Mechanized Logical Relations for Termination-Insensitive Noninterference

Simon O. Gregersen
joint work with Johan Bay, Amin Timany, and Lars Birkedal

POPL 2021, January 20 - 22, Online
The prevailing basic semantic notion of secure information flow is noninterference.
Program $e$ satisfies termination-insensitive noninterference, abbr. TINI($e$), when

$$e[v_1/x] \Downarrow o_1 \quad \text{and} \quad e[v_2/x] \Downarrow o_2 \quad \text{implies} \quad o_1 \approx o_2$$

for all secrets $v_1$ and $v_2$. 
Information-flow control enforcement is often specified using a static type system:

$$\Gamma \vdash e : t^\ell \quad \text{implies} \quad \text{TINI}(e)$$

To be useful, it must support the same features as modern programming languages:

- higher types,
- reference types,
- abstract types,
- ...

The difficulty of proving the type system sound, however, increases.
The problem

Information-flow control enforcement is often specified using a static type system:

\[ \Gamma \vdash e : t^\ell \quad \text{implies} \quad \text{TINI}(e) \]

To be useful, it must support the same features as modern programming languages:

- higher types,
- reference types,
- abstract types,
- ...

The difficulty of proving the type system sound, however, increases.
The main goal of this work is to

- show that such a rich type system satisfies \textit{termination-insensitive noninterference}
- using a semantic model

\[ \Gamma, x : \tau_2 \vdash e_1 : \tau_1 \quad \text{and} \quad e_2 \in [\tau_2] \quad \text{then} \quad \text{TINI}(e_1[e_2/x]) \]

- with full mechanization of all results in Coq
This work

The main goal of this work is to

• show that such a rich type system satisfies **termination-insensitive noninterference**
• using a semantic model
  ⇒ compositional integration of syntactically well-typed and ill-typed components:

\[ \Gamma, x : \tau_2 \vdash e_1 : \tau_1 \quad \text{and} \quad e_2 \in \llbracket \tau_2 \rrbracket \quad \text{then} \quad \text{TINI}(e_1[e_2/x]) \]

• with full mechanization of all results in Coq
Example (Multiplying by zero)

\[ \lambda v. v \ast 0 \]

cannot be syntactically typed at \( \mathbb{N}^\top \rightarrow \mathbb{N}^\perp \).

Example (Temporary explicit leak)

\[ \text{let } x = ! \text{ in } l \leftarrow ! h; \ldots; l \leftarrow x \]

is not syntactically well-typed.

More interesting examples found at the end of the presentation and in the paper.
Example (Multiplying by zero)

\[ \lambda v. v \times 0 \]

cannot be syntactically typed at \( \mathbb{N}^T \rightarrow \mathbb{N}^\bot \).

Example (Temporary explicit leak)

```plaintext
let x = !l in l ← !h; . . . ; l ← x
```

is not syntactically well-typed.

More interesting examples found at the end of the presentation and in the paper.
Consider if secret then f() — if f has public side-effects we would leak secret.

For this presentation, we consider $L = \{\bot, \top\}$ where $\bot \sqsubseteq \top$ and $\top \not\sqsubseteq \bot$.
Consider if secret then \( f() \) — if \( f \) has public side-effects we would leak secret.
Consider if \( \text{secret} \) then \( f() \) — if \( f \) has public side-effects we would leak \( \text{secret} \).

For this presentation, we consider \( \mathcal{L} = \{\bot, \top\} \) where \( \bot \subseteq \top \) and \( \top \nsubseteq \bot \).
Typing judgment

\[ \Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau \]
Typing judgment

Term-level context

\[ \Xi \vdash_\Psi \Gamma \vdash_{pc} e : \tau \]
Typing judgment

Term-level context

\[ \Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau \]

Type-level context
Typing judgment

\[ \Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau \]

Term-level context

Type-level context

Label context
Typing judgment

\[ \Xi \vdash \Psi \mid \Gamma \vdash_{pc} e : \tau \]

- Term-level context
- Type-level context
- Program counter label
- Label context
Type system

\[
\begin{align*}
\text{T-IF} & \quad \Xi \mid \Psi \mid \Gamma \vdash_{pc} e : B^\ell & \forall i \in \{1, 2\} \cdot \Xi \mid \Psi \mid \Gamma \vdash_{pc \sqcup \ell} e_i : \tau & \quad \Psi \vdash \tau \downharpoonright \ell \\
\Xi \mid \Psi \mid \Gamma \vdash_{pc} \text{if } e \text{ then } e_1 \text{ else } e_2 : \tau
\end{align*}
\]

\[
\begin{align*}
\text{T-STORE} & \quad \Xi \mid \Psi \mid \Gamma \vdash_{pc} e_1 : \text{ref}(\tau)^\ell & \Xi \mid \Psi \mid \Gamma \vdash_{pc} e_2 : \tau & \quad \Psi \vdash \tau \downharpoonright \neg \text{pc} \sqcup \ell \\
\Xi \mid \Psi \mid \Gamma \vdash_{pc} e_1 \leftarrow e_2 : 1^\bot
\end{align*}
\]

\[
\begin{align*}
\text{T-TLAM} & \quad \Xi, \alpha \mid \Psi \mid \Gamma \vdash_{\ell} e : \tau \\
\Xi \mid \Psi \mid \Gamma \vdash_{pc} \Lambda e : (\forall_{\ell} \alpha. \tau)^\bot
\end{align*}
\]
Type system

T-IF
\[ \forall i \in \{1, 2\}. \Xi \;
T-STORE
\[ \Xi \;
T-TLAM
\[ \Xi, \alpha \;
\]
Type system

**T-IF**
\[ \begin{array}{c}
\Xi | \Psi | \Gamma \vdash_{pc} e : B^{\ell} \\
\forall i \in \{1, 2\} . \Xi | \Psi | \Gamma \vdash_{pc \uplus \ell} e_i : \tau \\
\Psi \vdash \tau \downarrow \ell
\end{array} \]

\[ \Xi | \Psi | \Gamma \vdash_{pc} \text{if } e \text{ then } e_1 \text{ else } e_2 : \tau \]

**T-STORE**
\[ \begin{array}{c}
\Xi | \Psi | \Gamma \vdash_{pc} e_1 : \text{ref}(\tau)^{\ell} \\
\Xi | \Psi | \Gamma \vdash_{pc} e_2 : \tau \\
\Psi \vdash \tau \downarrow \downarrow pc \uplus \ell
\end{array} \]

\[ \Xi | \Psi | \Gamma \vdash_{pc} e_1 \leftarrow e_2 : 1^{\perp} \]

**T-TLAM**
\[ \begin{array}{c}
\Xi, \alpha | \Psi | \Gamma \vdash_{\ell_e} e : \tau
\end{array} \]

\[ \Xi | \Psi | \Gamma \vdash_{pc} \Lambda e : (\forall_{\ell_e} \alpha . \tau)^{\perp} \]
Theorem (Termination-Insensitive Noninterference)

If

\[
x : \mathbb{B}^T \vdash \bot : \mathbb{B}^\perp,
\]

\[
\vdash \bot : \mathbb{B}^T,
\]

and

\[
\vdash \bot : \mathbb{B}^T
\]

then

\[
(\emptyset, e[v_1/x]) \rightarrow^* (\sigma_1, v'_1) \text{ and } (\emptyset, e[v_2/x]) \rightarrow^* (\sigma_2, v'_2) \text{ then } v'_1 = v'_2.
\]
Theorem (Termination-Insensitive Noninterference)

If

\[ x : B^\top \vdash_e B^\bot, \quad \vdash v_1 : B^\top, \quad \text{and} \quad \vdash v_2 : B^\top \]

then

\[ (\emptyset, e[v_1/x]) \rightarrow^* (\sigma_1, v'_1) \quad \text{and} \quad (\emptyset, e[v_2/x]) \rightarrow^* (\sigma_2, v'_2) \quad \text{then} \quad v'_1 = v'_2. \]
Theorem (Termination-Insensitive Noninterference)

If

\[ x : \mathbb{B}^\top \vdash_\bot e : \mathbb{B}^\bot, \quad \vdash_\bot v_1 : \mathbb{B}^\top, \quad \text{and} \quad \vdash_\bot v_2 : \mathbb{B}^\top \]

then

\[ (\emptyset, e[v_1/x]) \rightarrow^* (\sigma_1, v'_1) \quad \text{and} \quad (\emptyset, e[v_2/x]) \rightarrow^* (\sigma_2, v'_2) \quad \text{then} \quad v'_1 = v'_2. \]
Theorem (Termination-Insensitive Noninterference)

If

\[ x : \mathbb{B}^\top \vdash e : \mathbb{B}^\bot, \quad \vdash v_1 : \mathbb{B}^\top, \quad \text{and} \quad \vdash v_2 : \mathbb{B}^\top \]

then

\[ (\emptyset, e[v_1/x]) \rightarrow^* (\sigma_1, v'_1) \quad \text{and} \quad (\emptyset, e[v_2/x]) \rightarrow^* (\sigma_2, v'_2) \quad \text{then} \quad v'_1 = v'_2. \]
Our approach

We set up a binary (logical) relation

\[ \Xi \mid \Psi \mid \Gamma \vdash e_1 \approx e_2 : \tau \]

such that

\[ \Xi \mid \Psi \mid \Gamma \vdash e : \tau \quad \Rightarrow \quad \Xi \mid \Psi \mid \Gamma \vdash e \approx e : \tau \]

\[ \Xi \mid \Psi \mid \Gamma \vdash e \approx e : \tau \quad \Rightarrow \quad \text{TINI}(e) \]

However, this requires manipulating and defining a complex semantic model.
We combat this complexity by using the separation logic framework Iris.

- Convenient modalities to express the relation,
- High-level logic to reason within, and
- Coq formalization and the Iris Proof Mode to mechanize proofs.
Existing works on “logical” logical relations prove (contextual) refinements. Intuitively, $e_1$ refines $e_2$ if

$$e_1 \xrightarrow{*} v_1 \Rightarrow e_2 \xrightarrow{*} v_2 \land v_1 \approx v_2.$$
Existing works on “logical” logical relations prove (contextual) refinements. Intuitively, $e_1$ refines $e_2$ if

$$e_1 \rightarrow^* v_1 \Rightarrow e_2 \rightarrow^* v_2 \land v_1 \approx v_2.$$ 

However, we need a termination-insensitive notion:

$$e_1 \rightarrow^* v_1 \land e_2 \rightarrow^* v_2 \Rightarrow v_1 \approx v_2.$$
Our approach cont’d

Existing works on “logical” logical relations prove (contextual) refinements.

Intuitively, $e_1$ refines $e_2$ if

$$e_1 \rightarrow^* v_1 \Rightarrow e_2 \rightarrow^* v_2 \land v_1 \approx v_2.$$ 

However, we need a termination-insensitive notion:

$$e_1 \rightarrow^* v_1 \land e_2 \rightarrow^* v_2 \Rightarrow v_1 \approx v_2.$$ 

For this, we define a novel theory of modal weakest preconditions.
A central idea in the model is to interpret types both as a

**Binary relation** for relating terms that are publicly equivalent and as a

**Unary relation** for characterizing terms that do not have public side-effects.
A central idea in the model is to interpret types both as a

**Binary relation** for relating terms that are publicly equivalent and as a

**Unary relation** for characterizing terms that do not have public side-effects.

Consider

\[
\models \text{if } v \text{ then } e_1 \text{ else } e_2 \approx \text{if } v' \text{ then } e_1 \text{ else } e_2 : t^\top
\]

where \( \models v \approx v' : \mathbb{B}^\top \) meaning \( v, v' \in \{\text{true, false}\} \).
A central idea in the model is to interpret types both as a

**Binary relation** for relating terms that are publicly equivalent and as a

**Unary relation** for characterizing terms that do not have public side-effects.

Consider

\[ \models \text{if } v \text{ then } e_1 \text{ else } e_2 \approx \text{if } v' \text{ then } e_1 \text{ else } e_2 : t^\top \]

where \( \models v \approx v' : B^\top \) meaning \( v, v' \in \{\text{true, false}\} \). This means proving, e.g.,

\[ \models e_1 \approx e_2 : t^\top \]
A central idea in the model is to interpret types both as a

**Binary relation** for relating terms that are publicly equivalent and as a

**Unary relation** for characterizing terms that do not have public side-effects.

Consider

\[ \models \text{if } v \text{ then } e_1 \text{ else } e_2 \approx \text{if } v' \text{ then } e_1 \text{ else } e_2 : t^\top \]

where \( \models v \approx v' : \mathbb{B}^\top \) meaning \( v, v' \in \{\text{true, false}\} \). This means proving, e.g.,

\[ \models e_1 \approx e_2 : t^\top \]

Crucially, they may not modify public references.
Recall

\[
\Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau \quad \Rightarrow \quad \Xi \mid \Psi \mid \Gamma \models e \approx e : \tau
\]
\[
\Xi \mid \Psi \mid \Gamma \models e \approx e : \tau \quad \Rightarrow \quad \text{TINI}(e)
\]

Importantly, the semantic relation is **not** defined in terms of the syntactic relation.
Semantic typing

Recall

$$\Xi | \Psi | \Gamma \vdash_{pc} e : \tau \quad \Rightarrow \quad \Xi | \Psi | \Gamma \models e \approx e : \tau$$

$$\Xi | \Psi | \Gamma \models e \approx e : \tau \quad \Rightarrow \quad \text{TINI}(e)$$

Importantly, the semantic relation is not defined in terms of the syntactic relation.

At the same time,

$$x : \tau_2 \models e_1 \approx e_1 : \tau_1 \quad \text{and} \quad \models e_2 \approx e_2 : \tau_2$$

implies

$$\models e_1[e_2/x] \approx e_1[e_2/x] : \tau_1$$
Consider

\[ \text{valDep} \triangleq \lambda f. \text{let } d = \text{ref(true, secret)} \text{ in } \]

\[ f \ d; \]

\[ \text{let } (b, v) = !d \text{ in } \]

\[ \text{if } b \text{ then } 42 \text{ else } v \]
Consider

\[
\text{valDep} \triangleq \lambda f. \text{let } d = \text{ref}(\text{true}, \text{secret}) \text{ in } \\
f \ d; \\
\text{let } (b, v) = !d \text{ in } \\
\text{if } b \text{ then } 42 \text{ else } v
\]

The program does not syntactically type check at \( \mathbb{N}^\perp \).
Value-dependent classification

Consider

\[ \text{valDep} \triangleq \lambda f. \ \text{let } d = \text{ref}(\text{true}, \text{secret}) \ \text{in} \]
\[ f \ d; \]
\[ \text{let } (b, v) = !d \ \text{in} \]
\[ \text{if } b \ \text{then } 42 \ \text{else } v \]

The program does not syntactically type check at \( \mathbb{N} \perp \) but, ideally,

\[ \text{secret} : \mathbb{N}^\top \models \text{valDep } f \approx \text{valDep } f : \mathbb{N} \perp \]

for “well-behaved” \( f \). We can use the logic to express and prove these requirements.
Value-dependent classification cont’d

However, this burdens the client with proof obligations. Instead, we can exploit existential types to conceal the proof obligations. E.g.,

\[
\text{valDepPack} \overset{\Delta}{=} \lambda d. \ \begin{cases} \text{let } (b, v) = !d \text{ in } \text{if } b \text{ then } \text{inj}_1 v \text{ else } \text{inj}_2 v \in \\
\begin{aligned}
\text{let } \text{setL} &= \lambda d, v. d \leftarrow (\text{false}, v) \in \\
\text{let } \text{setH} &= \lambda d, v. d \leftarrow (\text{true}, v) \in \\
\text{pack} \ (\text{ref}(\text{true}, \text{secret}), \text{get}, \text{setL}, \text{setH})
\end{aligned}
\end{cases}
\]

for which it holds

\[
\text{secret} : \mathbb{N}^\top \vdash \text{valDepPack} \simeq \text{valDepPack} : \\
\exists \alpha. \left( \alpha \perp \times \left( \alpha \perp \rightarrow \mathbb{N}^\top + \mathbb{N} \perp \right) \times \left( \alpha \perp \rightarrow \mathbb{N} \perp \rightarrow 1 \right) \times \left( \alpha \perp \rightarrow \mathbb{N}^\top \rightarrow 1 \right) \right)
\]

16
Conclusion

In summary, we have

• defined a novel semantic model of an expressive IFC type system with support for impredicative polymorphism, label polymorphism, recursive types, and general references,
  • unary and binary logical-relations models
  • a theory of Modal Weakest Preconditions

• showed that the type system entails termination-insensitive noninterference, and

• illustrated how the model can be used to reason about syntactically ill-typed but semantically secure code with compositional integration.
Conclusion

In summary, we have

- defined a novel semantic model of an expressive IFC type system with support for impredicative polymorphism, label polymorphism, recursive types, and general references,
  - unary and binary logical-relations models
  - a theory of Modal Weakest Preconditions
- showed that the type system entails termination-insensitive noninterference, and
- illustrated how the model can be used to reason about syntactically ill-typed but semantically secure code with compositional integration.
Thank you for watching

gregersen@cs.au.dk


https://github.com/logsem/iris-tini