A Monadic Analysis of Information Flow Security with Mutable State

Karl Crary, Aleksey Kliger, *Frank Pfenning*

Carnegie Mellon University

FCS Workshop

Turku, Finland, July 2004

The ConCert Project

- Certified distributed computation
- Technical basis
  - Typed assembly language (TAL, TALT)
  - Certifying compilation (TILT, PCC)
- Some technical challenges
  - Types for distributed computation
  - Resource bound certification
  - Architecture verification
  - *Information flow*

Information Flow in TAL

- Typed assembly language
  - Imperative
  - Functional
  - Sequentialized
- Abstract to high-level functional language
  - Capture analagous features
  - Easier to design, prove correct, understand
  - Future work: transfer to TAL

Language Overview

- Information flow only through store
- Effects encapsulated in monad
- Other computations and values remain pure
- Monad and locations indexed by security levels
- Subtyping to avoid security level coercions
- Allow upcalls via informativeness judgment

Outline

- Monadic encapsulation of effects
- Information flow and store
- Upcalls and informativeness
- Proof of non-interference
- Embedding value-oriented languages

Pure Functional Core

- Standard constructs

$$\text{Types} \quad A \quad ::= \quad \textsf{bool} \mid 1 \mid A \to B \mid \ldots$$

- Standard judgments
  - Typing $\Gamma \vdash M : A$
  - Value M *val* (write V for values)
  - Reduction $M \to M'$
- Call-by-value (could be by name or by need)
- Curry-Howard isomorphism (omit recursion)

Sample Rules: Functions

- Typing

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : A \to B} \to I \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B} \to E$$

- Evaluation

$$\frac{}{\lambda x{:}A.M\ \textit{val}} \qquad \frac{M \to M'}{M\ N \to M'\ N}$$

$$\frac{V\ \textit{val} \quad N \to N'}{V\ N \to V\ N'} \qquad \frac{V\ \textit{val}}{(\lambda x{:}A.M)\ V \to M[V/x]}$$

Monadic Encapsulation

- New type $\bigcirc A$ for effectful computations
- New syntactic category: expressions

$$\begin{array}{rrcl} \text{Terms} & M & ::= & \ldots \mid \mathsf{val}\ E \\ \text{Expressions} & E & ::= & \mathsf{let\ val}\ x = M\ \mathsf{in}\ E \mid M \end{array}$$

- Expressions include terms
- Sequencing of effects via let val
- Further expressions for specific monads

Lax Typing

- Lax typing $\Gamma \vdash E \div A$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M \div A}$$

$$\frac{\Gamma \vdash E \div A}{\Gamma \vdash \textsf{val}\ E : \bigcirc A}\ \bigcirc I \qquad \frac{\Gamma \vdash M : \bigcirc A \quad \Gamma, x{:}A \vdash E \div C}{\Gamma \vdash \textsf{let val}\ x = M\ \textsf{in}\ E \div C}\ \bigcirc E$$

- Restriction on elimination enforces sequencing
- Related to *lax logic* by Curry-Howard isomorphism

Operational Semantics

- Computation steps $(H, E) \longrightarrow (H', E')$ for store H

$$\frac{}{\text{val } E \textit{ val}} \qquad \frac{M \longrightarrow M'}{(H, M) \longrightarrow (H, M')}$$

$$\frac{M \longrightarrow M'}{(H, \text{let val } x = M \text{ in } F) \longrightarrow (H, \text{let val } x = M' \text{ in } F)}$$

$$\frac{(H, E) \longrightarrow (H', E')}{(H, \text{let val } x = \text{val } E \text{ in } F) \longrightarrow (H', \text{let val } x = \text{val } E' \text{ in } F)}$$

$$\frac{V \textit{ val}}{(H, \text{let val } x = \text{val } V \text{ in } F) \longrightarrow (H, F[V/x])}$$

Security Levels

- Fixed lattice $a \sqsubseteq b$
- Operations $\bot$, $\top$, $\sqcap$, $\sqcup$
- Store locations l have security level a, type A (write: l_a^A, omit when clear)
- Computation $E \div_{(r,w)} A$ has security levels
  - r: can read only at r or below
  - w: can write only at w or above
  - *operation level* $o = (r, w)$ for $r \sqsubseteq w$
- Terms $M : A$ have no effect, no security level

Stores

- Store locations l_a^A with intrinsic security level a
- Store locations are terms (no effect)
- Store locations are values

$$\frac{}{l_a^A\ \textit{val}}$$

- Stores uniquely bind locations to values

$$\text{Store}\quad H \quad ::= \quad \cdot \mid H, l_a^A \mapsto V$$

Allocation, Reading, Writing

- Assign most precise type; others by subtyping
- Write $E \div (r, w)\ A$ for readability
- Allocation neither reads nor writes

$$\frac{}{\Gamma \vdash l_a^A : \mathsf{ref}_a\ A} \qquad \frac{\Gamma \vdash M : A}{\Gamma \vdash \mathsf{ref}_a\ M \div (\bot, \top)\ \mathsf{ref}_a\ A}$$

- Reading and writing are *effects*

$$\frac{\Gamma \vdash M : \mathsf{ref}_a\ A}{\Gamma \vdash\ !M \div (a, \top)\ A} \qquad \frac{\Gamma \vdash M : \mathsf{ref}_a\ A \quad \Gamma \vdash N : A}{\Gamma \vdash M := N \div (\bot, a)\ 1}$$

Subtyping

- $A \leq B$ A is subtype of B
- $o \preceq p$ o is less strict than p
- Subsumption rules

$$\frac{\Gamma \vdash M : A \quad A \leq B}{\Gamma \vdash M : B}$$

$$\frac{\Gamma \vdash E \div_o A \quad o \preceq p}{\Gamma \vdash E \div_p A} \qquad \frac{\Gamma \vdash E \div_o A \quad A \leq B}{\Gamma \vdash E \div_o B}$$

Variance

- Recall $E \div (r, w)\ A$
  - reads only below r
  - writes only above w
- Co-variant in read, contra-variant in write

$$\frac{r \sqsubseteq r' \quad w' \sqsubseteq w}{(r, w) \preceq (r', w')} \qquad \frac{A \leq B \quad o \preceq p}{\bigcirc_o A \leq \bigcirc_p B}$$

- ref$_a$ A is non-variant (paper: refr$_r$ A and refw$_w$ A)
- Other subtyping standard

Operational Semantics Revisited

- Standard rules for reduction with store
- Example: allocation

$$\frac{M \longrightarrow M'}{(H, \mathsf{ref}_a\, M) \longrightarrow (H, \mathsf{ref}_a\, M')}$$

$$\frac{V \ \textit{val} \quad l_a \notin \mathsf{dom}(H)}{(H, \mathsf{ref}_a\, V) \longrightarrow ((H, l \mapsto V), l)}$$

Lax Typing Revisited

- Lax security typing $\Gamma \vdash E \div_o A$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M \div (\bot, \top)\ A}$$

$$\frac{\Gamma \vdash E \div_o A}{\Gamma \vdash \mathsf{val}\ E : \bigcirc_o A} \bigcirc I \qquad \frac{\Gamma \vdash M : \bigcirc_o A \quad \Gamma, x{:}A \vdash E \div_o C}{\Gamma \vdash \mathsf{let\ val}\ x = M\ \mathsf{in}\ E \div_o C} \bigcirc E$$

- $(\bot, \top)$ is minimal for $\preceq$

Upcalls

- Consider a call of E at high security from within F at low security

$$E \div (\top, \top)\ 1$$
$$z{:}1 \vdash F \div (\bot, \bot)\ 1$$
$$\mathsf{let\ val}\ z = \mathsf{val}\ E\ \mathsf{in}\ F \div (?, \bot)\ 1$$

- Current rules force $? = \top$
- Does E leak information?
- Depends of type of returned value (here, 1)

Informativeness

- $A \nearrow r$ A is informative only at r and above
- Use to demote reading level of expressions

$$\frac{\Gamma \vdash E \div (r, w)\ A \quad A \nearrow r}{\Gamma \vdash E \div (\bot, w)\ A}$$

- Some rules

$$\frac{}{1 \nearrow r} \qquad \frac{B \nearrow b}{A \rightarrow B \nearrow b}$$

Informativeness of Computations

- Storage locations

$$\frac{}{\mathsf{ref}_b\, A \nearrow b} \qquad \frac{A \nearrow a}{\mathsf{ref}_b\, A \nearrow a}$$

- Computations

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

General Information Laws

- Contra-variant in security level

$$\frac{}{A \nearrow \bot} \qquad \frac{A \nearrow a \quad b \sqsubseteq a}{A \nearrow b}$$

$$\frac{A \nearrow b \quad A \nearrow c}{A \nearrow b \sqcup c}$$

- Now can type *untilFalse* : $\bigcirc_{(\top,\top)}\text{bool} \longrightarrow \bigcirc_{(\bot,\top)}1$ [see paper]
- Do not consider termination channel

Theorems

- Write $\vdash H$ if store is well-typed
- Write $\vdash (H, E) \div_o A$ if $\vdash H$ and $\vdash E \div_o A$
- Language so far satisfies
  - *Preservation:* If $\vdash (H, E) \div_o A$, and $(H, E) \longrightarrow (H', E')$ then $\vdash (H', E') \div_o A$.
  - *Progress:* If $\vdash (H, E) \div_o A$ then either $E = V$ for V *val* or $(H, E) \longrightarrow (H', E')$ for some (H', E')
  - *Non-interference:* “Computations at low security cannot observe high-security values”

Sketch of Non-Interference

- Define *in-view locations* for level ζ:

$$\downarrow(\zeta) = \{l_a \mid a \sqsubseteq \zeta\}$$

- Define equivalence on in-view locations $H_1 \approx_\zeta H_2$ and $(H_1, E_1) \approx_\zeta (H_2, E_2) \div_o A$
- **Theorem:** If $\vdash H$ and $x{:}A \vdash E \div_{(r,w)} B$ and $V_1 \approx_r V_2 : A$ then if $(H, E[V_1/x]) \longrightarrow^* S_1$ and $(H, E[V_2/x]) \longrightarrow S_2$ then $S_1 \approx_r S_2 \div_{(r,w)} B$.
- **Proof:** Syntactic, using Church-Rosser modulo in-view equivalence with respect to r.

Related Work

- Information flow inference for ML [Pottier&Simonet'03]
  - Any term may have an effect
  - Emphasis on inference
  - Here: monadic encapsulation, checking
- Dependency Core Calculus (DCC) [Abadi,Banerjee,Heintze,Riecke'99]
  - Monads for sealing values, not state
  - Protectedness $\sim$ informativeness

Related Work

- $\lambda_{\mathrm{SEC}}^{\mathrm{REF}}$ [Zdancewic'02]
  - Security levels for values, not locations
  - Can be mapped to our language [see paper]
- Information flow for π-calculus [Honda&Yoshida'02]
  - Different computational setting
  - Tampering levels $\sim$ informativeness
- Domain separation [Harrison,Tullsen,Hook'03]
  - State insulation via monads
  - No interaction between monads

Future Work

- Additional effects (I/O, control effects)
- Information flow in TAL (register re-use)
- Decomposing the monad into $\Box$, $\Diamond$ [Pf.&Davies'01]
- Dependent type theory with information flow

Summary

- Type system for information flow
  - Higher-order functional language
  - Store monad, indexed by operation levels
  - Security levels for locations, not values
- Conservative over base language
- Upcalls permitted via informativeness
- Preservation, progress, non-interference