FLOWSTLC: An Information Flow Control Type System Based On Graded Modality

1st Zhige Chen
Dept. of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12413315@mail.sustech.edu.cn

2nd Junqi Huang
Dept. of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12212226@mail.sustech.edu.cn

3rd Zhiyuan Cao
Dept. of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen, China
12311109@mail.sustech.edu.cn

***Abstract*—In this project, we introduce the design and implementation of a simple functional programming language with a type system that enforces secure information flow. Building on the simply-typed lambda calculus (STLC), we extend the type system with a graded modality with a security semiring. Inspired by Graded Modal Dependent Type Theory (Moon, Eades, Orchard), our system statically prevents unauthorized flows of sensitive data into public outputs, ensuring the noninterference property.**

***Index Terms*—Computer security, information flow, noninterference, security-type systems**

I. INTRODUCTION

II. FLOWSTLC: THE CORE CALCULUS

A. *Syntax*

B. *The Security Semiring*

C. *Typing*

D. *Evaluation*

E. *Graded Modality*

III. METATHEORY

A. *Substitution*

Lemma 1 (Permutation)**.** *If $\Gamma \vdash t : T$ and Δ is a permutation of Γ, then $\Delta \vdash t : T$ and the derivation depth of the latter is the same as the former.*

The proof for the permutation lemma is trivial since assumptions in contexts are unrelated in a simply-typed context. The permutation lemma enables us to state the two substitution lemmas in a cleaner form, we first prove the well-typedness of substitution through regular assumptions:

Lemma 2 (Well-typed Substitution)**.** *If $\Gamma, x : S \vdash t : T$ and $\Delta \vdash s : S$, then $\Gamma, \Delta \vdash [x \mapsto s]t : T$.*

Proof. By induction on a derivation of the form $\Gamma, x : S \vdash t : T$. For a given derivation, we proceed by cases on the final typing rule used in the proof.

- *Case* T-VAR: $t = z$ with $z : T \in (\Gamma, x : S)$
  We need to consider two subcases:
  1) if $z = x$, then $[x \mapsto s]z \to [x \mapsto s]x \to s$. So we need to show $\Gamma, \Delta \vdash s : S$. To get this we apply the T-WEAK rule and the permutation lemma to the assumption $\Delta \vdash s : S$ which will give us desired result.
  2) otherwise $[x \mapsto s]z \to z$, and the result is immediate.
- *Case* T-ABS: $t = \lambda y : T_2.t_1$, $T = T_2 \to T_1$, $\Gamma, x : S, y : T_2 \vdash t_1 : T_1$
  First by the permutation lemma and weakening rule, we have $\Gamma, \Delta, y : T_2, x : S \vdash t_1 : T_1$. Again we apply the two rules on the assumption $\Delta \vdash s : S$ to get $\Gamma, \Delta, y : T_2 \vdash s : S$. By the induction hypothesis, we have $\Gamma, \Delta, y : T_2 \vdash [x \mapsto s]t_1 : T_1$, then by T-ABS $\Gamma, \Delta \vdash \lambda y : T_2.[x \mapsto s]t_1 : T_2 \to T_1$, and this is our desired result.
- *Case* T-APP: $t = t_1\ t_2$, $\Gamma, x : S \vdash t_1 : T_2 \to T_1$, $\Gamma, x : S \vdash t_2 : T_2$, $T = T_1$
  By the induction hypothesis, we have $\Gamma \vdash [x \mapsto s]t_1 : T_2 \to T_1$ and $\Gamma \vdash [x \mapsto s]t_2 : T_2$. Then by T-APP, $\Gamma \vdash [x \mapsto s]t_1\ [x \mapsto s]t_2 : T_1$, i.e., $\Gamma \vdash [x \mapsto s](t_1\ t_2) : T_1$.
- *Case* T-WEAK: $t = t_1$, $\Gamma' \vdash t_1 : T$ where $\Gamma' \subseteq (\Gamma, x : S)$ and $(\Gamma, x : S) - \Gamma'$ does not contain assumptions graded by `Public`.
  By the induction hypothesis, we have $\Gamma' \vdash [x \mapsto s]t_1 : T$, then apply the T-WEAK, we have $\Gamma, x : S \vdash [x \mapsto s]t_1 : T$.
- *Case* T-DER:
- *Case* T-PRO:
- *Case* T-LET:
- *Case* T-APPROX:

□

Lemma 3 (Well-typed Graded Substitution)**.** *If $\Gamma, x : [S]_\ell \vdash t : T$ and $[\Delta] \vdash s : S$, then $\Gamma, \ell \cdot \Delta \vdash [x \mapsto s]t : T$.*

Proof. □

B. *Type Preservation*

Theorem 1 (Type Preservation)**.**

C. Progression

D. Strong Normalization

E. Noninterference

F. Type Checking Algorithm

IV. Implementation and Examples

V. Related Work

VI. Further Work and Conclusion

Acknowledgment

References

Appendix

A. Complete Specification of FlowSTLC

1) Syntax:

Term	t	::=	x	*variable*
		|	$t\ t$	*application*
		|	$\lambda x.t$	*abstraction*
		|	$[t]$	*packing*
		|	$\textbf{let}\ [x] = t \colon T\ \textbf{in}\ t$	*unpacking*
Type	T	::=	$T \to T$	*function type*
		|	$\Box_r T$	*graded modality*
Value	v	::=	$\lambda x \colon T.t$	*abstraction value*
Context	Γ	::=	$\emptyset$	*empty context*
		|	$\Gamma, x \colon T$	*assumption*
		|	$\Gamma, x \colon [T]_r$	*graded assumption*
Security	ℓ	::=	`Secret`	
		|	`Public`	

2) Security Level Semiring:

Definition 1 (Security Level Semiring)**.** *The security level semiring is a two-point lattice of security levels* $\{\texttt{Public} \sqsubseteq \texttt{Secret}\}$ *with*

- $0 = \texttt{Secret}$
- $1 = \texttt{Public}$
- *Addition as the meet:* $r + s = r \sqcap s$
- *Multiplication as the join:* $r \cdot s = r \sqcup s$

See Appendix B for a proof that this algebra is a indeed a semiring.

3) Auxiliary Definitions:

Definition 2 (Context Concatenation)**.**

$$
\begin{aligned}
\emptyset + \Gamma &= \Gamma \\
\Gamma + \emptyset &= \Gamma \\
(\Gamma, x : T) + \Gamma' &= (\Gamma + \Gamma'), x : T \textit{ iff } x \notin \mathit{dom}(\Gamma') \\
\Gamma + (\Gamma', x : T) &= (\Gamma + \Gamma'), x : T \textit{ iff } x \notin \mathit{dom}(\Gamma') \\
(\Gamma, x : [T]_r) + (\Gamma', x : [T]_s) &= (\Gamma + \Gamma'), x : [T]_{r+s}
\end{aligned}
$$

Definition 3 (Context Scalar Multiplication)**.**

$$
\begin{aligned}
r \cdot \emptyset &= \emptyset \\
r \cdot (\Gamma, x : [T]_s) &= (r \cdot \Gamma), x : [T]_{(r \cdot s)}
\end{aligned}
$$

4) Typing Rules:

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T}\ \textsc{T-Var}$$

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma \vdash \lambda x : T_1.t : T_1 \to T_2}\ \textsc{T-Abs}$$

$$\frac{\Gamma_1 \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma_2 \vdash t_2 : T_{11}}{\Gamma_1 + \Gamma_2 \vdash t_1\ t_2 : T_{12}}\ \textsc{T-App}$$

$$\frac{\Gamma \vdash t : T}{\Gamma, \Gamma' \vdash t : T}\ \textsc{T-Weak}$$

where Γ' denotes a context containing only regular assumptions and assumptions graded by `Public`.

$$\frac{\Gamma, x : T_1 \vdash t : T_2}{\Gamma, x : [T_1]_{\texttt{Public}} \vdash t : T_2}\ \textsc{T-Der}$$

$$\frac{[\Gamma] \vdash t : T}{\ell \cdot [\Gamma] \vdash [t] : \Box_\ell T}\ \textsc{T-Pro}$$

$$\frac{\Gamma_1 \vdash t_1 : \Box_\ell T_1 \qquad \Gamma_2, x : [T_1]_\ell \vdash t_2 : T_2}{\Gamma_1 + \Gamma_2 \vdash \textbf{let}\ [x] = t_1\ \textbf{in}\ t_2 : T_2}\ \textsc{T-Let}$$

$$\frac{\Gamma, x : [T]_{\ell_1}, \Gamma' \vdash t : T \qquad \ell_1 \sqsubseteq \ell_2}{\Gamma, x : [T]_{\ell_2}, \Gamma' \vdash t : T}\ \textsc{T-Approx}$$

5) Evaluation Rules:

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2}\ \textsc{E-App1}$$

$$\frac{t_2 \to t_2'}{v_1\ t_2 \to v_1\ t_2'}\ \textsc{E-App2}$$

$$(\lambda x : T.t)\ v \to [x \mapsto v]t \quad \textsc{E-AppAbs}$$

$$\frac{t \to t'}{[t] \to [t']}\ \textsc{E-Pro}$$

$$\frac{t_1 \to t_1'}{\textbf{let}\ [x] = t_1\ \textbf{in}\ t_2 \to \textbf{let}\ [x] = t_1'\ \textbf{in}\ t_2}\ \textsc{E-Let-Eval}$$

$$\textbf{let}\ [x] = v\ \textbf{in}\ t \to [x \mapsto v]t \quad \textsc{E-Let-Unbox}$$

B. Proofs

1) Security Level Semiring:

Proof.

- **Associativity of addition**: $(a + b) + c = a + (b + c)$
  This is trivial since the semiring addition is meet, and meet is associative in a lattice.
- **Commutativity of addition**: $a + b = b + a$
  This is also trivial since meet is commutative in a lattice.
- **Additive identity**: $a + 0 = a$ for all a
  Since $0 = \texttt{Secret}$ and $\texttt{Public} \sqsubseteq \texttt{Secret}$, we have

$$
\begin{aligned}
\texttt{Public} \sqcap \texttt{Secret} &= \texttt{Public} \\
\texttt{Secret} \sqcap \texttt{Secret} &= \texttt{Secret}
\end{aligned}
\tag{1}
$$

- **Associativity of multiplication**: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

This is trivial since the semiring multiplication is join, and join is associative in a lattice.

- **Multiplication distributes over addition**: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
  Since the lattice is a chain (and thus distributive), this holds.
- **Multiplicative identity**: $1 \cdot a = a \cdot 1 = a$ for all a
  Since $1 = \texttt{Public}$ and $\texttt{Public} = \texttt{Secret}$, we have

$$\begin{aligned} \texttt{Public} \sqcup \texttt{Public} &= \texttt{Public} \\ \texttt{Public} \sqcup \texttt{Secret} &= \texttt{Secret} \\ \texttt{Secret} \sqcup \texttt{Public} &= \texttt{Secret} \end{aligned} \tag{2}$$

- **Multiplication by 0 annihilates**: $0 \cdot a = a \cdot 0 = 0$ for all a
  Since $0 = \texttt{Secret}$ and $\texttt{Public} \sqsubseteq \texttt{Secret}$, we have

$$\begin{aligned} \texttt{Secret} \sqcup \texttt{Public} &= \texttt{Secret} \\ \texttt{Secret} \sqcup \texttt{Secret} &= \texttt{Secret} \\ \texttt{Public} \sqcup \texttt{Secret} &= \texttt{Secret} \end{aligned} \tag{3}$$

Thus the security level algebra is a semiring. $\square$