A Modal Foundation for Secure Information Flow

Kenji Miyamoto and Atsushi Igarashi
Graduate School of Informatics,
Kyoto University, JAPAN

Background

- Many type-based techniques for information flow analysis(IFA) (e.g. SLam [Heintze and Riecke POPL98])
- However, the essence of the type systems is not very clear
  - Subtle differences among their cores
  - It is not clear whether the differences are essential or not

Our goal

- Clarification of the essence of type-based IFA

↓

- Uniform framework which can represent various type systems for IFA

Approach

- To show a relationship between
  - type-based IFA
  - modal logic
- Development of a typed calculus based on the modal logic
  - Via Curry-Howard isomorphism
- Encoding existing calculi for IFA to $\lambda_S^{\Box}$

Contribution

- We show modal logic of local validity corresponds to type-based IFA
- Formalization of $\lambda_S^{\Box}$ based on the modal logic
  - Simple proof of noninterference
- Encoding of a core of the SLam calculus to $\lambda_S^{\Box}$

Contents

- <u>Information flow analysis</u>
- Modal logic
- $\lambda_S^{\Box}$
- Encoding the SLam calculus
- Related work
- Conclusion and future work

Information flow analysis

- Program analysis to ensure
  - The absence of data leakage
    - e.g. private data(your salary) does not leak to public
  - a.k.a. the noninterference property

Security level

- Level of secrecy of data
- We assign security level to each datum
- Some data have high security level
- Some data have low security level
  - For example, private data(your salary) has higher security level than public data(everybody can read)

Leakage of data

- Two kinds of leakage
  - Direct leakage of data
  - Indirect leakage of data
- IFA detects both kinds of leakage

Direct leakage of data

```
int pub:=0^L; //L means public
int salary:=400^H; //H means private
…
pub:=salary;
print(pub);
```

By printing the value of pub, we can know the value of salary

Indirect leakage of data

int pub:=0^L; //L means public

int salary:=400^H; //H means private

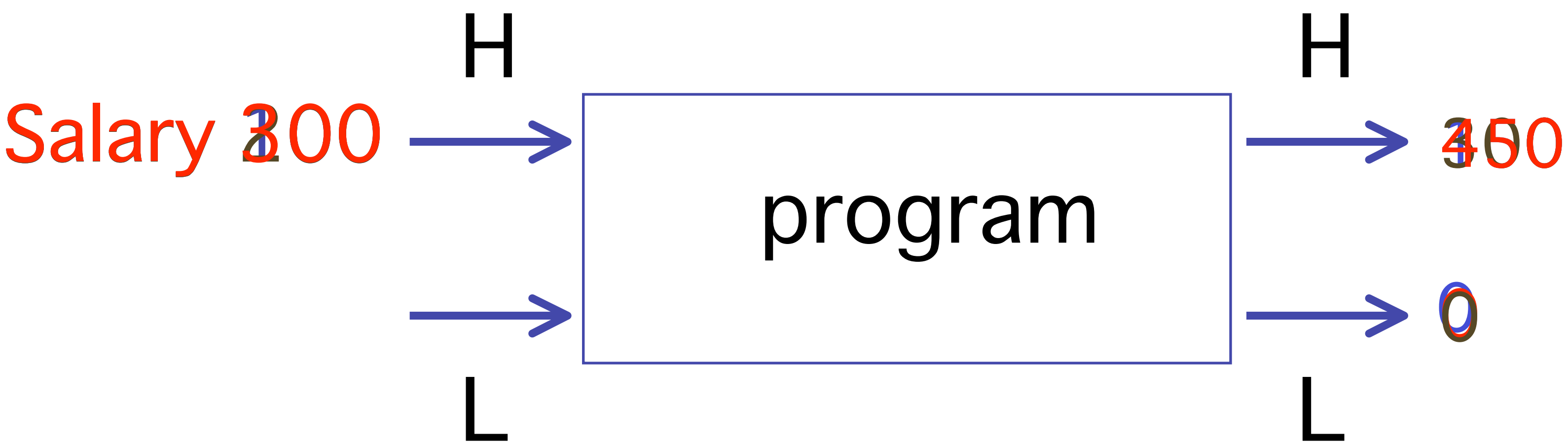
…

if salary>300 then pub:=1 else pub:=2;

By reading a value of pub, we can know whether salary is over 300 or not

Noninterference

- Correctness property of IFA



Whatever high security input is given,
low security output is unchanged

Contents

- Information flow analysis
- Modal logic
- $\lambda_S^{\Box}$
- Encoding the SLam calculus
- Related work
- Conclusion and future work

Relationship between IFA and modal logic

- We can consider
  - Security levels as possible worlds
  - Order of security as reachability relation
    - High security world is reachable from low security world

$$\downarrow$$

What kind of modality is appropriate?

Local validity as modality

- "A holds at all worlds reachable from a certain world *S*"
  - We write it $\Box_S A$
- It is appropriate because, in IFA, low security level data can be read at high security level
  - We represent low security data type as $\Box_{low}$int

Contents

- Information flow analysis
- Modal logic
- $\underline{\lambda}_{s}^{\Box}$
- Encoding the SLam calculus
- Related work
- Conclusion and future work

$\lambda_S^{\Box}$

- Term calculus for logic of local validity
- Extension of simply typed lambda calculus with modal types
- Type system for IFA

Syntax

S: element of poset of security levels

Type $A ::= K \mid A \to A \mid \Box_S A$

Base type $K ::=$ unit | int | string | $\cdots$

Term $M ::= c \mid x \mid u$

$\mid (\lambda x{:}A.M) \mid (MM)$

$\mid (\mathrm{box}_S\, M)$

$\mid (\mathrm{let\ box}_S\, u{=}M \mathrm{\ in\ } M)$

box and let box

- $box_S M$
  - Seals M at security level S
- let box_S u=M in N
  - Unseals M, binds u to the unsealed value, and executes N

Main reduction rules

- $(\lambda x{:}A.M)N \rightarrow [N/x]M$
- let box_S u= box_S M in $N \rightarrow [M/u]N$

Judgment

- Context consists of two parts:
  - Modal context Δ containing locally valid assumptions $u_1{::}^{L1}A_1$, $u_2{::}^{L2}A_2$, ...
  - Ordinary context Γ containing truth assumptions $x_1{:}B_1$, $x_2{:}B_2$, ...
    - c.f. Davies and Pfenning's formalization of modal logic [Davies and Pfenning POPL96]
- Judgments are of the form:

$$\Delta \; ; \; \Gamma \vdash^S M{:}A$$

  - M has type A at level S, under Δ and Γ

Main typing rules(1/3)

Rule for modal variables

$$\frac{u::^{S1} A \in \Delta \qquad S_1 \leqq S_2}{\Delta ; \Gamma \vdash^{S2} u{:}A} \ \text{(T-Mvar)}$$

- Current level S_2 must be reachable from u's level
  - Data readable at low security level S_1 also readable at high security level S_2

Main typing rules(2/3)

Rule for box

$$\frac{\Delta;\cdot\ \vdash^{S1} M{:}A}{\Delta;\Gamma\ \vdash^{S2} \mathrm{box}_{S1}\, M{:}\Box_{S1} A}\ \text{(T-Box)}$$

- The rule corresponds to □-introduction
- The premise means $\Delta;\cdot\ \vdash^{S} M{:}A$ can be derived *for any level* $S \geqq S_1$
  - Ordinary context is empty
  - The levels of modal variables in M are higher than S_1

Main typing rules(3/3)

Rule for let box

$$\frac{\Delta;\Gamma \vdash^{S1} M:\Box_{S2}A \quad \Delta, u::^{S2} A;\Gamma \vdash^{S1} N:B}{\Delta;\Gamma \vdash^{S1} \text{let box}_{S2}\ u = M \text{ in } N:B} \text{(T-Letbox)}$$

- The rule corresponds to $\Box$-elimination
- "$\Box_{S2}A$ is true" turns into "A is valid at S_2"
- We can unseal M :$\Box_{S2}A$ at any security level, but usage of u is limited by the rule T-Mvar

Example

The example of indirect leakage

print:($\Box_L$int)→unit

salary:$\Box_H$int

print(let box$_H$ u=salary in

box$_L$ (if u>300 then 1 else 2))

- We cannot use u in box$_L$ due to T-Mvar. Thus, this program is not typed.

Properties

- Subject reduction
- Church-Rosser
- Strong Normalization
- Noninterference

Noninterference Theorem

- If
  - $u ::^{S} int; \cdot \vdash^{T} M : int$
  - $S > T$
- Then
  - there exists a unique normal form M' such that
    - for any N, if $\vdash^{S} N : int$ then $[N/u]M \to^{*} M'$

Proof sketch

- Lemma
  - If $u::^{S}int\ ;\ \cdot\ \vdash^{T} M:int$ and M is a normal form and $u \in FMV(M)$ then $S \leqq T$
- $\exists! M'$ s.t. $M:int \rightarrow^{*} M':int$ and M' is normal form
- $[N/u]M \rightarrow^{*} [N/u]M' = M'$ (by the contraposition of the lemma)

Contents

- Information flow analysis
- Modal logic
- $\lambda_S^{\Box}$
- <u>Encoding the SLam calculus</u>
- Related work
- Conclusion and future work

SLam calculus[Heintze & Riecke 98]

- Type-based IFA for higher-order language i.e. λ-calculus
- Secure types
  - Security level is attached to each type constructor
    - $T ::= \text{unit}^S \mid \text{int}^S \mid T \to^S T \mid \ldots$

Encoding to $\lambda_S^{\Box}$

- Source: SLam — recursion and protected
- Overview of encoding
  - $\Delta \vdash e{:}t^S \Rightarrow |\Delta|;\ \cdot \vdash^S |e| : |t|$
  - int^H is translated to $\Box_H int$
  - Subsumption translates to coercion
    - $(unit, H) \leqq (unit, L)$ to $\lambda x{:}\Box_L unit.\mathrm{let}\ box_L\ u_x = x\ \mathrm{in}\ u_x$
- Properties
  - Encoding preserves typing
  - Translated programs enjoy noninterference

Contents

- Information flow analysis
- Modal logic
- $\lambda_S^{\Box}$
- Encoding the SLam calculus
- <u>Related work</u>
- Conclusion and future work

Related work(1/2)

- Type-based IFA for functional languages
  - Fairly complex proofs of noninterference using
    - denotational semantics[Heintze and Reicke, POPL98]
    - non-standard operational semantics[Pottier and Simonet TOPLAS03]
  - Noninterference of our system is proved in a simple manner
    - Our proof is similar to the proof of noninterference of $FOb_{1<:}$[Barthe and Serpette FLOPS99]

Related work(2/2)

- DCC[Abadi et al POPL99]
  - A calculus to unify dependency analyses
  - SLam is one of the instances of DCC
  - DCC is monadic type based

  - Monadic types of DCC are similar to modal types in their roles, but
  - Typing rules are rather different

Contents

- Information flow analysis
- Modal logic
- $\lambda_S^{\Box}$
- Encoding the SLam calculus
- Related work
- <u>Conclusion and future work</u>

Conclusion

- Relationship between IFA and modal logic
- $\lambda_S^{\Box}$ enjoys subject reduction, Church-Rosser, strong normalization, and noninterference
- A translation from SLam to $\lambda_s^{\Box}$

Future work

- To compare $\lambda_S^{\square}$ with other calculi for IFA
- To figure out how modal types of $\lambda_s^{\square}$ and monadic types of DCC correspond to each other
- Adding side effects and recursion

End

Kiitos

$$\frac{\Gamma, x{:}s_1 \vdash e_0{:}s_2}{\Gamma \vdash (\lambda x{:}s_1.e_0)_L{:}(s_1 \rightarrow s_2, L)}$$