

A Monadic Analysis of Information Flow Security with Mutable State

Karl Crary, Aleksey Klieger, and Frank Pfenning

October 24, 2007

Presentation by Neel Krishnaswami <neelk@cs.cmu.edu>

Overview

- ▶ Programming language, not authorization logic

Overview

- ▶ Programming language, not authorization logic
- ▶ Information flow language (*a la* SLam)

Overview

- ▶ Programming language, not authorization logic
- ▶ Information flow language (*a la* SLam)
- ▶ ...except:

Overview

- ▶ Programming language, not authorization logic
- ▶ Information flow language (*a la* SLam)
- ▶ ...except:
 - ▶ store-oriented, not value-oriented

Overview

- ▶ Programming language, not authorization logic
- ▶ Information flow language (*a la* SLam)
- ▶ ...except:
 - ▶ store-oriented, not value-oriented
 - ▶ authorizations via indexed monads

Overview

- ▶ Programming language, not authorization logic
- ▶ Information flow language (*a la* SLam)
- ▶ ...except:
 - ▶ store-oriented, not value-oriented
 - ▶ authorizations via indexed monads
 - ▶ comparable to DCC in style, only cleaner

Types and Security Levels

Basic types $A ::= 1 \mid \text{bool} \mid A \rightarrow B$
Pointer type $\mid \text{ref}_a A \mid \text{refr}_a A \mid \text{refw}_a A$
Monad type $\mid \bigcirc_o A$

$(\mathcal{L}, \sqsubseteq, \top, \perp, \sqcup)$ lattice $a, r, w \in \mathcal{L}$

read and write levels $o ::= (r, w)$

$(r, w) \preceq (r', w')$ if and only if $r \sqsubseteq r'$ and $w' \sqsubseteq w$

Terms and Expressions

Terms $M ::= () \mid \text{true} \mid \text{false} \mid \text{if}(M, M, M)$
| $x \mid M M \mid \lambda x : A. M$
| $I \mid \text{val } E$

Expressions $E ::= [M] \mid \text{let val } x = M \text{ in } E$
| $\text{ref}_a(M : A) \mid !M \mid M := M$

Terms and Expressions

Terms $M ::= () \mid \text{true} \mid \text{false} \mid \text{if}(M, M, M)$
 $\mid x \mid M M \mid \lambda x : A. M$
 $\mid l \mid \text{val } E$

Expressions $E ::= [M] \mid \text{let val } x = M \text{ in } E$
 $\mid \text{ref}_a(M : A) \mid !M \mid M := M$

- ▶ Terms are pure and effect-free

Terms and Expressions

Terms $M ::= () \mid \text{true} \mid \text{false} \mid \text{if}(M, M, M)$
 $\mid x \mid M M \mid \lambda x : A. M$
 $\mid l \mid \text{val } E$

Expressions $E ::= [M] \mid \text{let val } x = M \text{ in } E$
 $\mid \text{ref}_a(M : A) \mid !M \mid M := M$

- ▶ Terms are pure and effect-free
- ▶ Expressions may have heap effects

Terms and Expressions

Terms $M ::= () \mid \text{true} \mid \text{false} \mid \text{if}(M, M, M)$
 $\mid x \mid M M \mid \lambda x : A. M$
 $\mid l \mid \text{val } E$

Expressions $E ::= [M] \mid \text{let val } x = M \text{ in } E$
 $\mid \text{ref}_a(M : A) \mid !M \mid M := M$

- ▶ Terms are pure and effect-free
- ▶ Expressions may have heap effects
- ▶ Pfenning-Davies style language

Judgements

Judgements

▶ $\Sigma; \Gamma \vdash M : A$

Typing for pure terms – no information flow annotations

Judgements

- ▶ $\Sigma; \Gamma \vdash M : A$
Typing for pure terms – no information flow annotations
- ▶ $\Sigma; \Gamma \vdash E \dot{\div}_{(r,w)} A$
Typing for expressions – indexed by read and write levels

Judgements

- ▶ $\Sigma; \Gamma \vdash M : A$
Typing for pure terms – no information flow annotations
- ▶ $\Sigma; \Gamma \vdash E \dot{\div}_{(r,w)} A$
Typing for expressions – indexed by read and write levels
- ▶ $A \leq B$
Subtyping judgement

Judgements

- ▶ $\Sigma; \Gamma \vdash M : A$
Typing for pure terms – no information flow annotations
- ▶ $\Sigma; \Gamma \vdash E \dot{\div}_{(r,w)} A$
Typing for expressions – indexed by read and write levels
- ▶ $A \leq B$
Subtyping judgement
- ▶ $A \nearrow a$
Informativeness judgement – similar to DCC's protectedness

Expression Typing

$$\frac{\Sigma; \Gamma \vdash M : A}{\Sigma; \Gamma \vdash [M] \dot{\div}_{(\perp, \top)} A}$$

$$\frac{\Sigma; \Gamma \vdash M : \bigcirc_o A \quad \Sigma; \Gamma, x : A \vdash E \dot{\div}_o B}{\Sigma; \Gamma \vdash \text{let val } x = M \text{ in } E \dot{\div}_o B}$$

$$\frac{\Sigma; \Gamma \vdash M : \text{refr}_a A}{\Sigma; \Gamma \vdash !M \dot{\div}_{(a, \top)} A}$$

$$\frac{\Sigma; \Gamma \vdash M : \text{refw}_a A \quad \Sigma; \Gamma \vdash N : A}{\Sigma; \Gamma \vdash M := N \dot{\div}_{(\perp, a)} \mathbf{1}}$$

$$\frac{\Sigma; \Gamma \vdash M : A}{\Sigma; \Gamma \vdash \text{ref}_a(M : A) \dot{\div}_{(\perp, \top)} \text{ref}_a A}$$

Expression Subsumption

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad o \preceq o'}{\Sigma; \Gamma \vdash E \div_{o'} A}$$

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad A \leq B}{\Sigma; \Gamma \vdash E \div_o B}$$

Expression Subsumption

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad o \preceq o'}{\Sigma; \Gamma \vdash E \div_{o'} A}$$

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad A \leq B}{\Sigma; \Gamma \vdash E \div_o B}$$

- ▶ Left rule: move expressions to more liberal security regime

Expression Subsumption

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad o \preceq o'}{\Sigma; \Gamma \vdash E \div_{o'} A}$$

$$\frac{\Sigma; \Gamma \vdash E \div_o A \quad A \leq B}{\Sigma; \Gamma \vdash E \div_o B}$$

- ▶ Left rule: move expressions to more liberal security regime
- ▶ Right rule: subtyping internalizes \preceq at types

Subtyping – $A \leq B$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refr}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refw}_b B \leq \text{refw}_a A}$$

$$\frac{A \leq B \quad o \preceq o'}{\bigcirc_o A \leq \bigcirc_{o'} B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_a A \leq \text{ref}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_b B \leq \text{ref}_a A}$$

Subtyping – $A \leq B$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refr}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refw}_b B \leq \text{refw}_a A}$$

$$\frac{A \leq B \quad o \preceq o'}{\bigcirc_o A \leq \bigcirc_{o'} B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_a A \leq \text{ref}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_b B \leq \text{ref}_a A}$$

- ▶ Subtyping follows security lattice

Subtyping – $A \leq B$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refr}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refw}_b B \leq \text{refw}_a A}$$

$$\frac{A \leq B \quad o \preceq o'}{\bigcirc_o A \leq \bigcirc_{o'} B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_b B \leq \text{refw}_a A}$$

- ▶ Subtyping follows security lattice
- ▶ $\text{ref}_a A$ has intersection type flavour – induces some more subtyping

Subtyping – $A \leq B$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refr}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{refw}_b B \leq \text{refw}_a A}$$

$$\frac{A \leq B \quad o \preceq o'}{\bigcirc_o A \leq \bigcirc_{o'} B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_a A \leq \text{refr}_b B}$$

$$\frac{A \leq B \quad a \sqsubseteq b}{\text{ref}_b B \leq \text{refw}_a A}$$

- ▶ Subtyping follows security lattice
- ▶ $\text{ref}_a A$ has intersection type flavour – induces some more subtyping
- ▶ This is a risky choice – causes separate informativeness judgement?

Summary and Next Steps

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?
 - ▶ Suppose $E \div (r, w)1$.

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?
 - ▶ Suppose $E \div (r, w)1$.
 - ▶ E reads data, but can only return ()

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?
 - ▶ Suppose $E \div (r, w)1$.
 - ▶ E reads data, but can only return $()$
 - ▶ $()$ is not *informative* about pointer reads at r

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?
 - ▶ Suppose $E \div (r, w)1$.
 - ▶ E reads data, but can only return $()$
 - ▶ $()$ is not *informative* about pointer reads at r
 - ▶ So it's safe to say $E \div (\perp, w)1$.

Summary and Next Steps

- ▶ So far, language strictly segregates security levels.
- ▶ Can we get still more expressiveness?
 - ▶ Suppose $E \div (r, w)1$.
 - ▶ E reads data, but can only return $()$
 - ▶ $()$ is not *informative* about pointer reads at r
 - ▶ So it's safe to say $E \div (\perp, w)1$.
- ▶ Generalize this idea to all types – $A \nearrow a$

$$\frac{\Sigma; \Gamma \vdash E \div_{(r,w)} A \quad A \nearrow r}{\Sigma; \Gamma \vdash E \div_{(\perp,w)} A}$$

Informativeness

$$\frac{}{A \nearrow \perp} \quad \frac{A \nearrow a \quad b \sqsubseteq a}{A \nearrow b} \quad \frac{A \nearrow a \quad A \nearrow b}{A \nearrow a \sqcup b} \quad \frac{}{1 \nearrow a}$$

(no rule for bool)

$$\frac{B \nearrow a}{A \rightarrow B \nearrow a} \quad \frac{A \nearrow a}{\bigcirc_{(r,w)} A \nearrow w \sqcap a}$$

$$\frac{}{\text{ref}_a A \nearrow a} \quad \frac{A \nearrow a}{\text{ref}_b A \nearrow a} \quad \frac{}{\text{ref}_r A \nearrow a} \quad \frac{A \nearrow a}{\text{ref}_b A \nearrow a}$$

$$\frac{}{\text{ref}_w A \nearrow a}$$

Metatheory

Besides type safety, show *noninterference*.

Informally:

- ▶ Fix security level $s \in \mathcal{L}$.
- ▶ *High security* is any level above s , *low security* below s .

Metatheory

Besides type safety, show *noninterference*.

Informally:

- ▶ Fix security level $s \in \mathcal{L}$.
- ▶ *High security* is any level above s , *low security* below s .
- ▶ Noninterference holds if low security sub-computations behave identically for all high security heaps

Note that the lack of pointer comparisons matters for the proof.
(Maybe ok to make comparison computational?)

Design Critique

Overall very clean. However:

Design Critique

Overall very clean. However:

- ▶ Problem:

- Q: Why is informativeness not part of the subtyping judgement?

Design Critique

Overall very clean. However:

- ▶ Problem:

- Q: Why is informativeness not part of the subtyping judgement?

- A: Problematic interaction with reference subtypes

Design Critique

Overall very clean. However:

- ▶ Problem:

- Q: Why is informativeness not part of the subtyping judgement?

- A: Problematic interaction with reference subtypes

- ▶ Is there an alternative decomposition?

Design Critique

Overall very clean. However:

- ▶ Problem:

- Q: Why is informativeness not part of the subtyping judgement?

- A: Problematic interaction with reference subtypes

- ▶ Is there an alternative decomposition?

- ▶ Split $\bigcirc_{(r,w)} A$ into $\square_r B$ and $\diamond_w C$?

Design Critique

Overall very clean. However:

- ▶ Problem:

- Q: Why is informativeness not part of the subtyping judgement?

- A: Problematic interaction with reference subtypes

- ▶ Is there an alternative decomposition?

- ▶ Split $\bigcirc_{(r,w)} A$ into $\square_r B$ and $\diamond_w C$?

- ▶ Can linearity + fraction permissions model allocation better?