Theorem A.1.5).

Clause 4. Here $M_1 = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ and $M_2 = (\mathrm{ref}_{l,\theta}\ \bar{M}_2)$ where $\bar{M}_1\ \mathcal{T}_{F,\mathit{low}}\ \bar{M}_2$, and $l \not\preceq_F \mathit{low}$. There are two cases.

- **$\bar{M}_1$ can compute.** In this case $M_1' = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ with $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 4 to conclude.
- **$\bar{M}_1$ is a value.** In this case we have $M_1' = a_{l,\theta}$, with a fresh for S_1, $F' = \emptyset$ and $S_1' = S_1 \cup \{a_{l,\theta} \mapsto \bar{M}_1\}$ (and therefore a is also fresh for S_2). Then $\bar{M}_2 \in \boldsymbol{Val}$ by Remark A.2.8, and therefore $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle a_{l,\theta}, S_2 \cup \{a_{l,\theta} \mapsto \bar{M}_2\}\rangle$. If we let $S_2' = S_2 \cup \{a_{l,\theta} \mapsto \bar{M}_2\}$ then $S_1' =^{F,\mathit{low}} S_2'$ since $l \not\preceq_F \mathit{low}$. We conclude using Clause 1 and Subject Reduction (Theorem A.1.5).

Clause 5. Here $M_1 = (!\ \bar{M}_1)$ and $M_2 = (!\ \bar{M}_2)$ where $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$. We distinguish two sub-cases.

- $\bar{M}_1$ **can compute.** In this case $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 5 to conclude.
- $\bar{M}_1$ **is a value.** Then $\bar{M}_1 = a_{l,\theta}$ and $M_1' \in \boldsymbol{Val}$, $S_1' = S_1$ and $F' = \emptyset$. By Remark A.2.8, $\bar{M}_2 \in \boldsymbol{Val}$, and since M_1 and M_2 have the same type, it must be a reference $a_{l',\theta}$. Then, $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle {M_2}', S_2\rangle$ with ${M_2}' \in \boldsymbol{Val}$, and we conclude using Clause 1 and Subject Reduction (Theorem A.1.5).

Clause 6. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$ and $\bar{N}_1\ \mathcal{T}_{F,low}\ \bar{N}_2$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_l$ for some θ and l such that $l \not\preceq_F low$. We distinguish three sub-cases.

- $\bar{M}_1$ **can compute.** In this case $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 6 to conclude.
- $\bar{M}_1$ **is value, but** $\bar{N}_1$ **can compute.** In this case we have $\langle \bar{N}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{N}_1', S_1'\rangle$. By Remark A.2.8 also $\bar{M}_2 \in \boldsymbol{Val}$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 6 to conclude.
- $\bar{M}_1$ **and** $\bar{M}_1$ **are values.** Then $\bar{M}_1 = a_{l,\theta}$ and $M_1' = ()$, $S_1' = \{V \mapsto \bar{M}_1\}S_1$ and $F' = \emptyset$. By Remark A.2.8, also $\bar{M}_2, \bar{N}_2 \in \boldsymbol{Val}$, and since $\bar{M}_1$ and $\bar{M}_2$ have the same type, $\bar{M}_2$ must be a reference $a_{l',\theta'}$. Then, $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle {M_2}', \{V' \mapsto \bar{M}_2\}S_2\rangle$ with $\bar{M}_2' \in \boldsymbol{Val}$. Since $l \not\preceq_F low$, then $\{V \mapsto \bar{M}_1\}S_1 =^{F\cup F', low} \{V' \mapsto \bar{M}_2\}S_2$. We then conclude using Clause 1 and Subject Reduction (Theorem A.1.5).

Clause 7. Here we have $M_1 = (\mathrm{flow}\ \bar{F}\ \mathrm{in}\ \bar{M}_1)$ and $M_2 = (\mathrm{flow}\ \bar{F}\ \mathrm{in}\ \bar{M}_2)$ and $\bar{M}_1\ \mathcal{T}_{F\cup\bar{F},low}\ \bar{M}_2$. There are two cases.

- $\bar{M}_1$ **can compute.** In this case $\langle \bar{M}_1, S_1\rangle \xrightarrow[F'']{} \langle \bar{M}_1', S_1'\rangle$ with $F' = \bar{F} \cup F''$. By induction hypothesis, we have $\langle \bar{M}_2, S_2\rangle \xrightarrow[F'']{} \langle \bar{M}_2', S_2'\rangle$, and $M_1'\ \mathcal{T}_{F\cup\bar{F},low}\ M_2'$ and $S_1' =^{F\cup\bar{F},low} S_2'$. Notice that $S_1' =^{F,low} S_2'$. We use Subject Reduction (Theorem A.1.5) and Clause 7 to conclude.
- $\bar{M}_1$ **is a value.** In this case $M_1' = \bar{M}_1$, $F' = \emptyset$, $N^{\bar{n}_k} = ()$ and $S_1' = S_1$. Then $\bar{M}_2 \in \boldsymbol{Val}$ by Remark A.2.8, and so $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle \bar{M}_2, S_2\rangle$. We conclude using Clause 1 and Subject Reduction (Theorem A.1.5).

□

We have seen in Remark A.2.8 that when two expressions are related by $\mathcal{T}_{F,low}$ and one of them is a value, then the other one is also a value. From a semantical point of view, when an expression has reached a value it means that it has successfully completed its computation.

The following lemma deduces of expressions from that of its subexpressions.

Lemma A.2.11 (Composition of High Expressions)**.** *Suppose that M is typable in F. Then:*

1. *If $M = (M_1\ M_2)$ and M_1 is a syntactically (F, low)-high function and $M_1, M_2 \in \mathcal{H}_{F,low}$, then $M \in \mathcal{H}_{F,low}$.*
2. *If $M = (\mathrm{if}\ M_1\ \mathrm{then}\ M_t\ \mathrm{else}\ M_f)$ and $M_1, M_t, M_f \in \mathcal{H}_{F,low}$, then $M \in \mathcal{H}_{F,low}$.*
3. *If $M = (\mathrm{ref}_{l,\theta}\ M_1)$ and $l \not\preceq_F low$ and $M_1 \in \mathcal{H}_{F,low}$, then $M \in \mathcal{H}_{F,low}$.*
4. *If $M = (M_1; M_2)$ and $M_1, M_2 \in \mathcal{H}_{F,low}$, then $M \in \mathcal{H}_{F,low}$.*

Definition A.2.13 ($\mathcal{R}_{F,low}$)**.** *We have that* $M_1\ \mathcal{R}_{F,low}\ M_2$ *if* $\Gamma \vdash_{A,F} M_1 : s_1, \tau$ *and* $\Gamma \vdash_{A,F} M_2 : s_2, \tau$ *for some* Γ, s_1, s_2 *and* τ *and one of the following holds:*

Clause 1'. $M_1, M_2 \in \mathcal{H}_{F,low}$, *or*

Clause 2'. $M_1 = M_2$, *or*

Clause 3'. $M_1 = (\text{if } \bar{M}_1 \text{ then } \bar{N}_t \text{ else } \bar{N}_f)$ *and* $M_2 = (\text{if } \bar{M}_2 \text{ then } \bar{N}_t \text{ else } \bar{N}_f)$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *and* $\bar{N}_t, \bar{M}_f \in \mathcal{H}_{F,low}$, *or*

Clause 4'. $M_1 = (\bar{M}_1\ \bar{N}_1)$ *and* $M_2 = (\bar{M}_2\ \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1, \bar{N}_2 \in \mathcal{H}_{F,low}$, *and* $\bar{M}_1, \bar{M}_2$ *are syntactically* (F, low)*-high functions, or*

Clause 5'. $M_1 = (\bar{M}_1\ \bar{N}_1)$ *and* $M_2 = (\bar{M}_2\ \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1\ \mathcal{R}_{F,low}\ \bar{N}_2$, *and* $\bar{M}_1, \bar{M}_2$ *are syntactically* (F, low)*-high functions, or*

Clause 6'. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *and* $\bar{N} \in \mathcal{H}_{F,low}$, *or*

Clause 7'. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *with* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$, *or*

Clause 8'. $M_1 = (\text{ref}_{l,\theta}\ \bar{M}_1)$ *and* $M_2 = (\text{ref}_{l,\theta}\ \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *and* $l \not\preceq_F low$, *or*

Clause 9'. $M_1 = (!\ \bar{M}_1)$ *and* $M_2 = (!\ \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *or*

Clause 10'. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{R}_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1, \bar{N}_2 \in \mathcal{H}_{F,low}$, *and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \text{ref}_l$ *for some* θ *and* l *such that* $l \not\preceq_F low$, *or*

Clause 11'. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1\ \mathcal{R}_{F,low}\ \bar{N}_2$, *and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \text{ref}_l$ *for some* θ *and* l *such that* $l \not\preceq_F low$, *or*

Clause 12'. $M_1 = (\text{flow } F' \text{ in } \bar{M}_1)$ *and* $M_2 = (\text{flow } F' \text{ in } \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}_{F \cup F', low}\ \bar{M}_2$.

Figure A.2: The relation $\mathcal{R}_{F,low}$

5. *If* $M = (M_1 := M_2)$ *and* M_1 *has type* $\theta\ \text{ref}_l$ *with* $l \not\preceq_F low$ *and* $M_1, M_2 \in \mathcal{H}_{F,low}$, *then* $M \in \mathcal{H}_{F,low}$.
6. *If* $M = (\text{flow } F \text{ in } M_2)$ *and* $M_2 \in \mathcal{H}_{F \cup F', low}$, *then* $M \in \mathcal{H}_{F,low}$.
7. *If* $M = (\text{allowed } F \text{ then } M_t \text{ else } M_f)$ *and* $M_t, M_f \in \mathcal{H}_{F,low}$, *then* $M \in \mathcal{H}_{F,low}$.

Lemma A.2.12. *If for some* F *and* low *we have that* $M_1\ \mathcal{T}_{F,low}\ M_2$ *and* $M_1 \in \mathcal{H}_{F,low}$, *then* $M_2 \in \mathcal{H}_{F,low}$.

Proof. By induction on the definition of $M_1\ \mathcal{T}_{F,low}\ M_2$. □

A.2.3 Behavior of Typable Low Expressions

In this second phase of the proof, we consider the general case of threads that are typable with any termination level. As in the previous subsection, we show that a typable expression behaves as a strong bisimulation, provided that it is operationally low. For this purpose, we make use of the properties identified for the class of low-terminating expressions by allowing only these to be followed by low-writes. Conversely, high-terminating expressions can only be followed by high-expressions (see Definitions A.2.3 and A.2.4).

We now design a binary relation on expressions that uses $\mathcal{T}_{F,low}$ to ensure that high-terminating expressions are always followed by operationally high ones. The definition of $\mathcal{R}_{G,F,low}$, abbreviated $\mathcal{R}_{F,low}$ when the global flow policy is G, is given in Figure A.2. The flow policy F is assumed to contain G. Notice that it is a symmetric relation. In order to ensure that expressions that are related by $\mathcal{R}_{F,low}$ perform the same changes to the low memory, its definition requires that the references that are created or written using (potentially) different values are high, and that the body of the functions that are applied are syntactically high.

Remark A.2.14. *If $M_1\ \mathcal{T}_{F,low}\ M_2$, then $M_1\ \mathcal{R}_{F,low}\ M_2$.*

The above remark is used to prove the following lemma.

Lemma A.2.15. *If for some F and low we have that $M_1\ \mathcal{R}_{F,low}\ M_2$ and $M_1 \in \mathcal{H}_{F,low}$, then $M_2 \in \mathcal{H}_{F,low}$.*

Proof. By induction on the definition of $M_1\ \mathcal{R}_{F,low}\ M_2$, using Lemma A.2.12. □

We have seen in Splitting Computations (Lemma A.2.1) that two computations of the same expression can split only if the expression is about to read a reference that is given different values by the memories in which they compute. In Lemma A.2.16 we saw that the relation $\mathcal{T}_{F,low}$ relates the possible outcomes of expressions that are typable with a low termination effect. Finally, from the following lemma one can conclude that the above relation $\mathcal{R}_{F,low}$ relates the possible outcomes of typable expressions in general.

Lemma A.2.16. *If there exist Γ, A, s, τ such that $\Gamma \vdash_{A,F} \mathrm{E}[(!\ a_{l,\theta})] : s, \tau$ with $l \not\preceq_{F \cup \lceil \mathrm{E} \rceil} low$, then for any values $V_0, V_1 \in \boldsymbol{Val}$ such that $\Gamma \vdash V_i : \theta$ we have $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$.*

Proof. By induction on the structure of E using Lemma A.1.4, Lemma A.2.9, Lemma A.2.5.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ a_{l,\theta})$. We have $V_0\ \mathcal{R}_{F,low}\ V_1$ by Clause 1'.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]\ M)$. By rule App we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau} \xrightarrow[A,F]{\bar{s}'} \bar{\sigma}$ and $\Gamma \vdash_{A,F} M : \bar{s}'', \bar{\tau}$ with $\bar{s}.r \preceq_F \bar{s}'.w$ and $\bar{s}.t \preceq_F \bar{s}''.w$. By Lemma A.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F \bar{s}'.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil} low$ (therefore $l \not\preceq_F low$), then $\bar{s}'.w \not\preceq_F low$, that is $\mathrm{E}_1[(!\ a_{l,\theta})]$ is a syntactically (F, low)-high function. By Lemma A.1.4, the same holds for $\mathrm{E}_1[V_0]$ and $\mathrm{E}_1[V_1]$. By induction hypothesis we conclude that $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$.

- **$\bar{s}.t \not\preceq_F low$.** Then $\bar{s}''.w \not\preceq_F low$ (and also $\bar{s}''.w \not\preceq low$) so by High Expressions (Lemma A.2.5) we have $M \in \mathcal{H}_{F,low}$. Therefore, we conclude $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 4' and Lemma A.1.4.
- **$\bar{s}.t \preceq_F low$.** Then by Lemma A.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. Therefore, since we have $M\ \mathcal{R}_{F,low}\ M$ by Clause 2', we conclude that $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 5' and Lemma A.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (V\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule App we have that $\Gamma \vdash_{A,F} V : \bar{s}, \bar{\tau} \xrightarrow[A,F]{\bar{s}'} \bar{\sigma}$ and $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}'', \bar{\tau}$ with $\bar{s}''.r \preceq_F \bar{s}'.w$. By Lemma A.2.2, we have $l \preceq \bar{s}''.r$, and so $l \preceq_F \bar{s}'.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil} low$ (therefore $l \not\preceq_F low$), then $\bar{s}'.w \not\preceq_F low$, that is V is a syntactically (F, low)-high function. By Clause 1 we have $V\ \mathcal{T}_{F,low}\ V$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. Therefore we conclude that $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 5' and Lemma A.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{if}\ \mathbf{E_1}[(!\ a_{l,\theta})]\ \mathbf{then}\ M_t\ \mathbf{else}\ M_f)$. By Cond we have that $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \mathsf{bool}$, and $\Gamma \vdash_{A,F} M_t : \bar{s}_t, \bar{\tau}$ and $\Gamma \vdash_{A,F} M_f : \bar{s}_f, \bar{\tau}$ with $\bar{s}.r \preceq_F \bar{s}_t.w, \bar{s}_f.w$. By Lemma A.2.2, we have $l \preceq \bar{s}.r$ and so $l \preceq_F \bar{s}_t.w, \bar{s}_f.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil} low$ (therefore $l \not\preceq_F low$), then $\bar{s}_t.w \not\preceq_F low$ and $\bar{s}_f.w \not\preceq_F low$. This implies that $M_t, M_f \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. Therefore we conclude that $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 3' and Lemma A.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]; M)$. By SEQ we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ and $\Gamma \vdash_{A,F} M : \bar{s}', \bar{\tau}'$ with $\bar{s}.t \preceq_F \bar{s}'.w$.

- $\bar{s}.t \not\preceq_F low$. Then $\bar{s}'.w \not\preceq_F low$ so by High Expressions (Lemma A.2.5) we have $M \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. We can then conclude that $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 6' and Lemma A.1.4.
- $\bar{s}.t \preceq_F low$. Then by Lemma A.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. Therefore, we conclude using Clause 7' and Lemma A.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{ref}_{\bar{l},\bar{\theta}}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By REF we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.r = s.r \preceq_F \bar{l}$ and $\bar{s}.t = s.t$. Therefore, since $l \not\preceq_{F \cup \mathrm{E}} low$ implies $l \not\preceq_{F \cup \mathrm{E}_1} low$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma A.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F low$. Therefore, $\bar{l} \not\preceq_F low$, and we conclude by Lemma A.1.4 and Clause 8'.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule DER we have $\Gamma \vdash_{A,F} \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. We conclude by Lemma A.1.4 and Clause 9'.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})] := M)$. By rule ASS we have that $\Gamma \vdash_{A,F} \mathrm{E}_1[a_{l,\theta}] : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l}}$ with $\bar{s}.r \preceq_F \bar{l}$ and $\bar{s}.t \preceq_F \bar{s}'.w$. By Lemma A.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F low$ and so $\bar{l} \not\preceq_F low$.

- $\bar{s}.t \not\preceq_F low$. Then $\bar{s}'.w \not\preceq_F low$ so by High Expressions (Lemma A.2.5) we have $M \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. We can then conclude that $\mathrm{E}[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}[V_1]$ by Clause 10' and Lemma A.1.4.
- $\bar{s}.t \preceq_F low$. Then by Lemma A.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}_{F,low}\ \mathrm{E}_1[V_1]$. Therefore, we conclude using Lemma A.1.4, Clause 11' and Clause 2' (regarding M).

$\mathbf{E}[(!\ a_{l,\theta})] = (V := \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule ASS we have that $\Gamma \vdash_{A,F} V : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l}}$, $\Gamma \vdash_{A,F} \mathrm{E}_1[a_{l,\theta}] : \bar{s}', \theta$ with $\bar{s}'.r \preceq_F \bar{l}$. By Lemma A.2.2 we have $l \preceq \bar{s}'.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F low$, since otherwise $l \preceq_{F \cup \mathrm{E}} low$. By Clause 1 we have that $V\ \mathcal{T}_{F,low}\ V$, and by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}_{F,low}\ \mathrm{E}_1[V_1]$. We then conclude by Lemma A.1.4 and Clause 11'.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{flow}\ F'\ \mathbf{in}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule FLOW we have $\Gamma \vdash_{A,F \cup F'} V : s, \tau$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}_{F \cup F',low}\ \mathrm{E}_1[V_1]$, so we conclude by Lemma A.1.4 and Clause 12'.

□

We now state a crucial result of the paper: the relation $\mathcal{R}_{F,low}$ is a sort of "strong bisimulation".

Proposition A.2.17 (Strong Bisimulation for Typable Low Threads)**.**
If $M_1\ \mathcal{R}_{F,low}\ M_2$ and $M_1 \notin \mathcal{H}_{F,low}$ and $\langle M_1, S_1\rangle \xrightarrow[F']{} \langle M_1', S_1'\rangle$, with $S_1 =^{F \cup F', low} S_2$ such that a is fresh for S_2 if $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$, then there exist M_2' and S_2' such that $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle M_2', S_2'\rangle$ with $M_1'\ \mathcal{R}_{F,low}\ M_2'$ and $S_1' =^{F,low} S_2'$.

Proof. We use Subject Reduction (Theorem A.1.5) to guarantee typability (with the same type) for m_j, *low* and F, which is a requirement for being in the $\mathcal{R}_{F,low}$ relation. We also use the Strong Bisimulation for Low Terminating Threads Lemma (Lemma A.2.10).

Clause 1'. This case is excluded by assumption.

Clause 2'. Here $M_1 = M_2$. By Guaranteed Transitions (Lemma A.2.6) there exist M_2' and S_2' such that $\langle M_2, S_2\rangle \xrightarrow[F']{} \langle M_2', S_2'\rangle$ with $S_1' =^{F \cup F', low} S_2'$.

- $\mathbf{M_2' = M_1'}$. Then we have $M_1'\ \mathcal{R}_{F,low}\ M_2'$, by Clause 2' and Subject Reduction (Theorem A.1.5).

$\boldsymbol{M_2' \neq M_1'}$**.** Then by Splitting Computations (Lemma A.2.1) we have that there exists E and $a_{l,\theta}$ such that $F' = \lceil \mathrm{E} \rceil$, $M_1' = \mathrm{E}[S_1(a_{l,\theta})]$, $M_2' = \mathrm{E}[S_2(a_{l,\theta})]$, $S_1' = S_1$ and $S_2' = S_2$. Since $S_1(a_{l,\theta}) \neq S_2(a_{l,\theta})$, we have $l \not\preceq_{F \cup F'} low$. Therefore, $M_1' \; \mathcal{R}_{F,low} \; M_2'$, by Lemma A.2.16 above.

Clause 3'. Here we have $M_1 = (\text{if } \bar{M}_1 \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ and $M_2 = (\text{if } \bar{M}_2 \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ with $\bar{M}_1 \; \mathcal{R}_{F,low} \; \bar{M}_2$ and $\bar{M}_t, \bar{M}_f \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1 \notin \mathcal{H}_{F,low}$, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Therefore, $M_1' = (\text{if } \bar{M}_1' \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ with $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. We use the induction hypothesis, Clause 3' and Subject Reduction (Theorem A.1.5) to conclude.

Clause 4'. Here $M_1 = (\bar{M}_1 \; \bar{N}_1)$ and $M_2 = (\bar{M}_2 \; \bar{N}_2)$ with $\bar{M}_1 \; \mathcal{R}_{F,low} \; \bar{M}_2$, $\bar{M}_1$ and $\bar{M}_2$ are syntactically (F, low)-high functions, and $\bar{N}_1, \bar{N}_2 \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1$ can compute, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Therefore, $M_1' = (\bar{M}_1' \; \bar{N}_1)$ with $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. We use the induction hypothesis, Clause 4' and Subject Reduction (Theorem A.1.5) to conclude.

Clause 5'. Here $M_1 = (\bar{M}_1 \; \bar{N}_1)$ and $M_2 = (\bar{M}_2 \; \bar{N}_2)$ with $\bar{M}_1 \; \mathcal{T}_{F,low} \; \bar{M}_2$, $\bar{M}_1$ and $\bar{M}_2$ are syntactically (F, low)-high functions, and $\bar{N}_1 \; \mathcal{R}_{F,low} \; \bar{N}_2$. We distinguish two sub-cases:

- $\boldsymbol{\bar{M}_1}$ **can compute.** In this case exists $\bar{M}_1'$ such that $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. Here we use Lemma A.2.10, Subject Reduction (Theorem A.1.5) and Clause 5' to conclude.
- $\boldsymbol{\bar{M}_1}$ **is a value.** Then by Remark A.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. We can assume that $\bar{N}_1, \bar{N}_2 \notin \mathcal{H}_{F,low}$, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Then, $\bar{N}_1$ can compute, and so there exist $\bar{N}_1'$ such that $\langle \bar{N}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{N}_1', S_1' \rangle$ with $M_1' = (\bar{M}_1 \; \bar{N}_1')$. We use the induction hypothesis, Clause 5' and Subject Reduction (Theorem A.1.5) to conclude.

Clause 6'. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ where $\bar{M}_1 \; \mathcal{R}_{F,low} \; \bar{M}_2$ and $\bar{N} \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1 \notin \mathcal{H}_{F,low}$, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Therefore, we have $M_1' = (\bar{M}_1'; \bar{N})$ with $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. We use the induction hypothesis, Clause 6' and Subject Reduction (Theorem A.1.5) to conclude.

Clause 7'. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ with $\bar{M}_1 \; \mathcal{T}_{F,low} \; \bar{M}_2$. We distinguish two sub-cases:

- $\boldsymbol{\bar{M}_1}$ **can compute.** In this case exists $\bar{M}_1'$ such that $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. Here we use Lemma A.2.10, Subject Reduction (Theorem A.1.5) and Clause 7' to conclude.
- $\boldsymbol{\bar{M}_1}$ **is a value.** Then $M_1' = \bar{N}$, $F = \emptyset$, $N = ()$ and $S_1' = S_1$. By Remark A.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. Then, we have $\langle M_2, S_1 \rangle \xrightarrow[F']{} \langle \bar{N}, S_1' \rangle$. We conclude using Lemma A.2.10 and Clause 2'.

Clause 8'. Here $M_1 = (\mathrm{ref}_{l,\theta} \; \bar{M}_1)$ and $M_2 = (\mathrm{ref}_{l,\theta} \; \bar{M}_2)$ where $\bar{M}_1 \; \mathcal{R}_{F,low} \; \bar{M}_2$, and $l \not\preceq_F low$. We can assume that $\bar{M}_1 \notin \mathcal{H}_{F,low}$, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Then, $\bar{M}_1$ can compute, and $M_1' = (\mathrm{ref}_{l,\theta} \; \bar{M}_1)$ with $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 8' to conclude.

Clause 9'. Here $M_1 = (! \; \bar{M}_1)$ and $M_2 = (! \; \bar{M}_2)$ where $\bar{M}_1 \; \mathcal{R}_{F,low} \; \bar{M}_2$. We know that $\bar{M}_1$ can compute, since otherwise $M_1 \in \mathcal{H}_{F,low}$. Then, we have $\langle \bar{M}_1, S_1 \rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1' \rangle$. We use the induction hypothesis, Subject Reduction (Theorem A.1.5) and Clause 9' to conclude.

Clause 10'. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1 \mathcal{R}_{F,low} \bar{M}_2$, and $\bar{N}_1, \bar{N}_2 \in \mathcal{H}_{F,low}$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_l$ for some θ and l such that $l \not\leq_F low$. We can assume that $\bar{M}_1$ can compute, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). Therefore, $M_1' = (\bar{M}_1' := \bar{N}_1)$ with $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use the induction hypothesis, Clause 10' and Subject Reduction (Theorem A.1.5) to conclude.

Clause 11'. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1 \mathcal{T}_{F,low} \bar{M}_2$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_l$ for some θ and l such that $l \not\leq_F low$, and $\bar{N}_1 \mathcal{R}_{F,low} \bar{N}_2$. We can assume that M_1 cannot be a redex, with $\bar{M}_1, \bar{N}_1 \in \boldsymbol{Val}$, since otherwise $M_1 \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma A.2.11). There are two cases to consider:

- **$\bar{M}_1$ can compute.** Then we have $\langle \bar{M}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{M}_1', S_1'\rangle$. We use Lemma A.2.10, Clause 11' and Subject Reduction (Theorem A.1.5) to conclude.
- **$\bar{M}_1$ is a value but $\bar{N}_1$ can compute.** Then by Remark A.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. Then we have $\langle \bar{N}_1, S_1\rangle \xrightarrow[F']{} \langle \bar{N}_1', S_1'\rangle$. We conclude using induction hypothesis, Clause 11' and Subject Reduction (Theorem A.1.5).

Clause 12'. Here $M_1 = (\text{flow } F' \text{ in } \bar{M}_1)$ and $M_2 = (\text{flow } F' \text{ in } \bar{M}_2)$ with $\bar{M}_1 \mathcal{R}_{F\cup F',low} \bar{M}_2$. We can assume that $\bar{M}_1 \notin \mathcal{H}_{F\cup F',low}$, since otherwise $\bar{M}_1 \notin \mathcal{H}_{F,low}$ and by Composition of High Expressions (Lemma A.2.11) $M_1 \in \mathcal{H}_{F,low}$. Therefore $\langle \bar{M}_1, S_1\rangle \xrightarrow[F'']{} \langle \bar{M}_1', S_1'\rangle$ with $F' = \bar{F} \cup F''$. By induction hypothesis, we have that $\langle \bar{M}_2, S_2\rangle \xrightarrow[F'']{} \langle \bar{M}_2', S_2'\rangle$, and that $M_1' \mathcal{R}_{F\cup\bar{F},low} M_2'$ and also $S_1' =^{F\cup\bar{F},low} S_2'$. Notice that $S_1' =^{F,low} S_2'$. We use Subject Reduction (Theorem A.1.5) and Clause 12' to conclude.

□

A.2.4 Behavior of Typable Expressions

We can now use Strong Bisimulation for Low Typable Threads (Proposition A.2.17) to prove the following:

Proposition A.2.18. *The relation $\mathcal{H}_{G,low} \cup \mathcal{R}_{G,low}$ is a low-bisimulation.*

To conclude, we now state the main result of the chapter, saying that our type system only accepts threads that can securely run concurrently with other typable threads.

Theorem A.2.19 (Soundness for Local Non-disclosure)**.** *If M is an expression for which there exist Γ, s and τ such that $\Gamma \vdash_{\emptyset,\emptyset} M : s, \tau$, then M satisfies the Non-disclosure policy.*

Proof. By Clause 2' of Definition A.2.13, for all choices of security levels *low*, we have that $M \mathcal{R}_{G,low} M$, and therefore, $M \mathcal{H}_{G,low} \cup \mathcal{R}_{G,low} M$. This holds in particular for $G = \emptyset$. □

A.3 Confinement to a flow policy

Theorem A.3.1 (Soundness of Typing Confinement)**.** *If M is an expression for which there exist Γ, s and τ such that $\Gamma \vdash_{\emptyset,\emptyset} M : s, \tau$, then M is operationally A-confined, for all A.*

Proof. Consider the following set:

$$C_A = \{M \mid \Gamma \vdash_{A',F'} M : s, \sigma \text{ with } F' \subseteq A'^* \text{ and } A' \subseteq A^*\}$$

We show that, for all A the set C_A is a set of operationally A-confined expressions. We use induction on the inference of $\Gamma \vdash_{A',F'} M : s, \sigma$, where $M \in C_A$, by case analysis on the last rule used in this typing proof, te show that if $A \vdash \langle M, S\rangle \xrightarrow[F]{} \langle M', S'\rangle$ then $F \subseteq A^*$ and $M' \in C_A$.

Flow. Here $M = (\text{flow } \bar{F} \text{ in } \bar{N})$, and we have $\Gamma \vdash_{A', F' \cup \bar{F}} \bar{N} : \bar{s}', \tau$ with $\bar{F} \subseteq A'^*$. There are two possibilities:

- $\bar{N}$ **can compute.** Then we have $A \vdash \langle \bar{N}, S \rangle \underset{\bar{F}'}{\longrightarrow} \langle \bar{N}', S' \rangle$ with $F = \bar{F}' \cup \bar{F}$. Notice that $F' \cup \bar{F} \subseteq A'^*$ by assumptions. By induction hypothesis, $\bar{F}' \subseteq A^*$. By Subject Reduction (Proposition A.1.5), $\Gamma \vdash_{A' \cup A'', F'} \bar{N}' : \bar{s}'', \tau$ with $A'' \subseteq A^*$. Therefore, $\bar{M}' \in C_A$.
- $\bar{N} = V$**.** Then we have $A \vdash \langle (\text{flow } \bar{M} \text{ in } \bar{N}), S \rangle \underset{F}{\longrightarrow} \langle V, S \rangle$ with $F = \emptyset$, so $F \subseteq A^*$ holds vacuously. Since $\Gamma \vdash V : \tau$, then $V \in C_A$.

Allow. Here $M = (\text{allowed } \bar{F} \text{ then } N_t \text{ else } N_f)$ and we have $\Gamma \vdash_{A' \cup \bar{F}, F'} N_t : \bar{s}_t, \tau$ and $\Gamma \vdash_{A', F'} N_f : \bar{s}_f, \tau$. There are two possibilities:

- $\bar{F} \subseteq A^*$**.** Then we have $A \vdash \langle M, S \rangle \underset{F}{\longrightarrow} \langle N_t, S \rangle$ with $F = \emptyset$, so $F \subseteq A^*$ holds vacuously. By assumption, it is obvious that $F' \subseteq A' \cup \bar{F}$. Also, in this case we have $\bar{F} \subseteq A^*$, so $A' \cup \bar{F} \subseteq A^*$ Therefore, $N_t \in C_A$.
- $\bar{F} \not\subseteq A^*$**.** Then we have $A \vdash \langle M, S \rangle \underset{F}{\longrightarrow} \langle N_f, S \rangle$ with $F = \emptyset$, so $F \subseteq A^*$ holds vacuously. Therefore, $N_f \in C$.

□

Appendix B

Global Perspective (proofs)

B.1 Subject Reduction

In order to establish the soundness of the type system of Figure 1.7 we need a Subject Reduction result, stating that types that are given to expressions are preserved by computation. To prove it we follow the usual steps [Wright and Felleisen(1994)].

Remark B.1.1. *If $W \in \boldsymbol{Pse}$ and $\Gamma \vdash^j_F W : s, \tau$, then for all flow policies F', we have that $\Gamma \vdash^j_{F'} W : \langle \bot, \top, \bot \rangle, \tau$.*

Lemma B.1.2.

1. *If $\Gamma \vdash^j_F M : s, \tau$ and $x \notin \mathrm{dom}(\Gamma)$ then $\Gamma, x : \sigma \vdash^j_F M : s, \tau$.*
2. *If $\Gamma, x : \sigma \vdash^j_F M : s, \tau$ and $x \notin \mathrm{fv}(M)$ then $\Gamma \vdash^j_F M : s, \tau$.*

Proof. By induction on the inference of the type judgment. □

Lemma B.1.3 (Substitution)**.**
If $\Gamma, x : \sigma \vdash^j_F M : s, \tau$ and $\Gamma \vdash W : \sigma$ then $\Gamma \vdash^j_F \{x \mapsto W\}M : s, \tau$.

Proof. By induction on the inference of $\Gamma, x : \tau \vdash^j_F M : s, \sigma$, and by case analysis on the last rule used in this typing proof, using the previous lemma. □

Lemma B.1.4 (Replacement)**.**
If $\Gamma \vdash^j_F \mathrm{E}[M] : s, \tau$ is a valid judgment, then the proof gives M a typing $\Gamma \vdash^j_{F \cup \lceil \mathrm{E} \rceil} M : \bar{s}, \bar{\tau}$ for some $\bar{s}$ and $\bar{\tau}$ such that $\bar{s}.r \preceq s.r$, $s.w \preceq \bar{s}.w$, $\bar{s}.t \preceq s.t$ and $s.d \preceq \bar{s}.d$. In this case, if $\Gamma \vdash^j_{F \cup \lceil \mathrm{E} \rceil} N : \bar{s}', \bar{\tau}$ with $\bar{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \bar{s}'.w$, $\bar{s}'.t \preceq \bar{s}.t$ and $\bar{s}.d \preceq \bar{s}'.d$, then $\Gamma \vdash^j_F \mathrm{E}[N] : s', \tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $s.d \preceq s'.d$.

Proof. By induction on the structure of E. □

Proposition B.1.5 (Subject Reduction)**.** *Consider a thread M^{m_j} for which there exist Γ, F, s and τ such that $\Gamma \vdash^j_F M : s, \tau$. Then, if $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$, there exists s' such that $\Gamma \vdash^j_F M' : s', \tau$, and where $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq s'.d$. Furthermore, $\exists s''$ such that $\Gamma \vdash^k_\emptyset N : s'', \mathsf{unit}$ and $s.w \preceq s''.w$ and $s.d \cup W(T(n_k)) \preceq s''.d$.*

Proof. We consider the smallest $\bar{M}$ such that $M = \bar{\mathrm{E}}[\bar{M}]$ in the sense that there is no $\hat{\mathrm{E}}, \hat{M}, \hat{N}$ such that $\hat{\mathrm{E}} \neq []$ and $\hat{\mathrm{E}}[\hat{M}] = \bar{M}$ for which we can write $\langle W, \{\hat{M}^{m_j}\}, T, S\rangle \xrightarrow[\hat{F}]{\hat{N}^{n_k}} \langle W, \{\hat{M}'^{m_j}\}, T', S'\rangle$.

We then proceed by case analysis on the transition $\langle W, \{\bar{M}^{m_j}\}, T, S\rangle \xrightarrow[\bar{F}]{\bar{N}^{n_k}} \langle W, \{\bar{M}'^{m_j}\}, T', S'\rangle$, using Lemmas B.1.3 and B.1.4.

Suppose that $M = \bar{\mathrm{E}}[\bar{M}]$ and that $\langle W, \{\bar{M}^{m_j}\}, T, S\rangle \xrightarrow[\bar{F}]{\bar{N}^{n_k}} \langle W, \{\bar{M}'^{m_j}\}, T', \bar{S}'\rangle$. We start by observing that this implies $F' = \bar{F} \cup \lceil \bar{\mathrm{E}} \rceil$, $M' = \bar{\mathrm{E}}[\bar{M}']$, $\bar{N} = N$ and $\bar{S}' = S'$. We can assume, without loss of generality, that $\bar{M}$ is the smallest in the sense that there is no $\hat{\mathrm{E}}, \hat{M}, \hat{N}$ such that $\hat{\mathrm{E}} \neq []$ and $\hat{\mathrm{E}}[\hat{M}] = \bar{M}$ for which we can write $\langle W, \{\hat{M}^{m_j}\}, T, S\rangle \xrightarrow[\hat{F}]{\hat{N}^{n_k}} \langle W, \{\hat{M}'^{m_j}\}, T', S'\rangle$.

By Lemma B.1.4, we have $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} \bar{M} : \bar{s}, \bar{\tau}$ in the proof of $\Gamma \vdash^{j}_{F} \bar{\mathrm{E}}[\bar{M}] : s, \tau$, for some $\bar{s}$ and $\bar{\tau}$. We proceed by case analysis on the transition $\langle W, \{\bar{M}^{m_j}\}, T, S\rangle \xrightarrow[\bar{F}]{\bar{N}^{n_k}} \langle W, \{\bar{M}'^{m_j}\}, T', S'\rangle$, and prove that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} \bar{M}' : \bar{s}', \bar{\tau}$, for some $\bar{s}'$ such that $\bar{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \bar{s}'.w$, $\bar{s}'.t \preceq \bar{s}.t$ and $\bar{s}.d \cup W(T(m_j)) \preceq \bar{s}'.d$.

$\bar{M} = ((\lambda x.\hat{M})\ V)$. Here we have $\bar{M}' = \{x \mapsto V\}\hat{M}$. By rule App, we have $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} (\lambda x.\hat{M}) : \hat{s}, \hat{\tau} \xrightarrow[F \cup \lceil \bar{\mathrm{E}} \rceil, j]{\hat{s}'} \hat{\sigma}$ and $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} V : \hat{s}'', \hat{\tau}$, where $\hat{s}'.r \preceq \bar{s}.r$, $\bar{s}.w \preceq \hat{s}'.w$ and $\hat{s}'.t \preceq \bar{s}.t$. and $\bar{s}.d \cup W(T(m_j)) \preceq \hat{s}'.d$. By Abs, then $\Gamma, x : \hat{\tau} \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} \hat{M} : \hat{s}', \hat{\sigma}$, and by Remark B.1.1 we have $\Gamma \vdash V : \hat{\tau}$. Therefore, by Lemma B.1.3, we get $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} \{x \mapsto V\}\hat{M} : \hat{s}', \hat{\sigma}$.

$\bar{M} =$ (if *tt* then N_t else N_f). Here we have $\bar{M}' = N_t$. By Cond, we have that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} N_t : s_t, \bar{\tau}$, where $s_t.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_t.w$, $s_t.t \preceq \bar{s}.t$ and $\bar{s}.d \cup W(T(m_j)) \preceq s_t.d$.

$\bar{M} = (\mathrm{ref}_{l,\theta}\ V)$. Here we have $\bar{M}' = a_{l,\theta}$. By Loc, we have $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} a : \langle \bot, \top, \bot, \top\rangle, \theta\ \mathrm{ref}_l$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$, $\bot \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq \top$.

$\bar{M} =$ (! $a_{l,\theta}$). Here we have $\bar{M}' = S(a_{l,\theta})$. By assumption, we have that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} S(a_{l,\theta}) : \langle \bot, \top, \bot, \top\rangle, \theta$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$ and $\bot \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq \top$.

$\bar{M} =$ (flow F' in V). Here we have $\bar{M}' = V$. By rule Flow, we have that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil \cup F'} V : \hat{s}', \tau$ and by Remark B.1.1, we have $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} V : \langle \bot, \top, \bot, \top\rangle, \bar{\tau}$, which satisfies $\bot \preceq s.r$, $s.w \preceq \top$, $\bot \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq \top$.

check flow conditions in Local and Global perspective

$\bar{M} =$ (allowed F' then N_t else N_f) and $F' \subseteq W(T(m_j))^*$. Here we have $\bar{M}' = N_t$. By Allow, we have that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} N_t : s_t, \bar{\tau}$, where $s_t.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_t.w$ and $s_t.t \preceq \bar{s}.t$. As to the declassification effect, we have that $\bar{s}.d = s.d \cup s_t.d - F' \cup s_f.d \cup W(T(m_j)) \preceq s_t.d$ holds because $s_t.d \subseteq (s_t.d - F' \cup W(T(m_j)))^* = (s_t.d \cup W(T(m_j)))^*$.

$\bar{M} =$ (allowed F' then N_t else N_f) and $F' \not\subseteq W(T(m_j))^*$. Here we have $\bar{M}' = N_f$. By Allow, we have that $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} N_f : s_f, \bar{\tau}$, where $s_f.r \preceq \bar{s}.r$, $\bar{s}.w \preceq s_f.w$, $s_f.t \preceq \bar{s}.t$ and $\bar{s}.d \cup W(T(m_j)) \preceq s_t.d$.

The proof for the case $\bar{M} = (\varrho x.W)$ is analogous to the one for $\bar{M} = ((\lambda x.\hat{M})\ V)$, while the proofs for the cases $\bar{M} = (\mathrm{if}\ ff\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f)$ and $\bar{M} = (V; \hat{M})$ are analogous to the one for $\bar{M} = (\mathrm{if}\ tt\ \mathrm{then}\ N_t\ \mathrm{else}\ N_f)$, and the ones for $\bar{M} = (a_{l,\theta} := V)$, $\bar{M} = (\mathrm{thread}_l\ \hat{M})$ is analogous to the one for $\bar{M} = (\mathrm{ref}_{l,\theta}\ V)$. By Lemma B.1.4, we can conclude that $\Gamma \vdash^{j}_{F} \bar{\mathrm{E}}[\bar{M}'] : s', \tau$, for some s' such that $s'.r \preceq s.r$, $s.w \preceq s'.w$, $s'.t \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq s'.d$.

Now, if $N^{n_k} \neq ()$ (N^{n_k} is created), then $\exists \hat{N} : M = \bar{\mathrm{E}}[(\mathrm{thread}_k\ \hat{N})]$ and $\bar{N} = \hat{N}$. By Lemma B.1.4, we have $\Gamma \vdash^{j}_{F \cup \lceil \bar{\mathrm{E}} \rceil} (\mathrm{thread}_k\ \hat{N}) : \hat{s}, \mathsf{unit}$ in the proof of $\Gamma \vdash^{j}_{F} \bar{\mathrm{E}}[(\mathrm{thread}\ \hat{N})] : s, \tau$, for some $\hat{s}$, and $\hat{\tau}$. By Thr, we have $\Gamma \vdash^{k}_{\emptyset} \hat{N} : \hat{s}, \mathsf{unit}$, where $\hat{s} = \langle \bot, s.w, \bot, s.d\rangle$. Therefore, $\hat{s}.r \preceq s.r$, $s.w \preceq \hat{s}.w$, $\hat{s}.t \preceq s.t$ and $s.d \cup W(T(m_j)) \preceq \hat{s}.d$. □

B.2 Non-diclosure for Networks

We now present the main steps for proving soundness of the type system of Figure 1.7 with respect to the notion of security of Definition 1.3.6.

B.2.1 Basic Properties

Properties of the Semantics One can prove that the semantics preserves the conditions for well-formedness, and that a configuration with a single expression has at most one transition, up to the choice of new names.

The following result states that, if the evaluation of a thread M^{m_j} differs in the context of two distinct states while not creating two distinct reference names or thread names, this is because either M^{m_j} is performing a dereferencing operation, which yields different results depending on the memory, or because M^{m_j} is testing the allowed policy.

Lemma B.2.1 (Splitting Computations)**.**
If we have that $\langle W, \{M^{m_j}\}, T_1, S_1\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{{M_1}'^{m_j}\}, T_1', S_1'\rangle$ *and also that* $\langle W, \{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N'^{n_k}} \langle W, \{{M_2}'^{m_j}\}, T_2', S_2'\rangle$ *with* ${M_1}'^{m_j} \neq {M_2}'^{m_j}$ *and* $\mathrm{dom}(T_2' - T_2) = \mathrm{dom}(T_1' - T_1)$, $\mathrm{dom}(S_2' - S_2) = \mathrm{dom}(S_1' - S_1)$, *then* $N^{n_k} = () = N'^{n_k}$ *and either:*

- *there exist* E *and* $a_{l,\theta}$ *such that* $F = \lceil \mathrm{E} \rceil = F'$, $M = \mathrm{E}[(!\ a_{l,\theta})]$, *and* $M' = \mathrm{E}[S_1(a_{l,\theta})]$, $M'' = \mathrm{E}[S_2(a_{l,\theta})]$ *with* $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$ *and* $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$, *or*
- *there exist* E *and* $\bar{F}$ *such that* $F = \lceil \mathrm{E} \rceil = \bar{F}$, $M = \mathrm{E}[(\text{allowed } F' \text{ then } N_t \text{ else } N_f)]$, *and* $T_1(m_j) \neq T_2(m_j)$ *with* $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$ *and* $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$, *or.*

Proof. Note that the only rules that depend on the state are those for the reduction of $\mathrm{E}[(!\ a_{l,\theta})]$ and of $\mathrm{E}[(\text{allowed } F' \text{ then } N_t \text{ else } N_f)]$. By case analysis on the transition $\langle W, \{M^{m_j}\}, T_1, S_1\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{{M_1}'^{m_j}\}, T_1', S_1'\rangle$. □

Effects

Lemma B.2.2 (Update of Effects)**.**

1. *If* $\Gamma \vdash_F^j \mathrm{E}[(!\ n_k.u_{l,\theta})] : s, \tau$ *then* $l \preceq s.r$.
2. *If* $\Gamma \vdash_F^j \mathrm{E}[(n_k.u_{l,\theta} := V)] : s, \tau$, *then* $s.w \preceq l$.
3. *If* $\Gamma \vdash_F^j \mathrm{E}[(\mathrm{ref}_{l,\theta}\ V)] : s, \tau$, *then* $s.w \preceq l$.
4. *If* $\Gamma \vdash_F^j \mathrm{E}[(\text{goto } d)] : s, \tau$, *then* $s.w \preceq j$.
5. *If* $\Gamma \vdash_F^j \mathrm{E}[(\text{allowed } F \text{ then } N_t \text{ else } N_f)] : s, \tau$, *then* $j \preceq s.t$.

Proof. By induction on the structure of E. □

High Expressions We can identify a class of threads that have the property of never performing any change in the "low" part of the memory. These are classified as being "high" according to their behavior[1]:

[1]The notion of "operationally high thread" that we define here should not not be confused with the notion of "high thread". The former refers to the security level that is associated with a thread, while the latter refers to the changes that the thread performs on the state.

Definition B.2.3 (Operationally High Threads)**.** *A set $\mathcal{H}$ of threads is said to be a set of* operationally (F,l)-high threads *if the following holds for any $M^{m_j} \in \mathcal{H}$:*

$$\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle \text{ implies}$$

$$\langle T, S\rangle =^{F,l} \langle T', S'\rangle \text{ and both } M'^{m_j}, N^{n_k} \in \mathcal{H}$$

The largest set of operationally (F,l)-high threads is denoted by $\mathcal{H}_{F,l}$. We then say that a thread M^{m_j} is operationally (F,l)-high, if $M^{m_j} \in \mathcal{H}_{F,l}$.

Remark that for any F and l there exists a set of operationally (F,l)-high threads, like for instance $\{V^{m_j} \mid V \in \boldsymbol{Val}\}$. Furthermore, the union of a family of sets of operationally (F,l)-high threads is a set of operationally (F,l)-high threads. Notice that if $F' \subseteq F$, then any operationally (F,l)-high thread is also operationally (F',l)-high.

Some expressions can be easily classified as "high" by the type system, which only considers their syntax. These cannot perform changes to the "low" memory simply because their code does not contain any instruction that could perform them. Since the writing effect is intended to be a lower bound to the level of the references that the expression can create or assign to, expressions with a high writing effect can be said to be *syntactically high*:

Definition B.2.4 (Syntactically High Expressions)**.** *An expression M is* syntactically (F,l,j)-high *if there exists Γ, A, s, τ such that $\Gamma \vdash^j_F M : s, \tau$ with $s.w \not\preceq_F l$. The expression M is a* syntactically (F,l,j)-high function *if there exists Γ, s, τ such that $\Gamma \vdash M : \tau \xrightarrow[F,j]{s} \sigma$ with $s.w \not\preceq_F l$.*

We can now state that syntactically high expressions have an operationally high behavior.

Lemma B.2.5 (High Expressions)**.** *If M is a syntactically (F,l,j)-high expression, then M^{m_j} is an operationally (F,l)-high thread.*

Proof. We show that if M is syntactically (F,l,j)-high, that is if there exists Γ, s, τ such that $\Gamma \vdash^j_F M : s, \tau$ with $s.w \not\preceq_F l$, and $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$ then $S' =^{F,l} S$. This is enough since, by Subject Reduction (Theorem B.1.5), both M' is syntactically (F,l,j)-high and N is syntactically (F,l,k)-high. We proceed by cases on the proof of the transition $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$. The lemma is trivial in all the cases where $\langle T, S\rangle = \langle T', S'\rangle$.

- $\boldsymbol{M = \mathrm{E}[(a_{\bar{l},\bar{\theta}} := V)]}$**.** Here $S' = [a_{\bar{l},\bar{\theta}} := V]S$ and so $s.w \preceq \bar{l}$ by Lemma B.2.2. This implies $\bar{l} \not\preceq_F l$, hence $S' =^{F,l} S$.
- $\boldsymbol{M = \mathrm{E}[(\mathbf{goto}\ d)]}$**.** Here $T' = [m_j := d]T$ and also $s.w \preceq j$ by Lemma B.2.2. This implies $j \not\preceq_F l$, hence $T' =^{F,l} T$.

The proof of the case $M = \mathrm{E}[(\mathrm{ref}_{l,\theta}\ V)]$ is analogous to the proof for $M = \mathrm{E}[(a_{l,\theta} := V)]$, while the proof for the case $M = \mathrm{E}[(\mathrm{thread}_l\ M_0)]$ is analogous to the one for $M = \mathrm{E}[(\mathrm{goto}\ d)]$. □

B.2.2 Behavior of "Low"-Terminating Expressions

Recall that, according to the intended meaning of the termination effect, the termination or non-termination of expressions with low termination effect should only depend on the low part of the state. In other words, two computations of a same thread running under two "low"-equal states should either both terminate or both diverge. In particular, this implies that termination-behavior of these expressions cannot be used to leak "high" information when composed with other expressions (via termination leaks).

The ability of a thread containing a migrating instruction to compute depends on whether it is typable with a declassification effect that complies to the allowed flow policy of the destination site. The following guaranteed-transition result holds for low-equal states.

Lemma B.2.6 (Guaranteed Transitions)**.** *Consider a thread M^{m_j} that is typable for F. If $\langle\{M^{m_j}\}, T_1, S_1\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{M_1'^{m_j}\}, T_1', S_1'\rangle$ such that $\bar{n}_{\bar{k}}$ is fresh for T_2 if $\bar{n}_{\bar{k}} \in \mathrm{dom}(T_1' - T_1)$ and a is fresh for S_2 if $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$ and for some F' we have $\langle T_1, S_1\rangle =^{F\cup F', low} \langle T_2, S_2\rangle$, then there exist M_2', T_2' and S_2' such that $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{M_2'^{m_j}\}, T_2', S_2'\rangle$ with $\langle T_1', S_1'\rangle =^{F\cup F', low} \langle T_2', S_2'\rangle$.*

Proof. By case analysis on the proof of $\langle\{M^{m_j}\}, T_1, S_1\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{M_1'^{m_j}\}, T_1', S_1'\rangle$. In most cases, this transition does not modify or depend on the state $\langle T_1, S_1\rangle$, and we may let $M_2' = M_1'$ and $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$.

$\boldsymbol{M = \mathrm{E}[(\mathbf{ref}_{l,\theta}\ V)]}$**.** Here $M' = \mathrm{E}[a_{l,\theta}]$, $F = \lceil\mathrm{E}\rceil$, $N^{\bar{n}_{\bar{k}}} = \langle\rangle$, $T_1' = T_1$ and $S_1' = S_1 \cup \{a_{l,\theta} \mapsto V\}$. Since a is fresh for S_2, we also have that $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{M_1'^{m_j}\}, T_2, S_2' \cup \{a_{l,\theta} \mapsto V\}\rangle$.

$\boldsymbol{M = \mathrm{E}[(!\ a_{l,\theta})]}$**.** Here $M' = \mathrm{E}[S_1(a_{l,\theta})]$, $F = \lceil\mathrm{E}\rceil$, $N^{\bar{n}_{\bar{k}}} = \langle\rangle$, and $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$. We have $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{\mathrm{E}[S_2(a_{l,\theta})]^{m_j}\}, T_2, S_2\rangle$.

$\boldsymbol{M = \mathrm{E}[(a_{l,\theta} := V)]}$**.** then $M' = \mathrm{E}[\langle\rangle]$, $F = \lceil\mathrm{E}\rceil$, $N^{\bar{n}_{\bar{k}}} = \langle\rangle$, $T_1' = T_1$ and $S_1' = [a_{l,\theta} := V]S_1$. We have $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{\mathrm{E}[\langle\rangle]^{m_j}\}, T_2, [a_{l,\theta} := V]S_2\rangle$.

$\boldsymbol{M = \mathrm{E}[(\mathbf{thread}_{\bar{k}}\ \bar{M})]}$**.** Here $M' = \mathrm{E}[\langle\rangle]$, $F = \emptyset$, $N^{\bar{n}_{\bar{k}}} = \bar{M}^{\bar{n}_{\bar{k}}}$, $T_1' = T_1 \cup \{\bar{n}_{\bar{k}} \mapsto T_1(m_j)\}$, and $S_1' = S_1$. Since n is fresh for T_2, $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{\mathrm{E}[\langle\rangle]^{m_j}\}, T_2 \cup \{\bar{n}_{\bar{k}} \mapsto T_2(m_j)\}, S_2\rangle$. Notice that $T_1 \cup \{\bar{n}_{\bar{k}} \mapsto T_1(m_j)\} =^{F\cup F', low} T_2 \cup \{\bar{n}_{\bar{k}} \mapsto T_2(m_j)\}$, because $T_1 =^{F\cup F', low} T_2$ and if $l \preceq_{F\cup F'} low$, then by the condition $j \preceq_{F\cup F'} l$ in rule Thr also $j \preceq_{F\cup F'} low$, in which case $T_1(m_j) = T_2(m_j)$.

$\boldsymbol{M = \mathrm{E}[(\mathbf{goto}\ d')]}$. Then $M' = \mathrm{E}[\langle\rangle]$, $F = \emptyset$, $N^{\bar{n}_{\bar{k}}} = \langle\rangle$, $T_1' = [m_j := d']T_1$ and $S_1' = S_1$. This means that $\Gamma \vdash^j_{\emptyset} \mathrm{E}[\langle\rangle] : s, \tau$ and that $s.d \subseteq W(d)^*$. Therefore, we also have $\langle\{M^{m_j}\}, T_2, S_2\rangle \xrightarrow[F]{N^{\bar{n}_{\bar{k}}}} \langle\{\mathrm{E}[\langle\rangle]^{m_j}\}, [m_j := d']T_2, S_2\rangle$. □

We aim at proving that any typable thread M^{m_j} that has a low-termination effect always presents the same behavior according to a *strong* bisimulation on low-equal states: if two continuations $M_1^{m_j}$ and $M_2^{m_j}$ of M^{m_j} are related, and if $M_1^{m_j}$ can perform an execution step over a certain state, then $M_2^{m_j}$ can perform the same low changes to any low-equal state in precisely one step, while the two resulting continuations are still related. This implies that any two computations of M^{m_j} under low-equal states should have the same "length", and in particular they are either both finite or both infinite. To this end, we design a reflexive binary relation on expressions with low-termination effects that is closed under the transitions of Guaranteed Transitions (Lemma B.2.6).

The definition of $\mathcal{T}^j_{F,low}$ is given in Figure B.1. Notice that it is a symmetric relation. In order to ensure that expressions that are related by $\mathcal{T}^j_{F,low}$ perform the same changes to the low memory, its definition requires that the references that are created or written using (potentially) different values are high.

Remark B.2.8. *If for some j, F and low we have $M_1\ \mathcal{T}^j_{F,low}\ M_2$ and $M_1 \in \boldsymbol{Val}$, then $M_2 \in \boldsymbol{Val}$.*

From the following lemma one can conclude that the relation $\mathcal{T}^{m_j}_{F,low}$ relates the possible outcomes of expressions that are typable with a low termination effect, and that perform a high read over low-equal memories.

Lemma B.2.9. *If there exist Γ, s, τ such that $\Gamma \vdash^j_F \mathrm{E}[(!\ a_{l,\theta})] : s, \tau$ with $s.t \preceq_F low$ and $l \not\preceq_{F\cup\lceil\mathrm{E}\rceil} low$, then for any values $V_0, V_1 \in \boldsymbol{Val}$ such that $\Gamma \vdash V_i : \theta$ we have $\mathrm{E}[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}[V_1]$.*

Definition B.2.7 ($\mathcal{T}^j_{F,low}$)**.** *We have that* $M_1\ \mathcal{T}^j_{F,low}\ M_2$ *if* $\Gamma \vdash^j_F M_1 : s_1, \tau$ *and* $\Gamma \vdash^j_F M_2 : s_2, \tau$ *for some* Γ*,* A*,* s_1*,* s_2 *and* τ *with* $s_1.t \preceq_F low$ *and* $s_2.t \preceq_F low$ *and one of the following holds:*

Clause 1. M_1 *and* M_2 *are both values, or*

Clause 2. $M_1 = M_2$*, or*

Clause 3. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *where* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$*, or*

Clause 4. $M_1 = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ *and* $M_2 = (\mathrm{ref}_{l,\theta}\ \bar{M}_2)$ *where* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$*, and* $l \not\preceq_F low$*, or*

Clause 5. $M_1 = (!\ \bar{M}_1)$ *and* $M_2 = (!\ \bar{M}_2)$ *where* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$*, or*

Clause 6. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$*, and* $\bar{N}_1\ \mathcal{T}^j_{F,low}\ \bar{N}_2$*, and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \mathrm{ref}_l$ *for some* θ *and* l *such that* $l \not\preceq_F low$*, or*

Clause 7. $M_1 = (\text{flow } F' \text{ in } \bar{M}_1)$ *and* $M_2 = (\text{flow } F' \text{ in } \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{T}^j_{F\cup F',low}\ \bar{M}_2$*.*

Figure B.1: The relation $\mathcal{T}^j_{F,low}$

Proof. By induction on the structure of E.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ a_{l,\theta})$. We have $V_0\ \mathcal{T}^j_{F,low}\ V_1$ by Clause 1.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]\ \boldsymbol{M})$. By App we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau} \xrightarrow[F,j]{\bar{s}'} \bar{\sigma}$ with $\bar{s}.r \preceq s.t.$ By Lemma B.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\boldsymbol{V}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule App we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}'', \bar{\tau}$ with $\bar{s}''.r \preceq s.t.$ By Lemma B.2.2, we have $l \preceq \bar{s}''.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{if}\ \mathbf{E_1}[(!\ a_{l,\theta})]\ \mathbf{then}\ \boldsymbol{M_t}\ \mathbf{else}\ \boldsymbol{M_f})$. By Cond we have that $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \mathsf{bool}$ with $\bar{s}.r \preceq s.t.$ By Lemma B.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F s.t$, which contradicts the assumption that both $s.t \preceq_F low$ and $l \not\preceq_{F\cup \mathrm{E}} low$ hold.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]; \boldsymbol{M})$. By Seq we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.t \preceq_F s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma B.1.4 and Clause 3 we can conclude.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{ref}_{l',\theta'}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Ref we have that $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.r = s.r \preceq_F l'$ and $\bar{s}.t = s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma B.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F low$. Therefore, $l' \not\preceq_F low$, and we conclude by Lemma B.1.4 and Clause 4.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ \mathbf{E_1}[a_{l,\theta}])$. By rule Der we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.t \preceq_F s.t.$ Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. We conclude by Lemma B.1.4 and Clause 5.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})] := \boldsymbol{M})$. By rule Ass we have that $\Gamma \vdash^j_F \mathrm{E}_1[a_{l,\theta}] : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l},\bar{n}_k}$ with $\bar{s}.t \preceq_F s.t$ and $\bar{s}.r \preceq_F \bar{l}$. Therefore $\bar{s}.t \preceq_F low$, and since $l \not\preceq_{F\cup \mathrm{E}} low$ implies $l \not\preceq_{F\cup \mathrm{E}_1} low$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. On the other hand, by Clause 2 we have $M\ \mathcal{T}^j_{F,low}\ M$. By Lemma B.2.2 we have $l \preceq \bar{s}.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F low$, since otherwise $l \preceq_{F\cup \mathrm{E}} low$. Therefore, we conclude by Lemma B.1.4 and Clause 6.

$\mathbf{E}[(!\ a_{l,\theta})] = (V := \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Ass we have that $\Gamma \vdash^j_F V : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l},\bar{n}_k}$, and $\Gamma \vdash^j_F \mathrm{E}_1[a_{l,\theta}] : \bar{s}', \theta$ with $\bar{s}'.t \preceq_F s.t$ and $\bar{s}'.r \preceq_F \bar{l}$. Therefore $\bar{s}'.t \preceq_F \mathit{low}$, and since $l \not\preceq_{F\cup \mathrm{E}} \mathit{low}$ implies $l \not\preceq_{F\cup \mathrm{E}_1} \mathit{low}$, then by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,\mathit{low}}\ \mathrm{E}_1[V_1]$. On the other hand, by Clause 2 we have $V\ \mathcal{T}^j_{F,\mathit{low}}\ V$. By Lemma B.2.2 we have $l \preceq \bar{s}'.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F \mathit{low}$, since otherwise $l \preceq_{F\cup \mathrm{E}} \mathit{low}$. We then conclude by Lemma B.1.4 and Clause 6.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{flow}\ F'\ \mathbf{in}\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule Flow we have $\Gamma \vdash^j_{F\cup F'} V : s, \tau$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F\cup F',\mathit{low}}\ \mathrm{E}_1[V_1]$, so we conclude by Lemma B.1.4 and Clause 7. Therefore $\bar{s}.t \preceq_F \mathit{low}$, and since $l \not\preceq_{F\cup \mathrm{E}} \mathit{low}$ implies $l \not\preceq_{F\cup \mathrm{E}_1} \mathit{low}$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,\mathit{low}}\ \mathrm{E}_1[V_1]$. By Lemma B.1.4 and Clause 8 we can conclude.

□

We can now prove that $\mathcal{T}^{m_j}_{F,\mathit{low}}$ behaves as a kind of "strong bisimulation":

Proposition B.2.10 (Strong Bisimulation for Low-Termination)**.**
If we have $M_1\ \mathcal{T}^{m_j}_{F,\mathit{low}}\ M_2$ and also $\langle W, \{{M_1}^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{M_1'}^{m_j}\}, T_1', S_1'\rangle$, with $\langle T_1, S_1\rangle =^{F\cup F',\mathit{low}} \langle T_2, S_2\rangle$ such that n is fresh for T_2 if $\bar{n}_{\bar{k}} \in \mathrm{dom}(T_1' - T_1)$ and a is fresh for S_2 if $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$, then there exist T_2', M_2' and S_2' such that $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{M_2'}^{m_j}\}, T_2', S_2'\rangle$ with $M_1'\ \mathcal{T}^{m_j}_{F,\mathit{low}}\ M_2'$ and $\langle T_1', S_1'\rangle =^{F,\mathit{low}} \langle T_2', S_2'\rangle$.

Proof. In the following, we use Subject Reduction (Theorem B.1.5) to guarantee that the termination effect of the expressions resulting from M_1 and M_2 is still low with respect to *low* and F. This, as well as typability (with the same type) for m_j, *low* and F, is a requirement for being in the $\mathcal{T}^{m_j}_{F,\mathit{low}}$ relation.

Clause 1. This case is not possible.

Clause 2. Here $M_1 = M_2$. By Guaranteed Transitions (Lemma B.2.6) there exist T_2', M_2' and S_2' such that $\langle W, \{M_2^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{M_2'}^{m_j}\}, T_2', S_2'\rangle$ with $\langle T_1', S_1'\rangle =^{F\cup F',\mathit{low}} \langle T_2', S_2'\rangle$.

- **$M_2' = M_1'$.** Then we have $M_1'\ \mathcal{T}^{m_j}_{F,\mathit{low}}\ M_2'$, by Clause 2 and Subject Reduction (Theorem B.1.5).
- **$M_2' \neq M_1'$.** Then by Splitting Computations (Lemma B.2.1) ($N^{\bar{n}_{\bar{k}}} = ()$) and we have two possibilities:
  (1) there exist E and $a_{l,\theta}$ such that $M_1' = \mathrm{E}[S_1(a_{l,\theta})]$, $F' = \lceil \mathrm{E} \rceil$, $M_2' = \mathrm{E}[S_2(a_{l,\theta})]$, $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$ and $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$. Since $S_1(a_{l,\theta}) \neq S_2(a_{l,\theta})$, we have $l \not\preceq_{F\cup F'} \mathit{low}$. Therefore, $M_1'\ \mathcal{T}^{m_j}_{F,\mathit{low}}\ M_2'$, by Lemma B.2.9 above.
  (2) there exists E such that $M_1' = \mathrm{E}[(\text{allowed } F' \text{ then } N_t \text{ else } N_f)]$, $F' = \lceil \mathrm{E} \rceil$, and $T_1(m_j) \neq T_2(m_j)$ with $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$ and $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$. Since $T_1(m_j) \neq T_2(m_j)$, we have $j \not\preceq_F \mathit{low}$, and by Lemma B.2.2 we have $s.t \not\preceq_F \mathit{low}$, which contradicts the assumption.

Clause 3. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ where $\bar{M}_1\ \mathcal{T}^{m_j}_{F,\mathit{low}}\ \bar{M}_2$. Then either:

- **$\bar{M}_1$ can compute.** In this case we have $M_1' = (\bar{M}_1'; \bar{N})$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Clause 3 and Subject Reduction (Theorem B.1.5) to conclude.
- **$\bar{M}_1$ is a value.** In this case $M_1' = \bar{N}$ and $F' = \emptyset$, $N^{\bar{n}_{\bar{k}}} = ()$ and $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$. We have $\bar{M}_2 \in \boldsymbol{Val}$ by Remark B.2.8, hence $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{N}^{m_j}\}, T_2, S_2\rangle$, and we conclude using Clause 2 and Subject Reduction (Theorem B.1.5).

Clause 4. Here $M_1 = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ and $M_2 = (\mathrm{ref}_{l,\theta}\ \bar{M}_2)$ where $\bar{M}_1\ \mathcal{T}^{m_j}_{F,low}\ \bar{M}_2$, and $l \not\preceq_F low$. There are two cases.

- **$\bar{M}_1$ can compute.** In this case we have $M_1' = (\mathrm{ref}_{l,\theta}\ \bar{M}_1)$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 4 to conclude.
- **$\bar{M}_1$ is a value.** In this case $M_1' = a_{l,\theta}$, with a fresh for S_1, $F' = \emptyset$, $N^{\bar{n}_{\bar{k}}} = ()$ and $\langle T_1', S_1'\rangle = \langle T_2, S_1 \cup \{a_{l,\theta} \mapsto \bar{M}_1\}\rangle$ (and therefore a is also fresh for S_2). Then $\bar{M}_2 \in \boldsymbol{Val}$ by Remark B.2.8, and therefore $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{a_{l,\theta}}^{m_j}\}, T_2', S_2 \cup \{a_{l,\theta} \mapsto \bar{M}_2\}\rangle$. If we let $S_2' = S_2 \cup \{a_{l,\theta} \mapsto \bar{M}_2\}$ then $\langle T_1', S_1'\rangle =^{F,low} \langle T_2', S_2'\rangle$ since $l \not\preceq_F low$. We conclude using Clause 1 and Subject Reduction (Theorem B.1.5).

Clause 5. Here $M_1 = (!\ \bar{M}_1)$ and $M_2 = (!\ \bar{M}_2)$ where $\bar{M}_1\ \mathcal{T}^{m_j}_{F,low}\ \bar{M}_2$. We distinguish two sub-cases.

- **$\bar{M}_1$ can compute.** In this case $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 5 to conclude.
- **$\bar{M}_1$ is a value.** Then $\bar{M}_1 = a_{l,\theta}$ and $M_1' \in \boldsymbol{Val}$, $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$, $F' = \emptyset$ and $N^{\bar{n}_{\bar{k}}} = ()$. By Remark B.2.8, $\bar{M}_2 \in \boldsymbol{Val}$, and since M_1 and M_2 have the same type, it must be a reference $a_{l',\theta}$. Then, $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{M_2}'^{m_j}\}, T_2, S_2\rangle$ with ${M_2}' \in \boldsymbol{Val}$, and we conclude using Clause 1 and Subject Reduction (Theorem B.1.5).

Clause 6. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1\ \mathcal{T}^{m_j}_{F,low}\ \bar{M}_2$ and $\bar{N}_1\ \mathcal{T}^{m_j}_{F,low}\ \bar{N}_2$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_{l,n_k}$ for some θ and l such that $l \not\preceq_F low$. We distinguish three sub-cases.

- **$\bar{M}_1$ can compute.** In this case $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 6 to conclude.
- **$\bar{M}_1$ is value, but $\bar{N}_1$ can compute.** In this case we have that $\langle W, \{\bar{N}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{N}_1'^{m_j}\}, T_1', S_1'\rangle$. By Remark B.2.8 also $\bar{M}_2 \in \boldsymbol{Val}$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 6 to conclude.
- **$\bar{M}_1$ and $\bar{M}_1$ are values.** Then $\bar{M}_1 = a_{l,\theta}$ and $M_1' = ()$, $\langle T_1', S_1'\rangle = \langle T_1, \{V \mapsto \bar{M}_1\}S_1\rangle$, $F' = \emptyset$ and $N^{\bar{n}_{\bar{k}}} = ()$. By Remark B.2.8, also $\bar{M}_2, \bar{N}_2 \in \boldsymbol{Val}$, and since $\bar{M}_1$ and $\bar{M}_2$ have the same type, $\bar{M}_2$ must be a reference $a_{l',\theta'}$. Then, $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{{M_2}'^{m_j}\}, T_2, \{V' \mapsto \bar{M}_2\}S_2\rangle$ with $\bar{M}_2' \in \boldsymbol{Val}$. Since $l \not\preceq_F low$, then we know that $\{V \mapsto \bar{M}_1\}S_1 =^{F\cup F',low} \{V' \mapsto \bar{M}_2\}S_2$. We conclude using Clause 1 and Subject Reduction (Theorem B.1.5).

Clause 7. Here we have $M_1 = (\mathrm{flow}\ \bar{F}\ \mathrm{in}\ \bar{M}_1)$ and $M_2 = (\mathrm{flow}\ \bar{F}\ \mathrm{in}\ \bar{M}_2)$ and $\bar{M}_1\ \mathcal{T}^{m_j}_{F\cup\bar{F},low}\ \bar{M}_2$. There are two cases.

- **$\bar{M}_1$ can compute.** In this case $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F'']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$ with $F' = \bar{F} \cup F''$. By induction hypothesis, we have $\langle W, \{\bar{M}_2^{m_j}\}, T_2, S_2\rangle \xrightarrow[F'']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_2'^{m_j}\}, T_2', S_2'\rangle$, and $M_1'\ \mathcal{T}^{m_j}_{F\cup\bar{F},low}\ M_2'$ and $\langle T_1', S_1'\rangle =^{F\cup\bar{F},low} \langle T_2', S_2'\rangle$. Notice that $\langle T_1', S_1'\rangle =^{F,low} \langle T_2', S_2'\rangle$. We use Subject Reduction (Theorem B.1.5) and Clause 7 to conclude.
- **$\bar{M}_1$ is a value.** In this case $M_1' = \bar{M}_1$, $F' = \emptyset$, $N^{\bar{n}_{\bar{k}}} = ()$ and $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$. Then $\bar{M}_2 \in \boldsymbol{Val}$ by Remark B.2.8, and so $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{\bar{n}_{\bar{k}}}} \langle W, \{\bar{M}_2^{m_j}\}, T_2, S_2\rangle$. We conclude using Clause 1 and Subject Reduction (Theorem B.1.5).

□

We have seen in Remark B.2.8 that when two expressions are related by $\mathcal{T}^j_{F,low}$ and one of them is a value, then the other one is also a value. From a semantical point of view, when an expression has reached a value it means that it has successfully completed its computation. We will now see that when two expressions are related by $\mathcal{T}^j_{F,low}$ and one of them is unable to *resolve* into a value, in any sequence of unrelated computation steps, then the other one is also unable to do so. We shall use the notion of *derivative of an expression* M:

Definition B.2.11 (Derivative of an Expression)**.** *We say that an expression M' is a* derivative *of an expression M if and only if*

- $M' = M$, *or*
- *there exist two states $\langle T_1, S_1\rangle$ and $\langle T'_1, S'_1\rangle$ and a derivative M'' of M such that, for some W, F, N^{n_k}:*

$$\langle W, M'', T_1, S_1\rangle \xrightarrow[F]{N^{n_k}} \langle W, M', T'_1, S'_1\rangle$$

Definition B.2.12 (Non-resolvable Expressions)**.** *We say that an expression M is* non-resolvable*, denoted $M\dagger$, if there is no derivative M' of M such that $M' \in \boldsymbol{Val}$.*

Lemma B.2.13. *If for some F, low and j we have that $M\ \mathcal{T}^j_{F,low}\ N$ and $M\dagger$, then $N\dagger$.*

Proof. Let us suppose that $\neg N\dagger$. That means that there exists a finite number of states $\langle T_1, S_1\rangle$, $\ldots$, $\langle T_n, S_n\rangle$, and $\langle T'_1, S'_1\rangle$, $\ldots$, $\langle T'_n, S'_n\rangle$, corresponding mappings of flow policies $W_1, \ldots W_n$, and of expressions N_1, $\ldots$, N_n such that

$$\begin{array}{rcl}
\langle W_1, \{N\}, T_1, S_1\rangle & \rightarrow & \langle W_1, \{N_1\}, T'_1, S'_1\rangle \text{ and} \\
\langle W_2, \{N_1\}, T_2, S_2\rangle & \rightarrow & \langle W_2, \{N_2\}, T'_2, S'_2\rangle \text{ and} \\
& \vdots & \\
\langle W_n, \{N_{n-1}\}, T_n, S_n\rangle & \rightarrow & \langle W_n, \{N_n\}, T'_n, S'_n\rangle
\end{array}$$

and such that $N_n \in \boldsymbol{Val}$. By Strong Bisimulation for Low-Terminating Threads (Proposition B.2.10), we have that there exists a finite number of states $\langle \bar{T}'_1, \bar{S}'_1\rangle$, $\ldots$, $\langle \bar{T}'_n, \bar{S}'_n\rangle$, corresponding mappings of flow policies $W_1, \ldots W_n$, and of expressions $\bar{M}_1$, $\ldots$, $\bar{M}_n$ such that

$$\begin{array}{rcl}
\langle W_1, \{M\}, T_1, S_1\rangle & \rightarrow & \langle W_1, \{M_1\}, \bar{T}'_1, \bar{S}'_1\rangle \text{ and} \\
\langle W_2, \{M_1\}, T_2, S_2\rangle & \rightarrow & \langle W_2, \{M_2\}, \bar{T}'_2, \bar{S}'_2\rangle \text{ and} \\
& \vdots & \\
\langle W_n, \{M_{n-1}\}, T_n, S_n\rangle & \rightarrow & \langle W_n, \{M_n\}, \bar{T}'_n, \bar{S}'_n\rangle
\end{array}$$

such that:

$$M_1\ \mathcal{T}^j_{F,low}\ \bar{N}_1, \text{ and } \ldots, \text{ and } M_n\ \mathcal{T}^j_{F,low}\ \bar{N}_n$$

By Remark B.2.8, we then have that $M_n \in \boldsymbol{Val}$. Since M_n is a derivative of M, we conclude that $\neg M\dagger$. □

The following lemma deduces operational "highness" of threads from that of its subexpressions.

Lemma B.2.14 (Composition of High Expressions)**.** *Suppose that M^{m_j} is typable in F. Then:*

1. *If $M = (M_1\ M_2)$ and M_1 is a syntactically (F, low, j)-high function and either*
   - *$M_1\dagger$ and ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, or*

- ${M_1}^{m_j}, {M_2}^{m_j} \in \mathcal{H}_{F,low}$,

then $M^{m_j} \in \mathcal{H}_{F,low}$.

2. *If* $M = (\text{if } M_1 \text{ then } M_t \text{ else } M_f)$ *and* ${M_1}^{m_j}, {M_t}^{m_j}, {M_f}^{m_j} \in \mathcal{H}_{F,low}$, *then* $M^{m_j} \in \mathcal{H}_{F,low}$.
3. *If* $M = (\text{ref}_{l,\theta}\ M_1)$ *and* $l \not\preceq_F low$ *and* ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, *then* $M^{m_j} \in \mathcal{H}_{F,low}$.
4. *If* $M = (M_1; M_2)$ *and either*
   - $M_1\dagger$ *and* ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, *or*
   - ${M_1}^{m_j}, {M_2}^{m_j} \in \mathcal{H}_{F,low}$,

   then $M^{m_j} \in \mathcal{H}_{F,low}$.
5. *If* $M = (M_1 := M_2)$ *and* M_1 *has type* $\theta\ \text{ref}_{l,n_k}$ *with* $l \not\preceq_F low$ *and either*
   - $M_1\dagger$ *and* ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, *or*
   - ${M_1}^{m_j}, {M_2}^{m_j} \in \mathcal{H}_{F,low}$,

   then $M^{m_j} \in \mathcal{H}_{F,low}$.
6. *If* $M = (\text{flow } F \text{ in } M_1)$ *and* ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, *then* $M^{m_j} \in \mathcal{H}_{F,low}$.
7. *If* $M = (\text{allowed } F \text{ then } M_t \text{ else } M_f)$ *and* ${M_t}^{m_j}, {M_f}^{m_j} \in \mathcal{H}_{F,low}$, *then* $M^{m_j} \in \mathcal{H}_{F,low}$.

Proof. We give the proof for the case where $M = (M_1\ M_2)$ and M_1 is a syntactically (F, low, j)-high function. The other cases are analogous or simpler.

$M_1\dagger$ and ${M_1}^{m_j} \in \mathcal{H}_{F,low}$. Let $\mathcal{F}$ be a set of threads that includes $\mathcal{H}_{F,low}$, and that contains the threads $(M_1\ M_2)^{m_j}$ provided that they are typable in F, and satisfy $M_1 \notin \textbf{\textit{Val}}$ and $M_1^{m_j} \in \mathcal{F}$ and M_1 is a (F, low, j)-high function. Assume that an application $M = (M_1\ M_2)$ such that $M_1\dagger$ and ${M_1}^{m_j} \in \mathcal{H}_{F,low}$ performs the transition $\langle W, M^{m_j}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, M'^{m_j}, T', S'\rangle$. We show that this implies $M'^{m_j}, N^{n_k} \in \mathcal{F}$ and $\langle T', S'\rangle =^{F,low} \langle T', S'\rangle$.

Since M_1 is non-resolvable, M_1 cannot be a value, and since M can compute, then also M_1 can compute. We then have $M' = (M_1'\ M_2)$ with $\langle W, {M_1}^{m_j}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, {M_1'}^{m_j}, T', S'\rangle$. Since ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, then also $M_1'^{m_j}, N^{n_k} \in \mathcal{H}_{F,low}$, thus $M_1'^{m_j}, N^{n_k} \in \mathcal{F}$, and $\langle T', S'\rangle =^{F,low} \langle T', S'\rangle$. By Subject Reduction (Theorem B.1.5), M_1' is a (F, low)-high function, and since $M_1\dagger$ then $M_1' \notin \textbf{\textit{Val}}$. Hence $M'^{m_j} \in \mathcal{F}$.

${M_1}^{m_j}, {M_2}^{m_j} \in \mathcal{H}_{F,low}$. Let $\mathcal{F}$ be a set of pools of threads that includes $\mathcal{H}_{F,low}$, and that contains threads $(M_1\ M_2)^{m_j}$ provided they are typable in F and satisfy $M_1^{m_j}, M_2^{m_j} \in \mathcal{F}$ and M_1 is a (F, low, j)-high function. Assume that such an application $M = (M_1\ M_2)$ performs the transition $\langle W, M^{m_j}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, M'^{m_j}, T', S'\rangle$. We show that this implies $M'^{m_j}, N^{n_k} \in \mathcal{F}$ and $\langle T', S'\rangle =^{F,low} \langle T', S'\rangle$.

M_1 and M_2 are values. Then $M_1 = (\lambda x.\bar{M_1})$, $M' = \{x \mapsto M_2\}\bar{M_1}$ and $N' = \langle\rangle$, $\langle T', S'\rangle = \langle T, S\rangle$. Since M_1 is a (F, low, j)-high function, then by Abs $\bar{M_1}$ is syntactically (F, low, j)-high, and by Substitution (Lemma B.1.3), also M' is syntactically (F, low, j)-high. Therefore, by High Expressions (Lemma B.2.5), $M'^{m_j} \in \mathcal{H}_{F,low}$.

M_1 can compute. Then $M' = (M_1'\ M_2)$ with $\langle W, {M_1}^{m_j}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, {M_1'}^{m_j}, T', S'\rangle$. Since ${M_1}^{m_j} \in \mathcal{H}_{F,low}$, then also $M_1'^{m_j}, N^{n_k} \in \mathcal{F}$ and $\langle T', S'\rangle =^{F,low} \langle T', S'\rangle$. By Subject Reduction (Theorem B.1.5) M_1' is a (F, low)-high function. Hence $M' \in \mathcal{F}$.

M_1 is a value but M_2 can compute. Then we have $M' = (M_1\ M_2')$ with $\langle W, {M_2}^{m_j}, T, S\rangle \xrightarrow[F']{N^{n_k}} \langle W, {M_2'}^{m_j}, T', S'\rangle$. Since ${M_2}^{m_j}, N^{n_k} \in \mathcal{H}_{F,low}$, then also ${M_2'}^{m_j}, N^{n_k} \in \mathcal{F}$ and $\langle T', S'\rangle =^{F,low} \langle T', S'\rangle$. Hence $M' \in \mathcal{F}$.

□

Lemma B.2.15. *If for some j, F and low we have that $M_1\ \mathcal{T}^j_{F,low}\ M_2$ and $M_1 \in \mathcal{H}_{F,low}$, then $M_2 \in \mathcal{H}_{F,low}$.*

Proof. By induction on the definition of $M_1\ \mathcal{T}^j_{F,low}\ M_2$.

Clause 1. Direct.

Clause 2. Direct.

Clause 3. Here $M_1 = (\bar{M_1}; \bar{N})$ and $M_2 = (\bar{M_2}; \bar{N})$ with $\bar{M_1}\ \mathcal{T}^j_{F,low}\ \bar{M_2}$. Clearly we have that $\bar{M_1} \in \mathcal{H}_{F,low}$, so by induction hypothesis, also $\bar{M_2} \in \mathcal{H}_{F,low}$. We distinguish two sub-cases:

- $\bar{N} \in \mathcal{H}_{F,low}$. Then, $\bar{M_2}, \bar{N} \in \mathcal{H}_{F,low}$. Therefore, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.
- $\bar{N} \notin \mathcal{H}_{F,low}$. Then $\bar{M_1}\dagger$, and by Lemma B.2.13 also $\bar{M_2}\dagger$. Therefore, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.

Clause 4. Here $M_1 = (\mathrm{ref}_{l,\theta}\ \bar{M_1})$ and $M_2 = (\mathrm{ref}_{l,\theta}\ \bar{M_2})$ where $\bar{M_1}\ \mathcal{T}^j_{F,low}\ \bar{M_2}$, and $l \not\preceq_F low$. Clearly we have that $\bar{M_1} \in \mathcal{H}_{F,low}$, so by induction hypothesis also $\bar{M_2} \in \mathcal{H}_{F,low}$. Therefore, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.

Clause 5. Here $M_1 = (!\ \bar{M_1})$ and $M_2 = (!\ \bar{M_2})$ where $\bar{M_1}\ \mathcal{T}^j_{F,low}\ \bar{M_2}$. Clearly we have that $\bar{M_1} \in \mathcal{H}_{F,low}$, so by induction hypothesis also $\bar{M_2} \in \mathcal{H}_{F,low}$. This implies that $\bar{M_2} \in \mathcal{H}_{F,low}$.

Clause 6. Here we have $M_1 = (\bar{M_1} := \bar{N_1})$ and $M_2 = (\bar{M_2} := \bar{N_2})$ where $\bar{M_1}\ \mathcal{T}^j_{F,low}\ \bar{M_2}$, and $\bar{M_1}, \bar{M_2}$ both have type $\theta\ \mathrm{ref}_l$ for some θ and l such that $l \not\preceq_F low$, and $\bar{N_1}\ \mathcal{T}^j_{F,low}\ \bar{N_2}$. Clearly we have that $\bar{M_1} \in \mathcal{H}_{F,low}$, so by induction hypothesis also $\bar{M_2} \in \mathcal{H}_{F,low}$. We distinguish two sub-cases:

- $\bar{N_2} \in \mathcal{H}_{F,low}$. Then, $\bar{M_2}, \bar{N_2} \in \mathcal{H}_{F,low}$ where $\bar{M_2}$ has type $\theta\ \mathrm{ref}_{l,n_k}$ for some θ and l such that $l \not\preceq_F low$. Therefore, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.
- $\bar{N_2} \notin \mathcal{H}_{F,low}$. Then $\bar{M_1}\dagger$, and by Lemma B.2.13 also $\bar{M_2}\dagger$. Therefore, since $\bar{M_2}$ has type $\theta\ \mathrm{ref}_{l,n_k}$ for some θ and l such that $l \not\preceq_F low$, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.

Clause 7. Here we have $M_1 = (\mathrm{flow}\ F'\ \mathrm{in}\ \bar{M_1})$ and $M_2 = (\mathrm{flow}\ F'\ \mathrm{in}\ \bar{M_2})$ with $\bar{M_1}\ \mathcal{T}^j_{F\cup F',low}\ \bar{M_2}$. Clearly we have that $\bar{M_1} \in \mathcal{H}_{F,low}$, so by induction hypothesis also $\bar{M_2} \in \mathcal{H}_{F,low}$. Therefore, by Composition of High Expressions (Lemma B.2.14) we have that $M_2 \in \mathcal{H}_{F,low}$.

□

B.2.3 Behavior of Typable Low Expressions

In this second phase of the proof, we consider the general case of threads that are typable with any termination level. As in the previous subsection, we show that a typable expression behaves as a strong bisimulation, provided that it is operationally low. For this purpose, we make use of the properties identified for the class of low-terminating expressions by allowing only these to be followed by low-writes. Conversely, high-terminating expressions can only be followed by high-expressions (see Definitions B.2.3 and B.2.4).

Lemma B.2.16 (High Threads might Split)**.** *Consider a thread M^{m_j} for which there exist Γ, F, s and τ such that $\Gamma \vdash^j_F M : s, \tau$ and suppose that $M = \mathrm{E}[(\text{allowed } F' \text{ then } N_t \text{ else } N_f)]$ with $j \not\preceq_F low$. Then $M^{m_j} \in \mathcal{H}_{F,low}$.*

Proof. By induction on the structure of E, using Lemma B.2.2 and Lemma B.2.5 and Lemma B.2.2. Consider $M = \mathrm{E}[M_0]$, where $M_0 = (\text{allowed } F' \text{ then } N_t \text{ else } N_f)$.

$\mathbf{E}[M_0] = M_0$. Then, using rule ALLOW, we have that $\Gamma \vdash^j_F (\text{allowed } F' \text{ then } N_t \text{ else } N_f) : s, \tau$ where $\Gamma \vdash^j_F N_t : s_t, \tau$, $\Gamma \vdash^j_F N_f : s_f, \tau$ and $j \preceq_F s_t.w, s_f.w$. This means $s_t.w, s_f.w \not\preceq_F low$, so by Lemma B.2.5, then ${N_t}^{m_j}, {N_f}^{m_j} \in \mathcal{H}_{F,low}$. By Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (\mathbf{E_1}[M_0]\ M_1)$. Then by rule APP we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \tau_1 \xrightarrow[F,j]{s_1'} \sigma_1$ and $\Gamma \vdash^j_F M_1 : s_1'', \tau_1$ with $s_1.r \preceq_F s_1'.w$ and $s_1.r \preceq_F s_1''.w$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $\mathrm{E}_1[M_0]$ is a syntactically (F, low, j)-high function and M_1 is (F, low, j)-high. By High Expressions (Lemma B.2.5) we have ${M_1}^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (V\ \mathbf{E_1}[M_0])$. Then by APP we have $\Gamma \vdash^j_F V : s_1, \tau_1 \xrightarrow[F,j]{s_1'} \sigma_1$ and $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1'', \tau_1$ with $s_1''.t \preceq_F s_1'.w$ and $s_1'.t \preceq_F s_1''.w$. By Lemma B.2.2 we have $j \preceq s_1''.t$, which implies that $j \preceq_F s_1''.t$ and $s_1''.t \not\preceq_F low$, and so $s_1'.w \not\preceq_F low$. Therefore, $s_1'.w \not\preceq_F low$, and $s_1''.w \not\preceq_F low$, which means that V is a syntactically (F, low, j)-high function and $\mathrm{E}_1[M_0]$ is (F, low, j)-high. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (\mathbf{if}\ \mathbf{E_1}[M_0]\ \mathbf{then}\ M_t\ \mathbf{else}\ M_f)$. Then by rule COND we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \mathsf{bool}$, and $\Gamma \vdash^j_F M_t : s_1', \tau_1$ and $\Gamma \vdash^j_F M_f : s_1'', \tau_1$ with $s_1.t \preceq_F s_1'.w, s_1''.w$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $s_1'.w, s_1''.w \not\preceq_F low$, so by High Expressions (Lemma B.2.5) we have ${M_t}^{m_j}, {M_t}^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (\mathbf{E_1}[M_0]; M_1)$. Then by SEQ we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \tau_1$ and $\Gamma \vdash^j_F M_1 : s_1', \tau_1'$ with $s_1.t \preceq_F s_1'.w$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $s_1'.w \not\preceq_F low$, and by High Expressions (Lemma B.2.5) we have ${M_1}^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (\mathbf{ref}_{l,\theta}\ \mathbf{E_1}[M_0])$. Then by REF we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \theta$ with $s_1.t \preceq_F l$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $l \not\preceq_F low$, and by induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (!\ \mathbf{E_1}[M_0])$. Easy, by induction hypothesis.

$\mathbf{E}[M_0] = (\mathbf{E_1}[M_0] := M_1)$. Then by ASS we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \theta\ \mathrm{ref}_{\bar{l}}$ and $\Gamma \vdash^j_F M_1 : s_1', \tau_1$ with $s_1.t \preceq_F s_1'.w$ and $s_1.t \preceq_F \bar{l}$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $\bar{l} \not\preceq_F low$ and $s_1'.w \not\preceq_F low$. Hence, by High Expressions (Lemma B.2.5) we have ${M_1}^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (V := \mathbf{E_1}[M_0])$. Then by ASS we have $\Gamma \vdash^j_F V : s_1, \theta\ \mathrm{ref}_{\bar{l},n_{\bar{k}}}$ and $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1', \tau_1$ with $s_1'.t \preceq_F \bar{l}$. By Lemma B.2.2 we have $j \preceq s_1'.t$, which implies that $j \preceq_F s_1'.t$ and $s_1'.t \not\preceq_F low$. Therefore, $\bar{l} \not\preceq_F low$, and by induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

Definition B.2.17 ($\mathcal{R}^j_{F,low}$)**.** *We have that* $M_1\ \mathcal{R}^j_{F,low}\ M_2$ *if* $\Gamma \vdash^j_F M_1 : s_1, \tau$ *and* $\Gamma \vdash^j_F M_2 : s_2, \tau$ *for some* Γ, s_1, s_2 *and* τ *and one of the following holds:*

Clause 1'. ${M_1}^{m_j}, {M_2}^{m_j} \in \mathcal{H}_{F,low}$, *or*

Clause 2'. $M_1 = M_2$, *or*

Clause 3'. $M_1 = (\text{if } \bar{M}_1 \text{ then } \bar{N}_t \text{ else } \bar{N}_f)$ *and* $M_2 = (\text{if } \bar{M}_2 \text{ then } \bar{N}_t \text{ else } \bar{N}_f)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *and* ${\bar{N}_t}^{m_j}, {\bar{M}_f}^{m_j} \in \mathcal{H}_{F,low}$, *or*

Clause 4'. $M_1 = (\bar{M}_1\ \bar{N}_1)$ *and* $M_2 = (\bar{M}_2\ \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1^{m_j}, \bar{N}_2^{m_j} \in \mathcal{H}_{F,low}$, *and* $\bar{M}_1, \bar{M}_2$ *are syntactically* (F, low, j)*-high functions, or*

Clause 5'. $M_1 = (\bar{M}_1\ \bar{N}_1)$ *and* $M_2 = (\bar{M}_2\ \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1\ \mathcal{R}^j_{F,low}\ \bar{N}_2$, *and* $\bar{M}_1, \bar{M}_2$ *are syntactically* (F, low, j)*-high functions, or*

Clause 6'. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *and* $\bar{N}^{m_j} \in \mathcal{H}_{F,low}$, *or*

Clause 7'. $M_1 = (\bar{M}_1; \bar{N})$ *and* $M_2 = (\bar{M}_2; \bar{N})$ *with* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$, *or*

Clause 8'. $M_1 = (\text{ref}_{l,\theta}\ \bar{M}_1)$ *and* $M_2 = (\text{ref}_{l,\theta}\ \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *and* $l \not\preceq_F low$, *or*

Clause 9'. $M_1 = (!\ \bar{M}_1)$ *and* $M_2 = (!\ \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *or*

Clause 10'. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1^{m_j}, \bar{N}_2^{m_j} \in \mathcal{H}_{F,low}$, *and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \text{ref}_{l,n_k}$ *for some* θ *and* l *such that* $l \not\preceq_F low$, *or*

Clause 11'. $M_1 = (\bar{M}_1 := \bar{N}_1)$ *and* $M_2 = (\bar{M}_2 := \bar{N}_2)$ *with* $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$, *and* $\bar{N}_1\ \mathcal{R}^j_{F,low}\ \bar{N}_2$, *and* $\bar{M}_1, \bar{M}_2$ *both have type* $\theta\ \text{ref}_{l,n_k}$ *for some* θ *and* l *such that* $l \not\preceq_F low$, *or*

Clause 12'. $M_1 = (\text{flow } F' \text{ in } \bar{M}_1)$ *and* $M_2 = (\text{flow } F' \text{ in } \bar{M}_2)$ *with* $\bar{M}_1\ \mathcal{R}^j_{F\cup F',low}\ \bar{M}_2$.

Figure B.2: The relation $\mathcal{R}^j_{F,low}$

$\mathbf{E}[M_0] = (\text{flow } F' \text{ in } \mathbf{E}_1[M_0])$. Then by rule FLOW we have $\Gamma \vdash^j_{F\cup F'} \mathrm{E}_1[M_0] : s_1, \tau_1$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F\cup F',low}$, which implies $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), we conclude that $M^{m_j} \in \mathcal{H}_{F,low}$.

$\mathbf{E}[M_0] = (\text{flow } \mathbf{E}_1[M_0] \text{ in } M_1)$. Then by FLOW we have that $\Gamma \vdash^j_F \mathrm{E}_1[M_0] : s_1, \mathsf{flow}_{\bar{F}}$ and $\Gamma \vdash^j_{F\cup\bar{F}} M_1 : s_1', \tau_1$ with $s_1.t \preceq_F s_1'.w$. By Lemma B.2.2 we have $j \preceq s_1.t$, which implies that $j \preceq_F s_1.t$ and $s_1.t \not\preceq_F low$. Therefore, $s_1'.w \not\preceq_F low$, and by High Expressions (Lemma B.2.5) we have ${M_1}^{m_j} \in \mathcal{H}_{F\cup\bar{F},low}$. By induction hypothesis $\mathrm{E}_1[M_0]^{m_j} \in \mathcal{H}_{F,low}$. Then, by Composition of High Expressions (Lemma B.2.14), $M^{m_j} \in \mathcal{H}_{F,low}$.

□

We now design a binary relation on expressions that uses $\mathcal{T}^j_{F,low}$ to ensure that high-terminating expressions are always followed by operationally high ones. The definition of $\mathcal{R}^j_{G,F,low}$, abbreviated $\mathcal{R}^j_{F,low}$ when the global flow policy is G, is given in Figure B.2. The flow policy F is assumed to contain G. Notice that it is a symmetric relation. In order to ensure that expressions that are related by $\mathcal{R}^j_{F,low}$ perform the same changes to the low memory, its definition requires that the references that are created or written using (potentially) different values are high, and that the body of the functions that are applied are syntactically high.

Remark B.2.18. *If* $M_1\ \mathcal{T}^j_{F,low}\ M_2$*, then* $M_1\ \mathcal{R}^j_{F,low}\ M_2$*.*

The above remark is used to prove the following lemma.

Lemma B.2.19. *If for some* j*,* F *and* low *we have that* $M_1\ \mathcal{R}^j_{F,low}\ M_2$ *and* $M_1 \in \mathcal{H}_{F,low}$*, then* $M_2 \in \mathcal{H}_{F,low}$*.*

Proof. By induction on the definition of $M_1\ \mathcal{R}^j_{F,low}\ M_2$, using Lemma B.2.15. □

We have seen in Splitting Computations (Lemma B.2.1) that two computations of the same expression can split only if the expression is about to read a reference that is given different values by the memories in which they compute. In Lemma B.2.20 we saw that the relation $\mathcal{T}^j_{F,low}$ relates the possible outcomes of expressions that are typable with a low termination effect. Finally, from the following lemma one can conclude that the above relation $\mathcal{R}^j_{F,low}$ relates the possible outcomes of typable expressions in general.

Lemma B.2.20. *If there exist* Γ, A, s, τ *such that* $\Gamma \vdash^j_F \mathrm{E}[(!\ a_{l,\theta})] : s, \tau$ *with* $l \not\preceq_{F \cup \lceil \mathrm{E} \rceil}$ *low, then for any values* $V_0, V_1 \in \boldsymbol{Val}$ *such that* $\Gamma \vdash V_i : \theta$ *we have* $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$*.*

Proof. By induction on the structure of E using Lemma B.1.4, Lemma B.2.9, Lemma B.2.5.

$\mathbf{E}[(!\ a_{l,\theta})] = (!\ a_{l,\theta})$. We have $V_0\ \mathcal{R}^j_{F,low}\ V_1$ by Clause 1'.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]\ M)$. By rule App we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau} \xrightarrow[F,j]{\bar{s}'} \bar{\sigma}$ and $\Gamma \vdash^j_F M : \bar{s}'', \bar{\tau}$ with $\bar{s}.r \preceq_F \bar{s}'.w$ and $\bar{s}.t \preceq_F \bar{s}''.w$. By Lemma B.2.2, we have $l \preceq \bar{s}.r$. Therefore $l \preceq_F \bar{s}'.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil}$ *low* (therefore $l \not\preceq_F$ *low*), then $\bar{s}'.w \not\preceq_F$ *low*, that is $\mathrm{E}_1[(!\ a_{l,\theta})]$ is a syntactically (F, low, j)-high function. By Lemma B.1.4, the same holds for $\mathrm{E}_1[V_0]$ and $\mathrm{E}_1[V_1]$. By induction hypothesis we conclude that $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$.

- **$\bar{s}.t \not\preceq_F$ *low*.** Then $\bar{s}''.w \not\preceq_F$ *low* (and also $\bar{s}''.w \not\preceq$ *low*) so by High Expressions (Lemma B.2.5) we have $M^{m_j} \in \mathcal{H}_{F,low}$. Therefore, we conclude $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 4' and Lemma B.1.4.
- **$\bar{s}.t \preceq_F$ *low*.** Then by Lemma B.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. Since $M\ \mathcal{R}^j_{F,low}\ M$ by Clause 2', we conclude that $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 5' and Lemma B.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (V\ \mathbf{E_1}[(!\ a_{l,\theta})])$. By rule App we have that $\Gamma \vdash^j_F V : \bar{s}, \bar{\tau} \xrightarrow[F,j]{\bar{s}'} \bar{\sigma}$ and $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}'', \bar{\tau}$ with $\bar{s}''.r \preceq_F \bar{s}'.w$. By Lemma B.2.2, we have $l \preceq \bar{s}''.r$, and so $l \preceq_F \bar{s}'.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil}$ *low* (therefore $l \not\preceq_F$ *low*), then $\bar{s}'.w \not\preceq_F$ *low*, that is V is a syntactically (F, low, j)-high function. By Clause 1 we have $V\ \mathcal{T}^j_{F,low}\ V$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. Therefore we conclude that $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 5' and Lemma B.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{if}\ \mathbf{E_1}[(!\ a_{l,\theta})]\ \mathbf{then}\ M_t\ \mathbf{else}\ M_f)$. By Cond we have that $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \mathsf{bool}$, and $\Gamma \vdash^j_F M_t : \bar{s}_t, \bar{\tau}$ and $\Gamma \vdash^j_F M_f : \bar{s}_f, \bar{\tau}$ with $\bar{s}.r \preceq_F \bar{s}_t.w, \bar{s}_f.w$. By Lemma B.2.2, we have $l \preceq \bar{s}.r$ and so $l \preceq_F \bar{s}_t.w, \bar{s}_f.w$. Since by hypothesis $l \not\preceq_{F \cup \lceil \mathrm{E}_1 \rceil}$ *low* (therefore $l \not\preceq_F$ *low*), then $\bar{s}_t.w \not\preceq_F$ *low* and $\bar{s}_f.w \not\preceq_F$ *low*. This implies that ${M_t}^{m_j}, {M_f}^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. Therefore we conclude that $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 3' and Lemma B.1.4.

$\mathbf{E}[(!\ a_{l,\theta})] = (\mathbf{E_1}[(!\ a_{l,\theta})]; M)$. By Seq we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ and $\Gamma \vdash^j_F M : \bar{s}', \bar{\tau}'$ with $\bar{s}.t \preceq_F \bar{s}'.w$.

- **$\bar{s}.t \not\preceq_F$ *low*.** Then $\bar{s}'.w \not\preceq_F$ *low* so by High Expressions (Lemma B.2.5) we have $M^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. Then, $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 6' and Lemma B.1.4.

$\bar{s}.t \preceq_F \mathit{low}$. Then by Lemma B.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. Therefore, we conclude using Clause 7' and Lemma B.1.4.

$\mathbf{E[(!\ a_{l,\theta})] = (ref_{\bar{l},\bar{\theta}}\ E_1[(!\ a_{l,\theta})])}$. By Ref we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$ with $\bar{s}.r = s.r \preceq_F \bar{l}$ and $\bar{s}.t = s.t$. Therefore, since $l \not\preceq_{F\cup \mathrm{E}} \mathit{low}$ implies $l \not\preceq_{F\cup \mathrm{E}_1} \mathit{low}$, then by induction hypothesis we have $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. By Lemma B.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F \mathit{low}$. Therefore, $\bar{l} \not\preceq_F \mathit{low}$, and we conclude by Lemma B.1.4 and Clause 8'.

$\mathbf{E[(!\ a_{l,\theta})] = (!\ E_1[(!\ a_{l,\theta})])}$. By rule Der we have $\Gamma \vdash^j_F \mathrm{E}_1[(!\ a_{l,\theta})] : \bar{s}, \bar{\tau}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. We conclude by Lemma B.1.4 and Clause 9'.

$\mathbf{E[(!\ a_{l,\theta})] = (E_1[(!\ a_{l,\theta})] := M)}$. By rule Ass we have that $\Gamma \vdash^j_F \mathrm{E}_1[a_{l,\theta}] : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l},\bar{n}_k}$ with $\bar{s}.r \preceq_F \bar{l}$ and $\bar{s}.t \preceq_F \bar{s}'.w$. By Lemma B.2.2 we have $l \preceq s.r$, so $s.r \not\preceq_F \mathit{low}$ and so $\bar{l} \not\preceq_F \mathit{low}$.

- $\bar{s}.t \not\preceq_F \mathit{low}$. Then $\bar{s}'.w \not\preceq_F \mathit{low}$ so by High Expressions (Lemma B.2.5) we have $M^{m_j} \in \mathcal{H}_{F,low}$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. Then, $\mathrm{E}[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}[V_1]$ by Clause 10' and Lemma B.1.4.
- $\bar{s}.t \preceq_F \mathit{low}$. Then by Lemma B.2.9 we have $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F,low}\ \mathrm{E}_1[V_1]$. Therefore, we conclude using Lemma B.1.4, Clause 11' and Clause 2' (regarding M).

$\mathbf{E[(!\ a_{l,\theta})] = (V := E_1[(!\ a_{l,\theta})])}$. By rule Ass we have that $\Gamma \vdash^j_F V : \bar{s}, \bar{\theta}\ \mathrm{ref}_{\bar{l},\bar{n}_k}$, $\Gamma \vdash^j_F \mathrm{E}_1[a_{l,\theta}] : \bar{s}', \theta$ with $\bar{s}'.r \preceq_F \bar{l}$. By Lemma B.2.2 we have $l \preceq \bar{s}'.r$, so $l \preceq_F \bar{l}$. Then, we must have $\bar{l} \not\preceq_F \mathit{low}$, since otherwise $l \preceq_{F\cup \mathrm{E}} \mathit{low}$. By Clause 1 we have that $V\ \mathcal{T}^j_{F,low}\ V$, and by induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{R}^j_{F,low}\ \mathrm{E}_1[V_1]$. We then conclude by Lemma B.1.4 and Clause 11'.

$\mathbf{E[(!\ a_{l,\theta})] = (flow\ F'\ in\ E_1[(!\ a_{l,\theta})])}$. By rule Flow we have $\Gamma \vdash^j_{F\cup F'} V : s, \tau$. By induction hypothesis $\mathrm{E}_1[V_0]\ \mathcal{T}^j_{F\cup F',low}\ \mathrm{E}_1[V_1]$, so we conclude by Lemma B.1.4 and Clause 12'.

□

We now state a crucial result of the paper: the relation $\mathcal{R}^j_{F,low}$ is a sort of "strong bisimulation".

Proposition B.2.21 (Strong Bisimulation for Typable Low Threads)**.**
If $M_1\ \mathcal{R}^j_{F,low}\ M_2$ *and* $M_1 \notin \mathcal{H}_{F,low}$ *and* $\langle W, \{{M_1}^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{{M_1'}^{m_j}\}, T_1', S_1'\rangle$, *with* $\langle T_1, S_1\rangle =^{F\cup F', low} \langle T_2, S_2\rangle$ *such that* n *is fresh for* T_2 *if* $n \in \mathrm{dom}(T_1' - T_1)$ *and* a *is fresh for* S_2 *if* $a_{l,\theta} \in \mathrm{dom}(S_1' - S_1)$, *then there exist* T_2', M_2' *and* S_2' *such that* $\langle W, \{{M_2}^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{{M_2'}^{m_j}\}, T_2', S_2'\rangle$ *with* $M_1'\ \mathcal{R}^j_{F,low}\ M_2'$ *and* $\langle T_1', S_1'\rangle =^{F,low} \langle T_2', S_2'\rangle$.

Proof. We use Subject Reduction (Theorem B.1.5) to guarantee typability (with the same type) for m_j, *low* and F, which is a requirement for being in the $\mathcal{R}^j_{F,low}$ relation. We also use the Strong Bisimulation for Low Terminating Threads Lemma (Lemma B.2.10).

Clause 1'. This case is excluded by assumption.

Clause 2'. Here $M_1 = M_2$. By Guaranteed Transitions (Lemma B.2.6) there exist T_2', M_2' and S_2' such that $\langle W, \{M_2^{m_j}\}, T_2, S_2\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{{M_2'}^{m_j}\}, T_2', S_2'\rangle$ with $\langle T_1', S_1'\rangle =^{F\cup F', low} \langle T_2', S_2'\rangle$.

- $\mathbf{M_2' = M_1'}$. Then we have $M_1'\ \mathcal{R}^j_{F,low}\ M_2'$, by Clause 2' and Subject Reduction (Theorem B.1.5).
- $\mathbf{M_2' \neq M_1'}$. Then by Splitting Computations (Lemma B.2.1) we have that $(N^{n_k} = ())$ and we have two possibilities:
  (1) there exists E and $a_{l,\theta}$ such that $F' = \lceil \mathrm{E} \rceil$, $M_1' = \mathrm{E}[S_1(a_{l,\theta})]$, $M_2' = \mathrm{E}[S_2(a_{l,\theta})]$, $\langle T_1', S_1'\rangle = \langle T_1, S_1\rangle$ and $\langle T_2', S_2'\rangle = \langle T_2, S_2\rangle$. Since $S_1(a_{l,\theta}) \neq S_2(a_{l,\theta})$, we have $l \not\preceq_{F\cup F'}$

low. Therefore, $M_1' \; \mathcal{R}^j_{F,low} \; M_2'$, by Lemma B.2.20 above.
(2) there exists E such that $M_1' = \mathrm{E}[(\text{allowed } F' \text{ then } N_t \text{ else } N_f)]$, $F' = \lceil \mathrm{E} \rceil$, and $T_1(m_j) \neq T_2(m_j)$ with $\langle T_1', S_1' \rangle = \langle T_1, S_1 \rangle$ and $\langle T_2', S_2' \rangle = \langle T_2, S_2 \rangle$. Since $T_1(m_j) \neq T_2(m_j)$, we have $j \not\preceq_F low$, and by Lemma B.2.16 $M_1 \in \mathcal{H}_{F,low}$, which contradicts our assumption.

Clause 3'. Here we have $M_1 = (\text{if } \bar{M}_1 \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ and $M_2 = (\text{if } \bar{M}_2 \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ with $\bar{M}_1 \; \mathcal{R}^j_{F,low} \; \bar{M}_2$ and $\bar{M}_t^{m_j}, \bar{M}_f^{m_j} \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1^{m_j} \notin \mathcal{H}_{F,low}$, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Therefore, $M_1' = (\text{if } \bar{M}_1' \text{ then } \bar{M}_t \text{ else } \bar{M}_f)$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use the induction hypothesis, Clause 3' and Subject Reduction (Theorem B.1.5) to conclude.

Clause 4'. Here $M_1 = (\bar{M}_1 \; \bar{N}_1)$ and $M_2 = (\bar{M}_2 \; \bar{N}_2)$ with $\bar{M}_1 \; \mathcal{R}^j_{F,low} \; \bar{M}_2$, $\bar{M}_1$ and $\bar{M}_2$ are syntactically (F, low, j)-high functions, and $\bar{N}_1^{m_j}, \bar{N}_2^{m_j} \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1$ can compute, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Therefore, $M_1' = (\bar{M}_1' \; \bar{N}_1)$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use the induction hypothesis, Clause 4' and Subject Reduction (Theorem B.1.5) to conclude.

Clause 5'. Here $M_1 = (\bar{M}_1 \; \bar{N}_1)$ and $M_2 = (\bar{M}_2 \; \bar{N}_2)$ with $\bar{M}_1 \; \mathcal{T}^j_{F,low} \; \bar{M}_2$, $\bar{M}_1$ and $\bar{M}_2$ are syntactically (F, low, j)-high functions, and $\bar{N}_1 \; \mathcal{R}^j_{F,low} \; \bar{N}_2$. We distinguish two sub-cases:

- $\bar{M}_1$ **can compute.** In this case there exists $\bar{M}_1'$ performing the transition $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use Lemma B.2.10, Subject Reduction (Theorem B.1.5) and Clause 5' to conclude.
- $\bar{M}_1$ **is a value.** Then by Remark B.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. We can assume that $\bar{N}_1^{m_j}, \bar{N}_2^{m_j} \notin \mathcal{H}_{F,low}$, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Then, $\bar{N}_1$ can compute, and so there exist $\bar{N}_1'$ such that $\langle W, \{\bar{N}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{N}_1'^{m_j}\}, T_1', S_1' \rangle$ with $M_1' = (\bar{M}_1 \; \bar{N}_1')$. We use the induction hypothesis, Clause 5' and Subject Reduction (Theorem B.1.5) to conclude.

Clause 6'. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ where $\bar{M}_1 \; \mathcal{R}^j_{F,low} \; \bar{M}_2$ and $\bar{N}^{m_j} \in \mathcal{H}_{F,low}$. We can assume that $\bar{M}_1^{m_j} \notin \mathcal{H}_{F,low}$, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Therefore, we have $M_1' = (\bar{M}_1'; \bar{N})$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use the induction hypothesis, Clause 6' and Subject Reduction (Theorem B.1.5) to conclude.

Clause 7'. Here $M_1 = (\bar{M}_1; \bar{N})$ and $M_2 = (\bar{M}_2; \bar{N})$ with $\bar{M}_1 \; \mathcal{T}^j_{F,low} \; \bar{M}_2$. We distinguish two sub-cases:

- $\bar{M}_1$ **can compute.** In this case there exists $\bar{M}_1'$ performing the transition $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use Lemma B.2.10, Subject Reduction (Theorem B.1.5) and Clause 7' to conclude.
- $\bar{M}_1$ **is a value.** Then $M_1' = \bar{N}$, $F = \emptyset$, $N^{n_k} = ()$ and $\langle T_1', S_1' \rangle = \langle T_1, S_1 \rangle$. By Remark B.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. Then, we have $\langle W, \{M_2^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{N}^{m_j}\}, T_1', S_1' \rangle$. We conclude using Lemma B.2.10 and Clause 2'.

Clause 8'. Here $M_1 = (\mathrm{ref}_{l,\theta} \; \bar{M}_1)$ and $M_2 = (\mathrm{ref}_{l,\theta} \; \bar{M}_2)$ where $\bar{M}_1 \; \mathcal{R}^j_{F,low} \; \bar{M}_2$, and $l \not\preceq_F low$. We can assume that $\bar{M}_1^{m_j} \notin \mathcal{H}_{F,low}$, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Then, $\bar{M}_1$ can compute, and $M_1' = (\mathrm{ref}_{l,\theta} \; \bar{M}_1)$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1 \rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1' \rangle$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 8' to conclude.

Clause 9'. Here $M_1 = (!\ \bar{M}_1)$ and $M_2 = (!\ \bar{M}_2)$ where $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$. We know that $\bar{M}_1$ can compute, since otherwise ${M_1}^{m_j} \in \mathcal{H}_{F,low}$. Then, we have $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Subject Reduction (Theorem B.1.5) and Clause 9' to conclude.

Clause 10'. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1\ \mathcal{R}^j_{F,low}\ \bar{M}_2$, and $\bar{N}_1^{m_j}, \bar{N}_2^{m_j} \in \mathcal{H}_{F,low}$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_{l,n_k}$ for some θ and l such that $l \not\preceq_F low$. We can assume that $\bar{M}_1$ can compute, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). Therefore, $M_1' = (\bar{M}_1' := \bar{N}_1)$ with $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use the induction hypothesis, Clause 10' and Subject Reduction (Theorem B.1.5) to conclude.

Clause 11'. Here we have $M_1 = (\bar{M}_1 := \bar{N}_1)$ and $M_2 = (\bar{M}_2 := \bar{N}_2)$ where $\bar{M}_1\ \mathcal{T}^j_{F,low}\ \bar{M}_2$, and $\bar{M}_1, \bar{M}_2$ both have type $\theta\ \mathrm{ref}_{l,n_k}$ for some θ and l such that $l \not\preceq_F low$, and $\bar{N}_1\ \mathcal{R}^j_{F,low}\ \bar{N}_2$. We can assume that M_1 cannot be a redex, with $\bar{M}_1, \bar{N}_1 \in \boldsymbol{Val}$, since otherwise $M_1^{m_j} \in \mathcal{H}_{F,low}$ by Composition of High Expressions (Lemma B.2.14). There are two cases to consider:

- **$\bar{M}_1$ can compute.** Then we have $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$. We use Lemma B.2.10, Clause 11' and Subject Reduction (Theorem B.1.5) to conclude.
- **$\bar{M}_1$ is a value but $\bar{N}_1$ can compute.** Then by Remark B.2.8, $\bar{M}_2 \in \boldsymbol{Val}$. Then we have $\langle W, \{\bar{N}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F']{N^{n_k}} \langle W, \{\bar{N}_1'^{m_j}\}, T_1', S_1'\rangle$. We conclude using induction hypothesis, Clause 11' and Subject Reduction (Theorem B.1.5).

Clause 12'. Here $M_1 = (\mathrm{flow}\ F'\ \mathrm{in}\ \bar{M}_1)$ and $M_2 = (\mathrm{flow}\ F'\ \mathrm{in}\ \bar{M}_2)$ with $\bar{M}_1\ \mathcal{R}^j_{F\cup F',low}\ \bar{M}_2$. We can assume that $\bar{M}_1^{m_j} \notin \mathcal{H}_{F\cup F',low}$, since otherwise $\bar{M}_1^{m_j} \notin \mathcal{H}_{F,low}$ and by Composition of High Expressions (Lemma B.2.14) $M_1^{m_j} \in \mathcal{H}_{F,low}$. Therefore $\langle W, \{\bar{M}_1^{m_j}\}, T_1, S_1\rangle \xrightarrow[F'']{N^{n_k}} \langle W, \{\bar{M}_1'^{m_j}\}, T_1', S_1'\rangle$ with $F' = \bar{F} \cup F''$. By induction hypothesis, we have that $\langle W, \{\bar{M}_2^{m_j}\}, T_2, S_2\rangle \xrightarrow[F'']{N^{n_k}} \langle W, \{\bar{M}_2'^{m_j}\}, T_2', S_2'\rangle$, and that $M_1'\ \mathcal{R}^j_{F\cup\bar{F},low}\ M_2'$ and also $\langle T_1', S_1'\rangle =^{F\cup\bar{F},low} \langle T_2', S_2'\rangle$. Notice that $\langle T_1', S_1'\rangle =^{F,low} \langle T_2', S_2'\rangle$. We use Subject Reduction (Theorem B.1.5) and Clause 12' to conclude.

□

B.2.4 Behavior of Sets of Typable Threads

We can now prove the main-result regarding Non-disclosure for Networks:

Theorem B.2.22 (Soundness of Typing Non-disclosure for Networks.)**.** *Consider a pool of threads P. If for all $M^{m_j} \in P$ there exist Γ, s and τ such that $\Gamma \vdash^j_\emptyset M : s, \tau$, then P satisfies the Non-disclosure for Networks policy.*

Proof. To conclude the proof of the Soundness Theorem, it remains to exhibit an appropriate bisimulation on pools of threads.

The definition of $\mathcal{R}^\star_{low}$ is inductively defined as follows:

$$a)\ \frac{M^{m_j} \in \mathcal{H}_{\emptyset,low}}{\{M^{m_j}\}\ \mathcal{R}^\star_{low}\ \emptyset} \qquad b)\ \frac{M^{m_j} \in \mathcal{H}_{\emptyset,low}}{\emptyset\ \mathcal{R}^\star_{low}\ \{M^{m_j}\}}$$

$$c)\ \frac{M_1\ \mathcal{R}^j_{\emptyset,low}\ M_2}{\{{M_1}^{m_j}\}\ \mathcal{R}^\star_{low}\ \{{M_2}^{m_j}\}} \qquad d)\ \frac{P_1\ \mathcal{R}^\star_{low}\ P_2 \quad Q_1\ \mathcal{R}^\star_{low}\ Q_2}{P_1 \cup Q_1\ \mathcal{R}^\star_{low}\ P_2 \cup Q_2}$$

One can check that the relation $\mathcal{R}^{\star}_{low}$ is a *low*-bisimulation, using Lemma B.2.19, Strong Bisimulation for Typable Low Threads (Proposition B.2.21), Subject Reduction (Theorem B.1.5), and also Lemma B.1.2. The main result then follows easily.

□

B.3 Confinement for Networks

Proposition B.3.1 (Meaning of the declassification effect)**.**
Consider a thread M^{m_j} for which there exist Γ, G, s and τ such that $\Gamma \vdash^{j}_{G} M : s, \tau$. If we have $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$, then $F \subseteq s.d$.

Proof. We use induction on the inference of $\Gamma \vdash^{j}_{G} M : s, \sigma$, by case analysis on the last rule used in this typing proof (we show only the most interesting cases).

Flow. Here $M = (\text{flow } \bar{F} \text{ in } \bar{N})$, and we have $\Gamma \vdash^{j}_{G \cup \bar{F}} \bar{N} : \bar{s}', \tau$ with $s.d = \bar{s}.d \curlyvee \bar{s}'.d \curlyvee \bar{F}$. There are two possibilities:

- $M = \bar{F}$ **and** $\bar{N}$ **can compute.** We have $\langle W, \{\bar{N}\}, T, S\rangle \xrightarrow[\bar{F}']{N^{n_k}} \langle W, \{\bar{N}'\}, T', S'\rangle$ with $F = \bar{F}' \cup \bar{F}$. By induction hypothesis, $\bar{F}' \subseteq \bar{s}'.d$. Since $\bar{s}'.d \subseteq s.d$, then $\bar{F} \subseteq s.d$, and since $\bar{F}' \subseteq s.d$, then $F \subseteq s.d$.
- $M = \bar{F}$ **and** $\bar{N} = V$**.** Then we have $\langle W, \{(\text{flow } \bar{F} \text{ in } \bar{N})\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{V\}, T, S\rangle$ with $F = \emptyset$, so $F \subseteq s.d$ holds vacuously.

Allow. Here $M = (\text{allowed } \bar{F} \text{ then } N_t \text{ else } N_f)$ and we have $\Gamma \vdash^{j}_{G} N_t : \bar{s}_t, \tau$ and $\Gamma \vdash^{j}_{G} N_f : \bar{s}_f, \tau$ with $s.d = \bar{s}.d \curlyvee (s_t.d - \bar{F}) \curlyvee s_f.d$. There are two possibilities:

- $\bar{M} = \bar{F} \subseteq W(T(m_j))^*$**.** Then we have $\langle W, M, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, N_t, T, S\rangle$ with $F = \emptyset$, so $F \subseteq s.d^*$ holds vacuously.
- $\bar{M} = \bar{F} \not\subseteq A^*$**.** Then we have $\langle W, M, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, N_f, T, S\rangle$ with $F = \emptyset$, so $F \subseteq s.d^*$ holds vacuously.

□

Theorem B.3.2 (Soundness of Typing Confinement for Networks)**.** *Consider a well formed network $\langle W, P, T\rangle$ such that for all $M^{m_j} \in P$ there exist Γ, s and τ such that $\Gamma \vdash^{j}_{G} M : s, \tau$ and $s.d \subseteq W(T(m_j))^*$. Then the set of triples $\mathrm{trp}(\langle W, P, T\rangle)$ is a set of operationally confined threads.*

Proof. We show that the set

$$C = \{\langle A, d, M^{m_j}\rangle \mid \exists \Gamma, G, s, \tau \text{ such that } \Gamma \vdash^{j}_{G} M : s, \tau \text{ and } s.d \subseteq A^*\}$$

is a set of operationally confined threads.

Consider a triple $\langle A, d, M^{m_j}\rangle \in C$, for which we have a transition $\langle W, \{M^{m_j}\}, T, S\rangle \xrightarrow[F]{N^{n_k}} \langle W, \{M'^{m_j}\}, T', S'\rangle$. By Proposition B.3.1 we have that $F \subseteq s.d \subseteq W(T(m_j))$. By Subject Reduction (Proposition B.1.5) we have there exists s' such that $\Gamma \vdash^{j}_{G} M' : s', \tau$ and $s.d \cup W(T(m_j)) \preceq s'.d$, i.e. $s'.d \subseteq (s.d \cup (W(T(m_j)))^*$, and that there exists s'' such that $\Gamma \vdash^{j}_{G} N : s'', \mathsf{unit}$ and $s.d \cup W(T(n_k)) \preceq s''.d$. Note that if $N \neq ()$, then $T(m_j) = T'(n_k)$, so $s''.d \subseteq W(T(m_j))^*$. It remains to prove that $s'.d \subseteq W(T'(m_j))^*$. The result is trivial when $T'(m_j) = T(m_j)$, so we will only check the case where $M = \mathrm{E}[(\text{goto } d')]$. We then have that $T'(m_j) = d'$ and $s.d \subseteq W(d')$. By MIG we have $s.d = s'.d$, so we conclude that $s'.d \subseteq W(T'(m_j))$. □

References

A. Almeida Matos. Non-disclosure for distributed mobile code. In Ramaswamy Ramanujam and Sandeep Sen, editors, *FSTTCS'05: 25th International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 3821 of *Lecture Notes in Computer Science*, pages 177–188. Springer, 2005.

A. Almeida Matos. *Typing Secure Information Flow: Declassification and Mobility*. PhD thesis, École Nationale Supérieure des Mines de Paris, 2006.

A. Almeida Matos. Flow-policy awareness for distributed mobile code (proofs). Technical report, Instituto Superior Técnico de Lisboa, 2008. URL `http://web.ist.utl.pt/ ana.matos/Research.html/08-A-flow+aware+distr-SUBM.pdf`.

A. Almeida Matos and G. Boudol. On declassification and the non-disclosure policy. In *CSFW'05: 18th IEEE Computer Security Foundations Workshop*, pages 226–240. IEEE Computer Society, 2005.

G. Barthe, T. Rezk, and D. Naumann. Deriving an information flow checker and certifying compiler for java. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 230–242, Washington, DC, USA, 2006. IEEE Computer Society.

G. Barthe, S. Cavadini, and T. Rezk. Tractable enforcement of declassification policies. In *CSF*, pages 83–97. IEEE Computer Society, 2008.

G. Boudol. A generic membrane model. In Corrado Priami and Paola Quaglia, editors, *GC'04: IST/FET International Workshop on Global Computing*, volume 3267 of *Lecture Notes in Computer Science*, pages 208–222. Springer, 2005.

G. Boudol and I. Castellani. Noninterference for concurrent programs and thread systems. *Theoretical Computer Science*, 281(1–2):109–130, 2002.

G. Boudol and M. Kolundzija. Access Control and Declassification. In *Computer Network Security*, volume 1 of *CCIS*, pages 85–98. Springer-Verlag, 2007.

N. Broberg and D. Sands. Flow locks: Towards a core calculus for dynamic flow policies. In *Programming Languages and Systems. 15th European Symposium on Programming, ESOP 2006*, volume 3924 of *Lecture Notes in Computer Science*. Springer Verlag, 2006.

S. Chong and A. C. Myers. Security policies for downgrading. In *Proceedings of the 11th ACM conference on Computer and communications security*. ACM Press, 2004.

S. Crafa, M. Bugliesia, and G. Castagna. Information flow security for boxed ambients. In Vladimiro Sassone, editor, *F-WAN'02: Workshop on Foundations of Wide Area Network Computing*, volume 66 of *Electronic Notes in Theoretical Computer Science*, pages 76–97. Elsevier, 2002.

D. E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5): 236–243, 1976.

J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Symposium on Security and Privacy*, pages 11–20. IEEE Computer Society, 1982.

D. Gorla, M. Hennessy, and V. Sassone. Security policies as membranes in systems for global computing. In *Foundations of Global Ubiquitous Computing, FGUC 2004*, ENTCS, pages 23–42. Elsevier, 2005.

P. Li and S. Zdancewic. Downgrading policies and relaxed noninterference. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM Press, 2005.

J. M. Lucassen and D. K. Gifford. Polymorphic effect systems. In *POPL'88: 15th ACM symposium on Principles of programming languages*, pages 47–57. ACM Press, 1988.

H. Mantel and A. Sabelfeld. A unifying approach to the security of distributed and multi-threaded programs. *Journal of Computer Security*, 11(4):615–676, 2003.

F. Martins and V.T. Vasconcelos. History-based access control for distributed processes. In *Proceedings of TGC'05*, LNCS. Springer-Verlag, 2005.

A. C. Myers and B. Liskov. Complete, safe information flow with decentralized labels. In *19th IEEE Computer Society Symposium on Security and Privacy*, pages 186–197. IEEE Computer Society, 1998.

A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification and qualified robustness. *J. Comput. Secur.*, 14(2):157–196, 2006.

G. C. Necula. Proof-carrying code. In *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 106–119, New York, NY, USA, 1997. ACM.

A. Sabelfeld and A. Myers. A model for delimited information release. In *International Symposium on Software Security (ISSS'03)*, volume 3233 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

A. Sabelfeld and A. C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1):5–19, 2003.

A. Sabelfeld and D. Sands. Dimensions and principles of declassification. In *CSFW'05: 18th IEEE Computer Security Foundations Workshop*, pages 255–269. IEEE Computer Society, 2005.

G. Smith. A new type system for secure information flow. In *CSFW'01: 14th IEEE Computer Security Foundations Workshop*, pages 115–125. IEEE Computer Society, 2001.

Andrew K. Wright and Matthias Felleisen. A syntactic approach to type soundness. *Informnation and Computation*, 115(1):38–94, 1994.

S. Zdancewic. Challenges for information-flow security. In *1st International Workshop on the Programming Language Interference and Dependence (PLID'04)*, 2004.

S. Zdancewic, L. Zheng, N. Nystrom, and A. C. Myers. Secure program partitioning. *ACM Transactions on Computer Systems*, 20(3):283–328, 2002.