Proving Noninterference by a Fully Complete Translation to the Simply Typed λ-calculus

Naokata Shikuma and Atsushi Igarashi

Graduate School of Informatics, Kyoto University
{naokata,igarashi}@kuis.kyoto-u.ac.jp

Abstract. Tse and Zdancewic have formalized the notion of noninterference for Abadi et al.'s DCC in terms of logical relations and given a proof by reduction to parametricity of System F. Unfortunately, their proof contains errors in a key lemma that their translation from DCC to System F preserves the logical relations defined for both calculi. We prove noninterference for a variant of DCC by reduction to the basic lemma of a logical relation for the simply typed λ-calculus, using a *fully complete* translation to the simply typed λ-calculus. Full completeness plays an important role in showing preservation of the two logical relations through the translation.

1 Introduction

Background. Dependency analysis is a family of static program analyses to trace dependencies between inputs and outputs of a given program. For example, information flow analysis [3], binding-time analysis [8], and call tracking [16] are its instances. One of the most important correctness criteria of the dependency analysis is called *noninterference* [5], which roughly means that, for any pair of program inputs that are equivalent from the viewpoint of an observer at some security level, the outputs are also equivalent for the observer. Various techniques for type-based dependency analyses have been proposed, especially, in the context of language-based security [15].

Abadi et al. proposed a unifying framework called *dependency core calculus* (DCC) [1] for type-based dependency analyses for higher-order functional languages, and showed noninterference for several type systems of concrete dependency analyses by embedding them into DCC.

Recently, Tse and Zdancewic formalized this property for DCC by using a syntactic *logical relation* [9]—a family of type-indexed relations, defined by induction on types, over programs—as the equivalence relation for inputs and outputs, thereby generalizing the notion of noninterference to higher-order programs. They also gave a proof of noninterference by reducing it to the parametricity theorem [14, 18], which was also formalized in terms of syntactic logical relations, of System F [13, 4]. Their technical development is summarized as follows:

1. Define a translation from DCC to System F;

2. Prove, by induction on the structure of types, that the translation is both sound and complete—that is, it preserves the logical relations in the sense that
$$e_1 \approx_D e_2 : t \iff f(e_1) \approx_F f(e_2) : f(t)$$
where t is a DCC type, f is the translation from DCC to System F and $\approx_D$ and $\approx_F$ represent the logical relations for DCC and System F, respectively; and
3. Prove noninterference by reduction to the parametricity theorem of System F, using the sound and complete translation above.

Unfortunately, in the second step, their proof contains an error, which we will briefly explain here. First note that, for function types $t_1 \to t_2$, the logical relations are defined by: $e_1 \approx_x e_2 : t_1 \to t_2$ if and only if, for any $e_1' \approx_x e_2' : t_1$, $e_1\, e_1' \approx_x e_2\, e_2' : t_2$ (x stands for either D or F) and that the type translation is homomorphic for function types, namely $f(t_1 \to t_2) = f(t_1) \to f(t_2)$. Then, consider the case where t is a function type $t_1 \to t_2$. To show the left-to-right direction, we must show that $f(e_1)\, M_1 \approx_F f(e_2)\, M_2 : f(t_2)$ for any $M_1 \approx_F M_2 : f(t_1)$, from the assumption $e_1 \approx_D e_2 : t_1 \to t_2$, but we get stuck because there is no applicable induction hypothesis. If the translation were *full* [6] (or surjective), M_1 and M_2 would be of the forms $f(e_1')$ and $f(e_2')$, making it possible to apply an induction hypothesis, and the whole proof would go through. Their translation, however, turns out *not* to be full; we have actually found a counterexample for the preservation of the equivalence from the failure of the fullness (see Section 5 for more details). So, although interesting, this indirect proof method fails at least for the combination of DCC and System F.

Our Contributions. In this paper, we prove noninterference by Tse and Zdancewic's method in a slightly different setting: In order to obtain a fully complete translation, we change the source language to what we call Sealing Calculus ($\lambda^{[]}$), which is a subset of extended DCC of Tse and Zdancewic [17], and use a simpler target language, namely the simply typed λ-calculus $\lambda^{\to}$. Then, the basic lemma for a logical relation of $\lambda^{\to}$ is used in place of the parametricity theorem. Our technical contributions can be summarized as follows:

- Development of a sound and fully complete translation from $\lambda^{[]}$ to $\lambda^{\to}$; and
- A proof of the noninterference theorem of $\lambda^{[]}$ by reducing to the basic lemma of $\lambda^{\to}$ with the translation.

The existence of a fully complete translation means that $\lambda^{[]}$ provides syntax that is rich enough to express every denotation in the model (that is, $\lambda^{\to}$). Thus, this result suggests that the extension of DCC proposed by Tse and Zdancewic is really an improvement over DCC, for which the previously proposed translation is not full.

Structure of the Paper. The rest of the paper is organized as follows. Section 2 introduces $\lambda^{[]}$ with its syntax, type system, reduction, and logical relations and

then the statement of the noninterference theorem. Section 3 introduces $\lambda^{\to}$ and defines a translation from $\lambda^{[]}$ to $\lambda^{\to}$ and its inverse. In Section 4, we complete our proof of noninterference by reducing it to the basic lemma of logical relations for $\lambda^{\to}$. Finally, Section 5 discusses related work and Section 6 gives concluding remarks.

2 Sealing Calculus

In this section, we define $\lambda^{[]}$, which is a simply typed λ-calculus extended with the notion of sealing. We write $[e]_a$ for sealing e by authority a; the sealed value can be extracted by unsealing e^a, whose result must not be leaked to anyone without the authority a. Authorities represent rights to access confidential data; so the power set of authorities naturally forms security levels, which are ordered by inclusion. To keep track of dependency by a type system, information on the authority used for sealing is attached to types of sealing $[t]_a$; furthermore, type judgments, written $\Gamma ; \ell \vdash e : t$, are augmented by a level $\ell = \{a_1, \ldots, a_n\}$, which is also called a protection context elsewhere [17]. This judgment means that the value of e has type t as usual and, moreover, cannot be leaked to levels that lack some of the authorities in ℓ.

2.1 Syntax

Let $\mathcal{A}$ be the countable set of *authorities*, and ranged over by a (possibly with subscripts). The metavariable ℓ ranges over *levels*, which are finite subsets of authorities. The metavariables x, y, and z (possibly with subscripts) range over the denumerable set of *variables*. Then, the types and terms of $\lambda^{[]}$ are defined as follows.

Definition 1 (Types). *The set of* types*, ranged over by* t, t', t_1, t_2, $\ldots$, *is defined as follows:*

$$t ::= \mathit{unit} \mid t \to t \mid t \times t \mid t + t \mid [t]_a$$

We call $[t]_a$ *a* sealed value type*.*

Definition 2 (Terms). *The set of* terms*, ranged over by* e, e', e_1, e_2, $\ldots$, *is defined as follows:*

$$\begin{aligned} e ::= {} & x \mid () \mid \lambda x{:}t.\, e \mid e\, e \mid \langle e, e \rangle \mid \pi_1(e) \mid \pi_2(e) \mid \iota_1(e) \mid \iota_2(e) \\ & \mid (\mathbf{case}\, e\, \mathbf{of}\, \iota_1(x_1).e \mid \iota_2(x_2).e) \mid [e]_a \mid e^a \end{aligned}$$

Terms of $\lambda^{[]}$ include the unit value, pairing, projection, injection, and case analysis as well as λ-abstraction and applications. As usual, x is bound in $\lambda x{:}t.\, e$ and x_1 and x_2 are bound in e_1 and e_2 of $(\mathbf{case}\, e_0\, \mathbf{of}\, \iota_1(x_1).e_1 \mid \iota_2(x_2).e_2)$, respectively. We say, for $[e]_a$, e is *sealed at* a, and call $[e]_a$ and e^a (a-)*sealing term* and (a-)*unsealing term*, respectively. In this paper, α-conversions are defined in a customary manner and implicit α-conversions are assumed to make all the bound variables distinct from other (bound and free) variables.

2.2 Type System

As mentioned above, the form of type judgement of $\lambda^{[]}$ is $\Gamma\,;\,\ell \vdash e : t$, where Γ is a (finite) mapping from variables to types. This judgement is read as "e is given type t at level ℓ under context Γ." Since the computation of e depends on authorities in ℓ, any information on its value should not be leaked to any other level ℓ', which is not a superset of ℓ.

The typing rules of $\lambda^{[]}$ are given as follows:

$$\frac{x : t \in \Gamma}{\Gamma\,;\,\ell \vdash x : t} \qquad \Gamma\,;\,\ell \vdash () : \mathit{unit} \qquad \frac{\Gamma,\, x : t_1\,;\,\ell \vdash e : t_2}{\Gamma\,;\,\ell \vdash \lambda x{:}t.\, e : t_1 \to t_2}$$

$$\frac{\Gamma\,;\,\ell \vdash e : t_1 \to t_2 \qquad \Gamma\,;\,\ell \vdash e' : t_1}{\Gamma\,;\,\ell \vdash e\, e' : t_2} \qquad \frac{\Gamma\,;\,\ell \vdash e_1 : t_1 \qquad \Gamma\,;\,\ell \vdash e_2 : t_2}{\Gamma\,;\,\ell \vdash \langle e_1,\, e_2\rangle : t_1 \times t_2}$$

$$\frac{\Gamma\,;\,\ell \vdash e : t_1 \times t_2 \qquad i \in \{1,\, 2\}}{\Gamma\,;\,\ell \vdash \pi_i(e) : t_i} \qquad \frac{\Gamma\,;\,\ell \vdash e : t \qquad i \in \{1,\, 2\}}{\Gamma\,;\,\ell \vdash \iota_i(e) : t_1 + t_2}$$

$$\frac{\Gamma\,;\,\ell \vdash e : t_1 + t_2 \qquad \Gamma,\, x_1 : t_1\,;\,\ell \vdash e_1 : t \qquad \Gamma,\, x_2 : t_2\,;\,\ell \vdash e_2 : t}{\Gamma\,;\,\ell \vdash (\mathbf{case}\, e\, \mathbf{of}\, \iota_1(x_1).e_1 \,|\, \iota_2(x_2).e_2) : t}$$

$$\frac{\Gamma\,;\,\ell \cup \{a\} \vdash e : t}{\Gamma\,;\,\ell \vdash [e]_a : [t]_a} \qquad \frac{\Gamma\,;\,\ell \vdash e : [t]_a \qquad a \in \ell}{\Gamma\,;\,\ell \vdash e^a : t}$$

All the rules but the last two are standard. The (second last) rule for sealing means that, by sealing with a, it is legal to leak $[e]_a$ to a level without a: at such a level, e cannot be unsealed, as is shown in the (last) rule for unsealing.

2.3 Reduction

The reduction relation for $\lambda^{[]}$ is written $e \longrightarrow e'$ and given as the least compatible relation closed by the following rules:

$$(\lambda x{:}t.\, e_1)\, e_2 \longrightarrow [e_2/x]e_1 \qquad \pi_i(\langle e_1,\, e_2\rangle) \longrightarrow e_i$$
$$(\mathbf{case}\, \iota_i(e)\, \mathbf{of}\, \iota_1(x_1).e_1 \,|\, \iota_2(x_2).e_2) \longrightarrow [e/x_i]e_i \qquad ([e]_a)^a \longrightarrow e$$

We write $[e/x]$ for a capture-avoiding substitution of e for the free occurrences of variable x. All rules are straightforward. The last rule says that the term sealed by a is opened by the same authority. In what follows, we use v for normal forms.

2.4 Basic Properties

We list the basic properties of $\lambda^{[]}$. The first lemma below means that, if e is permitted at some level, then it is permitted also at a higher level.

Lemma 1 (Level Weakening). *If $\Gamma\,;\,\ell \vdash e : t$, then $\Gamma\,;\,\ell \cup \{a\} \vdash e : t$, and the derivations of these judgements have the same size.*

The following three theorems are standard.

Theorem 1 (Subject Reduction). *If $\Gamma\,;\,\ell \vdash e : t$ and $e \longrightarrow e'$, then $\Gamma\,;\,\ell \vdash e' : t$.*

Theorem 2 (Strong Normalization). *If* $\Gamma ; \ell \vdash e : t$*, then* e *is strongly normalizing.*

Theorem 3 (Church-Rosser Property). *If* $\Gamma ; \ell \vdash e : t$ *and* $e \longrightarrow^* e_1$ *and* $e \longrightarrow^* e_2$*, then there exists a term* e' *such that* $e_i \longrightarrow^* e'$ $(i = 1, 2)$*.*

2.5 Logical Relation and Noninterference

Now we define logical relations to express equivalence of terms from the viewpoint of an observer at some level, and then state the noninterference theorem. To take level information into account, the logical relations (for close terms and normal forms) are indexed by levels as well as types. $e_1 \approx_\ell e_2 : t$ means that closed terms e_1 and e_2 of type t are logically related at level ℓ. Similarly, $v_1 \sim_\ell v_2 : t$ means that closed normal forms v_1 and v_2 of t, are logically related at ℓ. We assume $\cdot ; \ell \vdash e_i : t$ and $\cdot ; \ell \vdash v_i : t$ for $i = 1, 2$ (we write $\cdot$ for the empty context).

Definition 3 (Logical Relations for $\lambda^{[]}$). *The relations* $v_1 \sim_\ell v_2 : t$ *and* $e_1 \approx_\ell e_2 : t$ *are defined by the following rules:*

$$() \sim_\ell () : \mathit{unit} \qquad \frac{\forall (e_1 \approx_\ell e_2 : t_1).\, v_1\, e_1 \approx_\ell v_2\, e_2 : t_2}{v_1 \sim_\ell v_2 : t_1 \to t_2} \qquad \frac{v_{11} \sim_\ell v_{21} : t_1 \qquad v_{12} \sim_\ell v_{22} : t_2}{\langle v_{11},\, v_{12} \rangle \sim_\ell \langle v_{21},\, v_{22} \rangle : t_1 \times t_2}$$

$$\frac{v_1 \sim_\ell v_2 : t_i \qquad i \in \{1, 2\}}{\iota_i(v_1) \sim_\ell \iota_i(v_2) : t_1 + t_2} \qquad \frac{v_1 \sim_\ell v_2 : t \qquad a \in \ell}{[v_1]_a \sim_\ell [v_2]_a : [t]_a} \qquad \frac{a \notin \ell}{[v_1]_a \sim_\ell [v_2]_a : [t]_a}$$

$$\frac{e_1 \longrightarrow^* v_1 \qquad e_2 \longrightarrow^* v_2 \qquad v_1 \sim_\ell v_2 : t}{e_1 \approx_\ell e_2 : t}$$

Most rules are standard. There are two rules for $[v_1]_a \sim_\ell [v_2]_a : [t]_a$. When $a \in \ell$, an observer at ℓ can examine v_i by unsealing $[v_i]_a$ $(i = 1, 2)$, so only when its contents are equivalent, these sealing terms are equivalent. Otherwise, the observer cannot distinguish them at all and those terms are always regarded equivalent.

We use γ to represent a simultaneous substitution of terms for variables and write $\gamma_1 \approx_\ell \gamma_2 : \Gamma$ if $\mathit{dom}(\gamma_1) = \mathit{dom}(\gamma_2) = \mathit{dom}(\Gamma)$ and $\gamma_1(x) \approx_\ell \gamma_2(x) : \Gamma(x)$ for any $x \in \mathit{dom}(\gamma_1)$. Then, the noninterference theorem is stated as follows:

Theorem 4 (Noninterference). *If* $\Gamma ; \ell \vdash e : t$ *and* $\gamma_1 \approx_\ell \gamma_2 : \Gamma$*, then* $\gamma_1(e) \approx_\ell \gamma_2(e) : t$*.*

As mentioned in the introduction, noninterference means that, for any pair of program inputs that are equivalent from the viewpoint of an observer at some security level, the outputs are also equivalent for the observer. Here, substitutions γ_1 and γ_2 play roles of equivalent inputs to program e. So, this property guarantees the correctness of the type system as a dependency analysis.

Although we could give a direct proof of this theorem by induction on the derivation of $\Gamma ; \ell \vdash e : t$ rather easily, we show an indirect proof that reduces the property to a corresponding property in $\lambda^{\to}$, namely the basic lemma of logical relations.

3 Translation

In this section, we define a formal translation from $\lambda^{[]}$ to the simply typed λ-calculus $\lambda^{\to}$ and its inverse. Both translations are shown to preserve typing. We start with reviewing $\lambda^{\to}$ briefly with logical relations for it.

3.1 $\lambda^{\to}$

$\lambda^{\to}$ introduced here is a standard one with unit, base, function, product, and sum types. We assume that base types, written α_a ($a \in \mathcal{A}$), have one-to-one correspondence with authorities. We use metavariables M for terms and A for types. The syntax of $\lambda^{\to}$ is given as follows:

$$\begin{aligned} A &::= \alpha \mid \mathit{unit} \mid A \to A \mid A \times A \mid A + A \\ M &::= x \mid () \mid \lambda x{:}A.\, M \mid M\, M \mid \langle M,\, M\rangle \mid \pi_i(M) \mid \iota_i(M) \\ &\quad \mid (\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M \mid \iota_2(x_2).M) \end{aligned}$$

The form of type judgement of $\lambda^{\to}$ is $\Delta \vdash M : A$, where Δ is a (finite) mapping from variables to $\lambda^{\to}$ types. For brevity, we omit typing rules, which are completely standard. The reduction of $\lambda^{\to}$ terms consists of standard β-reduction and the following commutative conversion.

$$\frac{(x_1, x_2 \notin FV(M'))}{(\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1 \mid \iota_2(x_2).M_2)\, M' \longrightarrow \mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1\, M' \mid \iota_2(x_2).M_2\, M'}$$

$$\frac{(i \in \{1,2\})}{\pi_i(\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1 \mid \iota_2(x_2).M_2) \longrightarrow \mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).\pi_i(M_1) \mid \iota_2(x_2).\pi_i(M_2)}$$

$$\begin{array}{c} (x_1, x_2 \notin FV(M_1') \cup FV(M_2')) \\ \hline \begin{array}{l} \mathbf{case}\, (\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1 \mid \iota_2(x_2).M_2)\, \mathbf{of}\, \iota_1(y_1).M_1' \mid \iota_2(y_2).M_2' \\ \quad \longrightarrow \mathbf{case}\, M\ \mathbf{of}\ \iota_1(x_1).(\mathbf{case}\, M_1\, \mathbf{of}\, \iota_1(y_1).M_1' \mid \iota_2(y_2).M_2') \\ \qquad\qquad \mid\ \iota_2(x_2).(\mathbf{case}\, M_2\, \mathbf{of}\, \iota_1(y_1).M_1' \mid \iota_2(y_2).M_2') \end{array} \end{array}$$

Here, we write $FV(M)$ for the set of free variables in M. In what follows, we use V for normal forms.

The resulting calculus (with commutative conversion) satisfies the standard properties of subject reduction, Church-Rosser, and strong normalization [2]. We say (the type derivation $\Delta \vdash M : A$ of) a term satisfies the subformula property when any type in the derivation is a subexpression of either A or a type occurring in Δ. Then, any well typed term can reduce to the one that satisfies the subformula property as in the theorem below, which makes it easy to ensure the fullness of the translation.

Theorem 5 (Subformula Property). *If $\Delta \vdash M : A$, then there exists a normal form V such that $M \longrightarrow^* V$ and $\Delta \vdash V : A$, which satisfies the subformula property.*

Remark 1. Commutative conversion is necessary for the above theorem to hold. Without commutative conversion,

$$\lambda x\colon \mathit{unit} + \mathit{unit}.\,((\mathbf{case}\, x\, \mathbf{of}\, \iota_1(x_1).\lambda y\colon \mathit{unit}.\,() \mid \iota_2(x_2).\lambda y\colon \mathit{unit}.\,()))\,()$$

of type $\mathit{unit} + \mathit{unit} \to \mathit{unit}$ would be a norml form, which does not satisfy the subformula property, because a subterm $\lambda y\colon \mathit{unit}.\,()$ has type $\mathit{unit} \to \mathit{unit}$, which does not occur in $\mathit{unit} + \mathit{unit} \to \mathit{unit}$. This theorem also requires full reduction, which allows any redex (even under λ) to reduce.

3.2 Logical Relation for $\lambda^{\to}$

We define syntactic logical relations for $\lambda^{\to}$ in the standard manner. As for $\lambda^{[]}$, there are relations for terms and normal forms, written $\Delta \vdash M_1 \approx M_2 : A$ (read "terms M_1 and M_2 of type A are logically related under context Δ") and $\Delta \vdash V_1 \sim V_2 : A$ (read similarly), respectively. We assume that $\Delta \vdash M_i : A$ and $\Delta \vdash V_i : A$ for $i = 1, 2$.

Definition 4 (Logical Relations for $\lambda^{\to}$). *The relations $\Delta \vdash M_1 \approx M_2 : A$ and $\Delta \vdash V_1 \sim V_2 : A$ are the least relation closed under the following rules:*

$$\Delta \vdash () \sim () : \mathit{unit} \qquad \Delta \vdash V_1 \sim V_2 : \alpha_a \qquad \frac{\Delta \vdash V_{11} \sim V_{21} : A_1 \qquad \Delta \vdash V_{12} \sim V_{22} : A_2}{\Delta \vdash \langle V_{11}, V_{12}\rangle \sim \langle V_{21}, V_{22}\rangle : A_1 \times A_2}$$

$$\frac{\Delta \vdash V_1 \sim V_2 : A_i \qquad i \in \{1,2\}}{\Delta \vdash \iota_i(V_1) \sim \iota_i(V_2) : A_1 + A_2} \qquad \frac{\forall(\Delta \vdash M_1 \approx M_2 : A_1).\, \Delta \vdash V_1\, M_1 \approx V_2\, M_2 : A_2}{\Delta \vdash V_1 \sim V_2 : A_1 \to A_2}$$

$$\frac{M_1 \longrightarrow^* V_1 \qquad M_2 \longrightarrow^* V_2 \qquad \Delta \vdash V_1 \sim V_2 : A}{\Delta \vdash M_1 \approx M_2 : A}$$

We write δ for a simultaneous substitution of ($\lambda^{\to}$) terms for variables and $\Delta' \vdash \delta_1 \approx \delta_2 : \Delta$ if $\mathit{dom}(\delta_1) = \mathit{dom}(\delta_2) = \mathit{dom}(\Delta)$ and for any $x \in \mathit{dom}(\delta_1)$, $\Delta' \vdash \delta_1(x) \approx \delta_2(x) : \Delta(x)$. Then, the basic lemma is as follows:

Lemma 2 (Basic Lemma). *If $\Delta \vdash M : A$ and $\Delta' \vdash \delta_1 \approx \delta_2 : \Delta$, then $\Delta' \vdash \delta_1(M) \approx \delta_2(M) : A$.*

Proof. By induction on the derivation of $\Delta \vdash M : A$.

Remark 2. Although the above logical relation for $\lambda^{\to}$ are not reflexive in general (for example $x : A + A \not\vdash x \approx x : A + A$), we have $\Delta \vdash M \approx M : A$ if all the types in Δ are base types α_a. This is derived from Lemma 2 and the fact that $\Delta \vdash x \approx x : \Delta(x)$ if $\Delta(x) = \alpha_a$, by definition.

3.3 From $\lambda^{[]}$ To $\lambda^{\to}$

We define the translation of $\lambda^{[]}$ to $\lambda^{\to}$. One of the main ideas of the translation is to translate sealing of type $[t]_a$ to a function from the base type α_a, which corresponds to a. The sealed value can be extracted by passing a term of α_a as an argument. Intuitively, the term of α_a serves as a "key" to unseal.

Definition 5 (Translation of Types and Contexts). $(\cdot)^\dagger$ *is a function from* $\lambda^{[]}$ *types to* $\lambda^{\rightarrow}$ *type, defined by:*

$$\mathit{unit}^\dagger = \mathit{unit} \qquad (t_1 \,\mathbf{op}\, t_2)^\dagger = t_1^\dagger \,\mathbf{op}\, t_2^\dagger \qquad ([t]_a)^\dagger = \alpha_a \to t^\dagger$$

where **op** *stands for* $\to, \times,$ *or* $+$. $(\cdot)^\dagger$ *is extended pointwise to contexts by:* $\Gamma^\dagger = \{x : t^\dagger \mid x : t \in \Gamma\}$.

The translation to $\lambda^{\rightarrow}$ is represented by $\Gamma; \sigma \vdash e : t \searrow M$, read "$\lambda^{[]}$ term e of type t is translated to M under Γ and σ," where σ is an injective finite map from authorities to variables. We assume that the range of σ and the domain of Γ are disjoint.

Definition 6 (Translation of Terms). *The relation* $\Gamma; \sigma \vdash e : t \searrow M$ *is defined as the least relation closed under the following rules:*

$$\Gamma; \sigma \vdash x : t \searrow x \qquad \Gamma; \sigma \vdash () : \mathit{unit} \searrow () \qquad \frac{\Gamma, x : t_1; \sigma \vdash e : t_2 \searrow M}{\Gamma; \sigma \vdash \lambda x{:}t_1.\, e : t_1 \to t_2 \searrow \lambda x{:}t_1^\dagger.\, M}$$

$$\frac{\Gamma; \sigma \vdash e : t_1 \to t_2 \searrow M \qquad \Gamma; \sigma \vdash e' : t_1 \searrow M'}{\Gamma; \sigma \vdash e\, e' : t_2 \searrow M\, M'} \qquad \frac{\Gamma; \sigma \vdash e_1 : t_1 \searrow M_1 \qquad \Gamma; \sigma \vdash e_2 : t_2 \searrow M_2}{\Gamma; \sigma \vdash \langle e_1,\, e_2\rangle : t_1 \times t_2 \searrow \langle M_1,\, M_2\rangle}$$

$$\frac{\Gamma; \sigma \vdash e : t_1 \times t_2 \searrow M \qquad i \in \{1,\, 2\}}{\Gamma; \sigma \vdash \pi_i(e) : t_i \searrow \pi_i(M)} \qquad \frac{\Gamma; \sigma \vdash e : t_i \searrow M \qquad i \in \{1,\, 2\}}{\Gamma; \sigma \vdash \iota_i(e) : t_1 + t_2 \searrow \iota_i(M)}$$

$$\frac{\Gamma; \sigma \vdash e : t_1 + t_2 \searrow M \qquad \Gamma, x_1 : t_1; \sigma \vdash e_1 : t \searrow M_1 \qquad \Gamma, x_2 : t_2; \sigma \vdash e_2 : t \searrow M_2}{\Gamma; \sigma \vdash (\mathbf{case}\, e\, \mathbf{of}\, \iota_1(x_1).e_1 \mid \iota_2(x_2).e_2) : t \searrow (\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1 \mid \iota_2(x_2).M_2)}$$

$$\frac{\Gamma; \sigma\{a \mapsto k\} \vdash e : t \searrow M \qquad k \text{ fresh or } k = \sigma(a)}{\Gamma; \sigma \vdash [e]_a : [t]_a \searrow \lambda k{:}\alpha_a.\, M} \qquad \frac{\Gamma; \sigma \vdash e : [t]_a \searrow M}{\Gamma; \sigma \vdash e^a : t \searrow M\, \sigma(a)}$$

Here, we write $\sigma\{a \mapsto k\}$ *for a mapping from* $\mathit{dom}(\sigma) \cup \{a\}$ *to variables defined by:* $\sigma\{a \mapsto k\}(a) = k$; *and* $\sigma\{a \mapsto k\}(a') = \sigma(a')$ *if* $a \neq a'$.

The translation of terms is easily derived from the translation rules for types. Here, the mapping σ, whose domain represents the level at which the SDC term is typed, records correspondence between authorities and variables that are used as keys. In the last rule, a key to open the sealing is retrieved from σ—if e is well typed at the level represented by $\mathit{dom}(\sigma)$, then a should be in the domain of σ. Then, well typed $\lambda^{[]}$ terms can be translated to well typed $\lambda^{\rightarrow}$ terms as in the theorem below. Here, we write $\sigma^\dagger$ for a context defined by: $\{\sigma(a) : \alpha_a \mid a \in \mathit{dom}(\sigma)\}$.

Theorem 6 (Translation Preserves Typing). *If* $\Gamma\,;\, \ell \vdash e : t$ *and* $\mathit{dom}(\sigma) = \ell$, *then there exists a* $\lambda^{\rightarrow}$ *term* M *such that* $\Gamma; \sigma \vdash e : t \searrow M$, *and that* $\Gamma^\dagger, \sigma^\dagger \vdash M : t^\dagger$.

Proof. By induction on the derivation of $\Gamma\,;\, \ell \vdash e : t$.

3.4 From $\lambda^{\to}$ To $\lambda^{[]}$

We define the inverse translation, represented by $\Gamma; \sigma \vdash M \nearrow e : t$. It is read "$\lambda^{\to}$ term M of type $t^\dagger$ under $\Gamma^\dagger$ and $\sigma^\dagger$ is translated back to a $\lambda^{[]}$ term e."

Definition 7 (Inverse Translation). *The relation $\Gamma; \sigma \vdash M \nearrow e : t$ is defined as the least relation closed by the following rules:*

$$\Gamma; \sigma \vdash x \nearrow x : t \qquad \Gamma; \sigma \vdash () \nearrow () : \mathit{unit} \qquad \frac{\Gamma, x : t_1; \sigma \vdash M \nearrow e : t_2}{\Gamma; \sigma \vdash \lambda x{:}t_1^\dagger.\, M \nearrow \lambda x{:}t_1.\, e : t_1 \to t_2}$$

$$\frac{\Gamma; \sigma \vdash M \nearrow e : t_1 \to t_2 \quad \Gamma; \sigma \vdash M' \nearrow e' : t_1}{\Gamma; \sigma \vdash M\, M' \nearrow e\, e' : t_2} \qquad \frac{\Gamma; \sigma \vdash M_1 \nearrow e_1 : t_1 \quad \Gamma; \sigma \vdash M_2 \nearrow e_2 : t_2}{\Gamma; \sigma \vdash \langle M_1, M_2\rangle \nearrow \langle e_1, e_2\rangle : t_1 \times t_2}$$

$$\frac{\Gamma; \sigma \vdash M \nearrow e : t_1 \times t_2 \qquad i \in \{1, 2\}}{\Gamma; \sigma \vdash \pi_i(M) \nearrow \pi_i(e) : t_i} \qquad \frac{\Gamma; \sigma \vdash M \nearrow e : t_i \qquad i \in \{1, 2\}}{\Gamma; \sigma \vdash \iota_i(M) \nearrow \iota_i(e) : t_1 + t_2}$$

$$\frac{\Gamma; \sigma \vdash M \nearrow e : t_1 + t_2 \qquad \Gamma, x_1 : t_1; \sigma \vdash M_1 \nearrow e_1 : t \qquad \Gamma, x_2 : t_2; \sigma \vdash M_2 \nearrow e_2 : t}{\Gamma; \sigma \vdash (\mathbf{case}\, M\, \mathbf{of}\, \iota_1(x_1).M_1 \mid \iota_2(x_2).M_2) \nearrow (\mathbf{case}\, e\, \mathbf{of}\, \iota_1(x_1).e_1 \mid \iota_2(x_2).e_2) : t}$$

$$\frac{a \notin \mathit{dom}(\sigma) \quad \Gamma; \sigma\{a \mapsto k\} \vdash M \nearrow e : t}{\Gamma; \sigma \vdash \lambda k{:}\alpha_a.\, M \nearrow [e]_a : [t]_a} \qquad \frac{a \in \mathit{dom}(\sigma) \quad \Gamma; \sigma\{a \mapsto k\} \vdash [k/\sigma(a)]M \nearrow e : t}{\Gamma; \sigma \vdash \lambda k{:}\alpha_a.\, M \nearrow [e]_a : [t]_a}$$

$$\frac{\Gamma; \sigma \vdash M \nearrow e : [t]_a \qquad \Gamma^\dagger, \sigma^\dagger \vdash M' : \alpha_a}{\Gamma; \sigma \vdash M\, M' \nearrow e^a : t}$$

The second to last rule says that some occurrences of keys for a can be abstracted as long as the $\lambda^{[]}$ term after sealing is still at the same level ($\mathit{dom}(\sigma) = \mathit{dom}(\sigma\{a \mapsto k\})$). Note that even if $\Gamma^\dagger, \sigma^\dagger \vdash M : t^\dagger$, the inverse translation of M is *not* always possible. However, we can give a sufficient condition for the inverse translation to exist and show the inverse translation also preserves typing:

Theorem 7 (Inverse Translation Preserves Typing). *If the derivation of $\Gamma^\dagger, \sigma^\dagger \vdash M : t^\dagger$ satisfies the subformula property, then there exists a $\lambda^{[]}$ term e such that $\Gamma; \sigma \vdash M \nearrow e : t$ and Γ ; $\mathit{dom}(\sigma) \vdash e : t$.*

Proof. By induction on the derivation of $\Gamma^\dagger$, $\sigma^\dagger \vdash M : t^\dagger$.

Remark 3. In the above theorem, the subformula property gives a sufficient condition to exclude "junk" terms such as $(\lambda x{:}\alpha_a \to \alpha_a.\, ())(\lambda k{:}\alpha_a.\, k)$. Since $\lambda k{:}\alpha_a.\, k$ has type $\alpha_a \to \alpha_a$, no rules of inverse translation can be applied and the inverse translation will fail. Its derivation, however, does not satisfy the subformula property, so this is not a counterexample for the theorem above. (Its normal form can be translated back to a $\lambda^{[]}$ term.)

4 Proof of Noninterference via Preservation of Logical Relations

In this section, we give an indirect proof of the noninterference theorem, which is obtained as an easy corollary of the theorem that the translation is sound and

complete, that is, the logical relation for $\lambda^{[]}$ is preserved by translation to $\lambda^{\to}$. The properties we would expect are

> *If* $e_1 \approx_{dom(\sigma)} e_2 : t$ *and* $\cdot; \sigma \vdash e_i : t \searrow M_i$ *for* $(i = 1, 2)$, *then* $\sigma^\dagger \vdash M_1 \approx M_2 : t^\dagger$,

and its converse

> *If* $\cdot; \sigma \vdash e_i : t \searrow M_i$ *for* $(i = 1, 2)$ *and* $\sigma^\dagger \vdash M_1 \approx M_2 : t^\dagger$, *then* $e_1 \approx_{dom(\sigma)} e_2 : t$.

It is not very easy, however, to prove them directly because logical relations are defined by induction on types whereas the translations are not. Thus, following Tse and Zdancewic [17], we introduce another logical relation (called *logical correspondence*) $e \approx\!\!\!\!\!\sim_\sigma M : t$ over terms of $\lambda^{[]}$ and $\lambda^{\to}$, then prove that it includes (the graphs of) the translations of both directions (Theorems 9 and 10). Then, after showing that the logical correspondence is full (Corollary 1), we finally prove preservation of logical relations by logical correspondence and reduce the noninterference theorem to the basic lemma (Lemma 2).

4.1 Logical Correspondence and Its Fullness

Definition 8 (Logical Correspondence). *The relations* $e \approx\!\!\!\!\!\sim_\sigma M : t$ *and* $v \rightsquigarrow_\sigma V : t$, *where we assume that* $\Gamma; \ell \vdash e : t$ *and* $\Gamma; \ell \vdash v : t$ *and* $\Delta \vdash M : A$ *and* $\Delta \vdash V : A$, *are defined as the least relation closed under the following rules:*

$$() \rightsquigarrow_\sigma () : \mathit{unit} \qquad \frac{\forall (e \approx\!\!\!\!\!\sim_\sigma M : t_1).\, v\, e \approx\!\!\!\!\!\sim_\sigma V\, M : t_2}{v \rightsquigarrow_\sigma V : t_1 \to t_2} \qquad \frac{v_1 \rightsquigarrow_\sigma V_1 : t_1 \qquad v_2 \rightsquigarrow_\sigma V_2 : t_2}{\langle v_1, v_2 \rangle \rightsquigarrow_\sigma \langle V_1, V_2 \rangle : t_1 \times t_2}$$

$$\frac{v \rightsquigarrow_\sigma V : t_i \qquad i \in \{1, 2\}}{\iota_i(v) \rightsquigarrow_\sigma \iota_i(V) : t_1 + t_2} \qquad \frac{\forall (\sigma^\dagger \vdash M : \alpha_a).\, v \approx\!\!\!\!\!\sim_\sigma V\, M : t}{[v]_a \rightsquigarrow_\sigma V : [t]_a}$$

$$\frac{e \longrightarrow^* v \qquad M \longrightarrow^* V \qquad v \rightsquigarrow_\sigma V : t}{e \approx\!\!\!\!\!\sim_\sigma M : t}$$

Intuitively, $e \approx\!\!\!\!\!\sim_\sigma M : t$ means that e and M exhibit the same behavior from the viewpoint of an observer at $dom(\sigma)$. The rule for $[t]_a$ expresses the fact that the existence of well-typed M of α_a under $\sigma^\dagger$ is equivalent to the existence of the authority a in $dom(\sigma)$. In other words, if a is not in $dom(\sigma)$, the premise is vacuously true, representing that the observer cannot distinguish anything.

Theorem 8 below shows that the logical correspondences are closed under the composition with the logical relation in $\lambda^{\to}$.

Theorem 8. *If* $e \approx\!\!\!\!\!\sim_\sigma M_1 : t$ *and* $\sigma^\dagger \vdash M_1 \approx M_2 : t^\dagger$, *then* $e \approx\!\!\!\!\!\sim_\sigma M_2 : t$.

Proof. By induction on the structure of t, using Remark 2 in the case where $t = [t']_a$.

The next theorem shows that these logical correspondences include the graphs of the translation to $\lambda^{\rightarrow}$. We write $\gamma \approx\!\!\!\!\!\rightarrow_{\sigma} \delta : \Gamma$ if $dom(\gamma) = dom(\delta) = dom(\Gamma)$ and $\gamma(x) \approx\!\!\!\!\!\rightarrow_{\sigma} \delta(x) : \Gamma(x)$ for any $x \in dom(\Gamma)$.

Theorem 9 (Inclusion of Translation). *If $\Gamma; \sigma \vdash e : t \searrow M$ and $\gamma \approx\!\!\!\!\!\rightarrow_{\sigma} \delta : \Gamma$, then $\gamma(e) \approx\!\!\!\!\!\rightarrow_{\sigma} \delta(M) : t$.*

Proof. By induction on the size of the derivation of $\Gamma; \sigma \vdash e : t \searrow M$.

It is slightly harder to show that the logical correspondence includes the graphs of the inverse translation, since the inverse translation is *not* quite a (right) inverse of the translation to $\lambda^{\rightarrow}$: The inverse translation followed by the forward translation may yield a term different from the original. For example, we can derive

$$x : [t]_a; \{a \mapsto k_1\} \vdash \lambda k_2 : \alpha_a.x\ k_1 \nearrow [x^a]_a : [t]_a$$

and

$$x : [t]_a; \{a \mapsto k_1\} \vdash [x^a]_a : [t]_a \searrow \lambda k_2' : \alpha_a.x\ k_2' \ .$$

Fortunately, the difference is only slight: They differ only in subterms of base types α_a and are in fact logically related. To identify terms only with this kind of differences, we introduce a (typed) equivalence relation $\Delta \vdash M_1 \equiv M_2 : A$, which is shown to be included in the logical relation.

Definition 9. *The relation $\Delta \vdash M_1 \equiv M_2 : A$ is defined as the least relation closed under the rules below:*

$$\frac{\Delta \vdash M : A}{\Delta \vdash M \equiv M : A} \qquad \frac{\Delta \vdash M_1 \equiv M_2 : A}{\Delta \vdash M_2 \equiv M_1 : A} \qquad \frac{\Delta \vdash M_1 : \alpha_a \qquad \Delta \vdash M_2 : \alpha_a}{\Delta \vdash M_1 \equiv M_2 : \alpha_a}$$

$$\frac{\Delta \vdash M_1 \equiv M_2 : A \qquad \Delta \vdash M_2 \equiv M_3 : A}{\Delta \vdash M_1 \equiv M_3 : A} \qquad \frac{\Delta \vdash M_1 \equiv M_2 : A \qquad \Delta' \vdash \mathcal{C}[M_1] : A'}{\Delta' \vdash \mathcal{C}[M_1] \equiv \mathcal{C}[M_2] : A'}$$

where $\mathcal{C}$ ranges over term contexts*, which are defined by:*

$$\begin{aligned}\mathcal{C} ::= {} & [\,] \mid \lambda x{:}A.\mathcal{C} \mid \mathcal{C}\,M \mid M\,\mathcal{C} \mid \langle \mathcal{C}, M\rangle \mid \langle M, \mathcal{C}\rangle \mid \pi_i(\mathcal{C}) \mid \iota_i(\mathcal{C}) \\ & \mid (\mathbf{case}\,\mathcal{C}\,\mathbf{of}\,\iota_1(x_1).M \mid \iota_2(x_2).M) \mid (\mathbf{case}\,M\,\mathbf{of}\,\iota_1(x_1).\mathcal{C} \mid \iota_2(x_2).M) \\ & \mid (\mathbf{case}\,M\,\mathbf{of}\,\iota_1(x_1).M \mid \iota_2(x_2).\mathcal{C})\end{aligned}$$

As mentioned above, the inverse translation followed by the translation yields a different term but it is still related by $\equiv$, which is included by the logical relation:

Lemma 3. *If $\Gamma; \sigma \vdash M \nearrow e : t$ and $\Gamma; \sigma \vdash e : t \searrow M'$, then $\Gamma^{\dagger}, \sigma^{\dagger} \vdash M \equiv M' : t^{\dagger}$.*

Proof. By induction on $\Gamma; \sigma \vdash M \nearrow e : t$.

Lemma 4. *If $\Delta \vdash M_1 \equiv M_2 : A$ and $\Delta' \vdash \delta_1 \approx \delta_2 : \Delta$, then $\Delta' \vdash \delta_1(M_1) \approx \delta_2(M_2) : A$.*

Proof. By induction on the derivation of $\Delta \vdash M_1 \equiv M_2 : A$.

Then, we can show the following theorem:

Theorem 10 (Inclusion of Inverse Translation). *If $\Gamma; \sigma \vdash M \nearrow e : t$ and $\gamma \approx\!\!\!\!\!\!\rightarrow_\sigma \delta : \Gamma$, then $\gamma(e) \approx\!\!\!\!\!\!\rightarrow_\sigma \delta(M) : t$.*

Proof. By Theorem 6, there exists M' such that $\Gamma; \sigma \vdash e : t \searrow M'$. Then, by Lemma 3, $\Gamma^\dagger$, $\sigma^\dagger \vdash M \equiv M' : t^\dagger$. Since $\sigma^\dagger \vdash \delta \approx \delta : \Gamma^\dagger$ (using Remark 2), $\sigma^\dagger \vdash \delta(M) \approx \delta(M') : t^\dagger$ by Lemma 4. Then, by Theorem 9, $\gamma(e) \approx\!\!\!\!\!\!\rightarrow_\sigma \delta(M') : t$ and, by Theorem 8 and the symmetricity of the logical relation, $\gamma(e) \approx\!\!\!\!\!\!\rightarrow_\sigma \delta(M) : t$.

As a corollary, the logical correspondences is shown to be full.

Corollary 1 (Fullness of Logical Correspondences). *If $\sigma^\dagger \vdash M : t^\dagger$, then there exists a $\lambda^{[]}$ term e such that $e \approx\!\!\!\!\!\!\rightarrow_\sigma M : t$.*

Proof. By Theorems 5, 7, and 10.

4.2 Preservation of Logical Relations

By using the logical correspondence introduced above, we prove that the logical relations are preserved by the logical correspondence.

Theorem 11 (Preservation of Equivalences).

1. *If $e_1 \approx_{dom(\sigma)} e_2 : t$ and $e_i \approx\!\!\!\!\!\!\rightarrow_\sigma M_i : t$ $(i = 1, 2)$, then $\sigma^\dagger \vdash M_1 \approx M_2 : t^\dagger$.*
2. *Conversely, if $e_i \approx\!\!\!\!\!\!\rightarrow_\sigma M_i : t$ for $(i = 1, 2)$ and $\sigma^\dagger \vdash M_1 \approx M_2 : t^\dagger$, then $e_1 \approx_{dom(\sigma)} e_2 : t$.*

Proof. We prove both simultaneously by induction on the structure of t. We show only the main cases:

Case 1 ($t = t_1 \to t_2$). To show (1), take arbitrary M_1' and M_2' such that $\sigma^\dagger \vdash M_1' \approx M_2' : t_1^\dagger$. By the fullness (Corollary 1), there exist e_i' such that $e_i' \approx\!\!\!\!\!\!\rightarrow_\sigma M_i' : t_1$ $(i = 1, 2)$, and by the induction hypothesis (2) for t_1, we have $e_1' \approx_{dom(\sigma)} e_2' : t_1$. Then, by definition, there exist v_i, V_i $(i = 1, 2)$ such that $e_i \longrightarrow^* v_i$ and $M_i \longrightarrow^* V_i$ and $v_i\, e_i' \approx\!\!\!\!\!\!\rightarrow_\sigma V_i\, M_i' : t_2$ for $(i = 1, 2)$, and $v_1\, e_1' \approx_{dom(\sigma)} v_2\, e_2' : t_2$. Applying the induction hypothesis (1) for t_2 to them, $\sigma^\dagger \vdash V_1\, M_1' \approx V_2\, M_2' : t_2^\dagger$. So we have $\sigma^\dagger \vdash V_1 \sim V_2 : t_1^\dagger \to t_2^\dagger$, and hence $\sigma^\dagger \vdash M_1 \approx M_2 : t_1^\dagger \to t_2^\dagger$. The statement (2) can be shown similarly, *without* the fullness.

Case 2 ($t = [t_1]_a$). To show (2), we have two subcases: $a \in dom(\sigma)$ or not. If $a \in dom(\sigma)$, then, by definition, $\sigma^\dagger \vdash \sigma(a) \approx \sigma(a) : \alpha_a$. Also, by definition, there exist v_i, V_i $(i = 1, 2)$ such that $e_i \longrightarrow^* [v_i]_a$ and $M_i \longrightarrow^* V_i$ and $v_i \approx\!\!\!\!\!\!\rightarrow_\sigma V_i\, \sigma(a) : t_1$ for $(i = 1, 2)$, and $\sigma^\dagger \vdash V_1\, \sigma(a) \approx V_2\, \sigma(a) : t_1^\dagger$. Applying the induction hypothesis (2) for t_1, we have $v_1 \approx_{dom(\sigma)} v_2 : t_1$, which is equivalent to $v_1 \sim_{dom(\sigma)} v_2 : t_1$, so $e_1 \approx_{dom(\sigma)} e_2 : [t_1]_a$. The case $a \notin \ell$ is trivial. Showing (1) is easy.

4.3 Noninterference

Then, we prove the noninterference theorem by reducing it to Lemma 2.

Corollary 2 (Noninterference). *If* $\Gamma ; \ell \vdash e : t$ *and* $\gamma_1 \approx_\ell \gamma_2 : \Gamma$*, then* $\gamma_1(e) \approx_\ell \gamma_2(e) : t$.

Proof. Choose an arbitrary σ such that $dom(\sigma) = \ell$ and $ran(\sigma) \cap dom(\Gamma) = \emptyset$. By Theorem 6, $\Gamma; \sigma \vdash e : t \searrow M$ and $\Gamma^\dagger, \sigma^\dagger \vdash M : t^\dagger$ for some M. Similarly, for any $x \in dom(\gamma_i)$ $(i = 1, 2)$, there exists M_{xi} such that $\cdot; \sigma \vdash \gamma_i(x) : \Gamma(x) \searrow M_{xi}$ and $\Gamma^\dagger, \sigma^\dagger \vdash M_{xi} : (\Gamma(x))^\dagger$. Define δ_i $(i = 1, 2)$ as a simultaneous substitution such that $dom(\delta_i) = dom(\gamma_i)$ and $\delta_i(x) = M_{xi}$ for $x \in dom(\delta_i)$. Then, by Theorem 9, $\gamma_i \approx\!\!\!\!\!\!\rightarrow_\sigma \delta_i : \Gamma$ for $(i = 1, 2)$ and so $\gamma_i(e) \approx\!\!\!\!\!\!\rightarrow_\sigma \delta_i(M) : t$ for $(i = 1, 2)$. By applying Theorem 11(1) to the assumption $\gamma_1 \approx_\ell \gamma_2 : \Gamma$, we have $\sigma^\dagger \vdash \delta_1 \approx \delta_2 : \Gamma^\dagger$. Thus, by Lemma 2 (with Remark 2), $\sigma^\dagger \vdash \delta_1(M) \approx \delta_2(M) : t^\dagger$. Finally, by Theorem 11(2), $\gamma_1(e) \approx_\ell \gamma_2(e) : t$.

5 Related Work

Proofs of Noninterference. There are many ways to prove noninterference theorems for type-based dependency analyses for higher-order languages. For example, Heintze and Riecke [7] and Abadi et al. [1] showed the noninterference theorem for SLam by using denotational semantics. Pottier and Simonet [12] proved it for Core ML with non-standard operational semantics. Moreover, Miyamoto and Igarashi [10], in the study of a modal typed calculus $\lambda_s^\Box$, showed that the noninterference theorem for certain types can be easily proved only by using simple nondeterministic reduction system. In comparison with these proofs, the proof technique presented in this paper might seem overwhelming to show only noninterference; nevertheless, we believe it is still interesting since the translation shows that the notion of dependency can be captured only in terms of simple types.

Fullness in DCC. As we mentioned in the introduction, the translation from DCC to System F given by Tse and Zdancewic is not full. Here, we explain the reason, after quickly reviewing DCC. DCC [1] is an extension of a computational λ-calculus [11] and uses monads (indexed by a security level) for sealing. Roughly speaking, a monadic type $T_\ell\ t$, the monadic unit $\eta_\ell\ e$, and the bind operation $\texttt{bind}\ x = e_1\ \texttt{in}\ e_2$ correspond to types for sealing, sealing terms, and unsealing terms. A type judgment of DCC lacks a level; instead, the notion of protected types is introduced to prevent information leakage and plays a key role in the typing rule for `bind`:

$$\frac{\Gamma \vdash e_1 : T_\ell\ t_1 \qquad \Gamma, x : t_1 \vdash e_2 : t_2 \qquad t_2 \text{ is protected at } \ell}{\Gamma \vdash \texttt{bind}\ x = e_1\ \texttt{in}\ e_2 : t_2}$$

Intuitively, "t is protected at ℓ" means that observers only at level ℓ (or higher) can obtain some bits of information by using the value of t. For example, $T_\ell\ t$

and $T_\ell\ t_1 \times T_\ell\ t_2$ and $t \to T_\ell\ t'$ are all protected at ℓ but $T_\ell\ t_1 + T_\ell\ t_2$ is not (see Abadi et al. [1] for the precise definition). So, this rule ensures that the value of the whole term cannot be examined at unrelated levels. However, `bind` is restrictive in the sense that η_ℓ must be placed within the scope of x to make t_2 protected. For example, the term $\lambda y : T_\ell\ \texttt{bool}.\texttt{bind}\ x = y\ \texttt{in}\ \eta_\ell\ x$ is given type $(T_\ell\ \texttt{bool}) \to (T_\ell\ \texttt{bool})$ while $\lambda y : T_\ell\ \texttt{bool}.\eta_\ell\ (\texttt{bind}\ x = y\ \texttt{in}\ x)$ cannot.

In fact, this restriction is a source of the failure of fullness of the translation by Tse and Zdancewic. Consider the DCC type $T_\ell((T_\ell\ \texttt{bool}) \to \texttt{bool})$, which is translated to $\alpha_\ell \to ((\alpha_\ell \to \texttt{bool}) \to \texttt{bool})$. Then, any DCC terms of the first type is equivalent to (sealed) constant functions $\eta_\ell(\lambda x : T_\ell\ \texttt{bool}.c)$ where c is either `true` or `false`. In System F, however, there is a term $\lambda k : \alpha_\ell.\lambda f\colon \alpha_\ell \to \texttt{bool}.fk$ of the translated type and it would correspond to an *ill typed* DCC term $\eta_\ell(\lambda y : T_\ell\ \texttt{bool}.\texttt{bind}\ x = y\ \texttt{in}\ x)$. From this, we can show their translation does not preserve the logical relations. In fact,

$$\lambda f.\texttt{bind}\ f' = f\ \texttt{in}\ \eta_\ell\ (f'\ (\eta_\ell\ \texttt{true}))$$

and

$$\lambda f.\texttt{bind}\ f' = f\ \texttt{in}\ \eta_\ell\ (f'\ (\eta_\ell\ \texttt{false}))$$

are logically related at the type $(T_\ell((T_\ell\ \texttt{bool}) \to \texttt{bool})) \to (T_\ell\ \texttt{bool})$ and ℓ since, in DCC, all we can pass to these functions are the constant functions above. Their translations, however, are not because, in System F, applying them to the term $\lambda k : \alpha_\ell.\lambda f\colon \alpha_\ell \to \texttt{bool}.fk$ above will distinguish them.

Tse and Zdancewic's Extended DCC. Interestingly, Tse and Zdancewic also noticed this restriction of DCC and proposed an extension of (a pure fragment of) DCC by introducing the notion of protection contexts in type judgments. This extension allows terms like $\lambda y : T_\ell\ \texttt{bool}.\eta_\ell(\texttt{bind}\ x = y\ \texttt{in}\ x)$ and $\eta_\ell(\lambda y : T_\ell\ \texttt{bool}.\texttt{bind}\ x = y\ \texttt{in}\ x)$ to be well typed. Our $\lambda^{[]}$ can be considered a simplification of this extension by dropping the notion of protected types completely while leaving protection contexts (namely, levels in type judgments). We also dropped the lattice structure of security levels in DCC and now call them authorities. We believe that a similar result can be shown when the set of authorities is equipped with such a structure.

6 Conclusion

We have formalized noninterference for a typed λ-calculus $\lambda^{[]}$ by logical relations and proved by reducing it to the basic lemma of logical relation for $\lambda^{\to}$ through a translation of $\lambda^{[]}$ to $\lambda^{\to}$. Our translation is sound and fully complete and, as a result, the image of the translation is a complete representation, which captures dependency of $\lambda^{[]}$ with typeability in $\lambda^{\to}$.

Acknowledgements. Comments from anonymous referees helped up improve the final presentation. We thank Masahito Hasegawa, Eijiro Sumii, Stephen Tse, and Steve Zdancewic for discussions on this subject. This work is supported in part by Grant-in-Aid for Scientific Research (B) No. 17300003.

References

[1] Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. A core calculus of dependency. In *POPL '99: Proceedings of 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 147–160, New York, NY, USA, 1999. ACM Press.

[2] Philippe de Groote. On the strong normalisation of intuitionistic natural deduction with permutative-conversions. *Information and Computation*, 178:441–464, August 2002.

[3] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. *Communications of the ACM*, 20(7):504–513, July 1977.

[4] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972. A summary appeared in the Proceedings of the Second Scandinavian Logic Symposium (J.E. Fenstad, editor), North-Holland, 1971 (pp. 63–92).

[5] J. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 11–20, 1982.

[6] Masahito Hasegawa. Girard translation and logical predicates. *Journal of Functional Programming*, 10(1):77–89, January 2000.

[7] Nevin Heintze and Jon G. Riecke. The SLam calculus: programming with secrecy and integrity. In *POPL '98: Proceedings of ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 365–377, 1998.

[8] Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice-Hall, 1993.

[9] John C. Mitchell. *Foundations for Programming Languages*. The MIT Press, 1996.

[10] Kenji Miyamoto and Atsushi Igarashi. A modal foundation for secure information flow. In *FCS '04: Proceedings of Workshop on Foundations of Computer Security*, pages 187–203, June 2004.

[11] Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 1:55–92, 1991.

[12] François Pottier and Vincent Simonet. Information flow inference for ML. *ACM Transactions on Programming Languages and Systems*, 25(1):117–158, 2003.

[13] John Reynolds. Towards a theory of type structure. In *Proc. Colloque sur la Programmation*, pages 408–425, New York, 1974. Springer-Verlag LNCS 19.

[14] John C. Reynolds. Types, abstraction and parametric polymorphism. In *IFIP Congress*, pages 513–523, 1983.

[15] Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal On Selected Areas In Communications*, 21(1):5–19, 2003.

[16] Yan Mei Tang and Pierre Jouvelot. Effect systems with subtyping. In *Proceedings of ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation(PEPM'95)*, pages 45–53, 1995.

[17] Stephen Tse and Steve Zdancewic. Translating dependency into parametricity. In *ICFP '04: Proceedings of 9th ACM International Conference on Functional Programming*, pages 115–125, New York, NY, USA, 2004. ACM Press.

[18] Philip Wadler. Theorems for free! In *Proceedings 4th Int. Conf. on Funct. Prog. Languages and Computer Arch., FPCA'89, London, UK, 11–13 Sept 1989*, pages 347–359. ACM Press, New York, 1989.