

Reasoning about the Consequences of Authorization Policies in a Linear Epistemic Logic

Henry DeYoung Frank Pfenning

Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213, USA

Manifest Security Meeting
May 20 and 21, 2009

1 Introduction

- 1 Introduction
- 2 Logic
 - Metatheory

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

Observation:

- Authorization policies are not stand-alone objects.
 - Permit actions that change a system's state.
 - Intended to allow only safe consequences.

Observation:

- Authorization policies are not stand-alone objects.
 - Permit actions that change a system's state.
 - Intended to allow only safe consequences.

Example:

Policy “A principal may read file F if F 's owner says so.”

Consequence “A principal may learn F 's contents if granted read access.”

Observation:

- Authorization policies are not stand-alone objects.
 - Permit actions that change a system's state.
 - Intended to allow only safe consequences.

Example:

Policy “A principal may read file F if F 's owner says so.”

Consequence “A principal may learn F 's contents if granted read access.”

Goal:

- Develop a general method for formally:
 - specifying both authorization policies and their semantic consequences; and
 - reasoning about the interface between them.

Our Proposed Method

- 1 Specify policies and semantics in a linear logic with security modalities.

Our Proposed Method

- 1 Specify policies and semantics in a linear logic with security modalities.
- 2 Define a notion of state for the system.

Our Proposed Method

- 1 Specify policies and semantics in a linear logic with security modalities.
- 2 Define a notion of state for the system.
- 3 Interpret semantic specifications as rewrite steps.

Our Proposed Method

- 1 Specify policies and semantics in a linear logic with security modalities.
- 2 Define a notion of state for the system.
- 3 Interpret semantic specifications as rewrite steps.
- 4 Using the rewrite steps, prove properties about the system.

Our Proposed Method: Specification

- 1 Specify the policies and semantics in a linear logic with security modalities.

Our Proposed Method: Specification

- 1 Specify the policies and semantics in a linear logic with security modalities.

Conceptually, policies and semantics should be stratified.

Our Proposed Method: Specification

- 1 Specify the policies and semantics in a linear logic with security modalities.

Conceptually, policies and semantics should be stratified.

Linearity:

- Models use-once authorizations and changes in system's state.

Our Proposed Method: Specification

- 1 Specify the policies and semantics in a linear logic with security modalities.

Conceptually, policies and semantics should be stratified.

Linearity:

- Models use-once authorizations and changes in system's state.

Security Modalities:

- Affirmation (for policies): K affirms A
- Knowledge (for semantics): K knows A
- Possession (for semantics): K has A

Our Proposed Method: Specification

- 1 Specify the policies and semantics in a linear logic with security modalities.

Conceptually, policies and semantics should be stratified.

Linearity:

- Models use-once authorizations and changes in system's state.

Security Modalities:

- Affirmation (for policies): K affirms A
- Knowledge (for semantics): K knows A
- Possession (for semantics): K has A

Modalities provide logical force; predicates do not.

Our Proposed Method: System State

- 2 Define a notion of state for the system.

Our Proposed Method: System State

- 2 Define a notion of state for the system.

Allow only predicates relevant to the domain.

- Money is important to a bookstore: allow K has *money*(\$5).
- Medical records are not: disallow $\langle \text{hosp} \rangle \text{may_read_rec}(D, R)$.

- 2 Define a notion of state for the system.

Allow only predicates relevant to the domain.

- Money is important to a bookstore: allow K has *money*(\$5).
- Medical records are not: disallow $\langle \text{hosp} \rangle \text{may_read_rec}(D, R)$.

Can also impose simple state invariants.

- “At most one *occupies_office*(K, O) assumption for each K .”

Our Proposed Method: System State

- 2 Define a notion of state for the system.

Allow only predicates relevant to the domain.

- Money is important to a bookstore: allow K has *money*(\$5).
- Medical records are not: disallow $\langle \text{hosp} \rangle \text{may_read_rec}(D, R)$.

Can also impose simple state invariants.

- “At most one *occupies_office*(K, O) assumption for each K .”

Provides leverage for listing possible rewrites in next step.

Our Proposed Method: Rewrite Steps

-
-
- 3 Interpret semantic specifications as rewrite steps.

Our Proposed Method: Rewrite Steps

- 3 Interpret semantic specifications as rewrite steps.

Use a rewriting interpretation of our security linear logic.

- **rewrite sequence = trace of semantic actions in system's evolution**
- System state circumscribes possible rewrite steps.

Our Proposed Method: Rewrite Steps

- 3 Interpret semantic specifications as rewrite steps.

Use a rewriting interpretation of our security linear logic.

- **rewrite sequence = trace of semantic actions in system's evolution**
- System state circumscribes possible rewrite steps.

Theorem (Rewrite Step Schema)

Each rewrite step from a system state is either: ...

Our Proposed Method: Rewrite Steps

- 3 Interpret semantic specifications as rewrite steps.

Use a rewriting interpretation of our security linear logic.

- **rewrite sequence = trace of semantic actions in system's evolution**
- System state circumscribes possible rewrite steps.

Theorem (Rewrite Step Schema)

Each rewrite step from a system state is either: ...

ω [CS09]: useful, but needs extension to our *security* linear logic.

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

Persistent Assumptions

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

Consumable Assumptions

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

- ω translates derivations to rewrite sequences.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow^* \Gamma'; \Delta'$$

Rewrite Steps in ω

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

- ω translates derivations to rewrite sequences.

$$\frac{\begin{array}{c} \Gamma'; \Delta' \vdash J \\ \vdots \\ \Gamma; \Delta \vdash J \end{array}}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow^* \Gamma'; \Delta'$$

- Semantic rules would correspond to rewrite *sequences*.
- Sequences might be interleaved or partially executed.
 - Resource deadlock is possible.

- ω translates sequent calculus left rules to rewrite steps.

$$\frac{\Gamma'; \Delta' \vdash J}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow \Gamma'; \Delta'$$

- ω translates derivations to rewrite sequences.

$$\frac{\begin{array}{c} \Gamma'; \Delta' \vdash J \\ \vdots \\ \Gamma; \Delta \vdash J \end{array}}{\Gamma; \Delta \vdash J} \rightsquigarrow \Gamma; \Delta \rightarrow^* \Gamma'; \Delta'$$

- Semantic rules would correspond to rewrite *sequences*.
- Sequences might be interleaved or partially executed.
 - Resource deadlock is possible.
- Rules must be atomic: we use Andreoli's focusing!

Special Case: \multimap_L Rule

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash J}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash J} \multimap_L$$

- Contains a minor premise: doesn't fit previous translation.
- ω inlines the rewrite sequence for the minor premise.

Special Case: \multimap_L Rule

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash J}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash J} \multimap_L$$

- Contains a **minor premise**: doesn't fit previous translation.
- ω inlines the rewrite sequence for the minor premise.

Special Case: \multimap_L Rule

$$\frac{\Gamma; \Delta_1 \vdash A \quad \Gamma; \Delta_2, B \vdash J}{\Gamma; \Delta_1, \Delta_2, A \multimap B \vdash J} \multimap_L$$

- Contains a minor premise: doesn't fit previous translation.
- ω inlines the rewrite sequence for the minor premise.
- Inlining will not work for our purpose. . .

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource.

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource.
- 2 To *receive* access, proofs must be submitted.

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. **Not effectful**
- 2 To *receive* access, proofs must be submitted.

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. Not effectful
- 2 To *receive* access, proofs must be submitted. Effectful

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. **Not effectful**
- 2 To *receive* access, proofs must be submitted. **Effectful**

Bob reasons:

- “fs says a principal may read a file, if the owner says so.”

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. **Not effectful**
- 2 To *receive* access, proofs must be submitted. **Effectful**

Bob reasons:

- “fs says a principal may read a file, if the owner says so.”
- “fs says Alice owns `alice.txt`.”

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. **Not effectful**
- 2 To *receive* access, proofs must be submitted. **Effectful**

Bob reasons:

- “fs says a principal may read a file, if the owner says so.”
- “fs says Alice owns `alice.txt`.”
- “Alice says I may read `alice.txt`.”

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. Not *effective*
- 2 To *receive* access, proofs must be submitted. *Effective*

Bob reasons:

- “fs says a principal may read a file, if the owner says so.”
- “fs says Alice owns `alice.txt`.”
- “Alice says I may read `alice.txt`.”
- “Therefore, fs says I may read `alice.txt`.”

Aside: Proof-Carrying Authorization

Proof-Carrying Authorization (PCA):

- 1 Proofs show *permission* to access a resource. **Not effectful**
- 2 To *receive* access, proofs must be submitted. **Effectful**

Bob reasons:

- “fs says a principal may read a file, if the owner says so.”
- “fs says Alice owns `alice.txt`.”
- “Alice says I may read `alice.txt`.”
- “Therefore, fs says I may read `alice.txt`.”

To read `alice.txt`, Bob must submit this proof to the file system.

Our Proposed Method: Rewrite Steps

We want “rewrite sequence = trace of semantic effects.”

Our Proposed Method: Rewrite Steps

We want “rewrite sequence = trace of semantic effects.”

- Proving permission *is not* effectful.
- Should not be in the rewrite sequence.

Our Proposed Method: Rewrite Steps

We want “rewrite sequence = trace of semantic effects.”

- Proving permission *is not* effectful.
- Should not be in the rewrite sequence.

Cannot inline minor premises—must allow branched derivations.

Our Proposed Method: Rewrite Steps

We want “rewrite sequence = trace of semantic effects.”

- Proving permission *is not* effectful.
- Should not be in the rewrite sequence.

Cannot inline minor premises—must allow branched derivations.

Problem:

- Now, semantic effects can appear in branches.
- Invisible to the rewrite sequence derived from the main branch.

Our Proposed Method: Rewrite Steps

We want “rewrite sequence = trace of semantic effects.”

- Proving permission *is not* effectful.
- Should not be in the rewrite sequence.

Cannot inline minor premises—must allow branched derivations.

Problem:

- Now, semantic effects can appear in branches.
- Invisible to the rewrite sequence derived from the main branch.

Our Solution:

- Use a monad to identify the main branch.
- In their specification, confine semantic effects to the monad.

Our Proposed Method: Proving Properties

- 4 Using the rewrite steps, prove properties about the system.

Our Proposed Method: Proving Properties

- 4 Using the rewrite steps, prove properties about the system.

Previous step yields:

Theorem (Rewrite Step Schema)

Each rewrite step from a system state is either: ...

Our Proposed Method: Proving Properties

- 4 Using the rewrite steps, prove properties about the system.

Previous step yields:

Theorem (Rewrite Step Schema)

Each rewrite step from a system state is either: ...

- Phrase properties in terms of these rewrite step schema.
- Prove them by analyzing rewrite steps.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau.A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau.A^- \mid \{A^+\} \end{aligned}$$

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau.A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau.A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$
Right Focusing	$\Gamma; \Delta \vdash [A^+]$
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau.A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau.A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$	Invertible Rules
Right Focusing	$\Gamma; \Delta \vdash [A^+]$	
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$	

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$	
Right Focusing	$\Gamma; \Delta \vdash [A^+]$	Noninvertible Right Rules
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$	

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$	
Right Focusing	$\Gamma; \Delta \vdash [A^+]$	
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$	Noninvertible Left Rules

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$
Right Focusing	$\Gamma; \Delta \vdash [A^+]$
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$

Unrestricted Hyps	$\Gamma ::= \cdot \mid \Gamma, A^- \text{ valid}$
Linear Hyps	$\Delta ::= \cdot \mid \Delta, A^+ \text{ hyp}$
Consequents	$J ::= A^- \text{ true} \mid A^+ \text{ lax}$

Weakly Focused Linear Logic

Polarized propositions:

$$\begin{aligned} A^+, B^+ &= p^+ \mid A^+ \otimes B^+ \mid \exists x:\tau. A^+ \mid \mathbf{1} \mid !A^- \mid A^- \\ A^-, B^- &= p^- \mid A^+ \multimap B^- \mid \forall x:\tau. A^- \mid \{A^+\} \end{aligned}$$

Three forms of sequent:

Neutral	$\Gamma; \Delta \vdash J$
Right Focusing	$\Gamma; \Delta \vdash [A^+]$
Left Focusing	$\Gamma; \Delta, [A^-] \vdash J$

Unrestricted Hyps	$\Gamma ::= \cdot \mid \Gamma, A^- \text{ valid}$
Linear Hyps	$\Delta ::= \cdot \mid \Delta, A^+ \text{ hyp}$
Consequents	$J ::= A^- \text{ true} \mid A^+ \text{ lax}$

Selected Rules:

$$\overline{\Gamma; [p^-] \vdash p^-} \text{ atom}^-$$

Selected Rules:

$$\overline{\Gamma; [p^-] \vdash p^-} \text{ atom}^-$$

$$\frac{\Gamma; \Delta_1 \vdash [A^+] \quad \Gamma; \Delta_2, [B^-] \vdash J}{\Gamma; \Delta_1, \Delta_2, [A^+ \multimap B^-] \vdash J} \multimap_L$$

Weakly Focused Linear Logic

Selected Rules:

$$\overline{\Gamma; [p^-] \vdash p^-} \text{ atom}^-$$

$$\frac{\Gamma; \Delta_1 \vdash [A^+] \quad \Gamma; \Delta_2, [B^-] \vdash J}{\Gamma; \Delta_1, \Delta_2, [A^+ \multimap B^-] \vdash J} \multimap_L$$

$$\frac{\Gamma; \Delta, A^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta, [\{A^+\}] \vdash C^+ \text{ lax}} \{\}_L$$

Adapt affirmation from [Garg+06] to weak focusing.

- Family of lax judgments indexed by principals:

Consequents $J ::= A^- \text{ true} \mid A^+ \text{ lax} \mid K \text{ affirms } A^+$

Affirmation

- All principals affirm true statements.

$$\frac{\Gamma; \Delta \vdash [A^+]}{\Gamma; \Delta \vdash K \text{ affirms } A^+} \text{affirms}_R$$

Affirmation

- All principals affirm true statements.

$$\frac{\Gamma; \Delta \vdash [A^+]}{\Gamma; \Delta \vdash K \text{ affirms } A^+} \text{ affirms}_R$$

- Internalize $K \text{ affirms } A^+$ as the proposition $\langle K \rangle A^+$, read “ K says A^+ .”

$$\frac{\Gamma; \Delta \vdash K \text{ affirms } A^+}{\Gamma; \Delta \vdash \langle K \rangle A^+} \langle \rangle_R$$

Affirmation

- All principals affirm true statements.

$$\frac{\Gamma; \Delta \vdash [A^+]}{\Gamma; \Delta \vdash K \text{ affirms } A^+} \text{ affirms}_R$$

- Internalize $K \text{ affirms } A^+$ as the proposition $\langle K \rangle A^+$, read “ K says A^+ .”

$$\frac{\Gamma; \Delta \vdash K \text{ affirms } A^+}{\Gamma; \Delta \vdash \langle K \rangle A^+} \langle \rangle_R$$

- Principals trust their own affirmations to be true.

$$\frac{\Gamma; \Delta, A^+ \vdash K \text{ affirms } C^+}{\Gamma; \Delta, [\langle K \rangle A^+] \vdash K \text{ affirms } C^+} \langle \rangle_L$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle fs \rangle (owns^-(K, F) \multimap may^-(K, read(F)))$$

- A principal may read a file if the owner says so.

$$\langle fs \rangle (owns^-(K, F) \otimes \langle K \rangle may^-(L, read(F)) \multimap may^-(L, read(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

Affirmation: Examples

- File owners may read their files.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, \text{read}(F)))$$

- A principal may read a file if the owner says so.

$$\langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, \text{read}(F)) \multimap \text{may}^-(L, \text{read}(F)))$$

- Policy rules begin with a $\langle - \rangle$ modality.

Adapt knowledge from [Garg+06] to weak focusing.

- Family of S4 judgments indexed by principals:

Unrestricted Hyps $\Gamma ::= \cdot \mid \Gamma, A^- \text{ valid} \mid \Gamma, K \text{ knows } A^-$

- If K knows (cf. believes) a fact, that fact will be provably true.

$$\frac{\Gamma, K \text{ knows } A^-; \Delta, [A^-] \vdash J}{\Gamma, K \text{ knows } A^-; \Delta \vdash J} \text{ knows}_L$$

- If K knows (cf. believes) a fact, that fact will be provably true.

$$\frac{\Gamma, K \text{ knows } A^-; \Delta, [A^-] \vdash J}{\Gamma, K \text{ knows } A^-; \Delta \vdash J} \text{ knows}_L$$

- Internalize $K \text{ knows } A^-$ as the proposition $\llbracket K \rrbracket A^-$.

$$\frac{\Gamma, K \text{ knows } A^-; \Delta \vdash J}{\Gamma; \Delta, \llbracket K \rrbracket A^- \vdash J} \llbracket \rrbracket_L$$

- Knowledge should be:

$$\frac{\Gamma|_K; \cdot \vdash A^-}{\Gamma; \cdot \vdash \llbracket K \rrbracket A^-} \quad \square_R$$

- Knowledge should be:
 Reusable: empty linear context

$$\frac{\Gamma|_K; \cdot \vdash A^-}{\Gamma; \cdot \vdash \llbracket K \rrbracket A^-} \text{R}$$

- Knowledge should be:

Reusable: empty linear context

Private: restriction operator $|_K$

$$\frac{\Gamma|_K; \cdot \vdash A^-}{\Gamma; \cdot \vdash \llbracket K \rrbracket A^-} \text{R}$$

$$\begin{aligned}(\cdot)|_K &= \cdot \\(\Gamma, A^- \text{ valid})|_K &= \Gamma|_K \\(\Gamma, K \text{ knows } A^-)|_K &= \Gamma|_K, K \text{ knows } A^- \\(\Gamma, L \text{ knows } A^-)|_K &= \Gamma|_K \text{ if } L \neq K\end{aligned}$$

- If granted read access to file F ,
a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- If granted read access to file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- If granted read access to file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- If granted read access to file F ,
a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &\text{[[fs]]} contents^-(F, S) \multimap \\ &\quad \{ \text{[[K]]} contents^-(F, S) \} \end{aligned}$$

- If granted read access to file F ,
a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- If granted read access to file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- Learning contents is an effect—confined to the monad.

- If granted read access to file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- Learning contents is an effect—confined to the monad.
- Semantic rules have monadic heads, $\{-\}$.
- Policy rules start with $\langle - \rangle$ and do not contain $\{-\}$.

- If granted read access to file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \} \end{aligned}$$

- Learning contents is an effect—confined to the monad.
- Semantic rules have monadic heads, $\{-\}$.
- Policy rules start with $\langle - \rangle$ and do not contain $\{-\}$.
- Policy and semantic rules are distinguishable syntactically.
 - Evidence of stratification.

Adapt possession from [Garg+06] to weak focusing.

- Possession is linear knowledge.
- Family of S4 judgments indexed by principals:

Linear Hyps $\Delta ::= \cdot \mid \Delta, A^+ \text{ hyp} \mid \Delta, K \text{ has } A^-$

$$\frac{\Gamma; \Delta, [A^-] \vdash J}{\Gamma; \Delta, K \text{ has } A^- \vdash J} \text{has}_L$$

$$\frac{\Gamma; \Delta, [A^-] \vdash J}{\Gamma; \Delta, K \text{ has } A^- \vdash J} \text{has}_L$$

$$\frac{\Gamma; \Delta, K \text{ has } A^- \vdash J}{\Gamma; \Delta, [K]A^- \vdash J} \boxed{L}$$

$$\frac{\Gamma; \Delta, [A^-] \vdash J}{\Gamma; \Delta, K \text{ has } A^- \vdash J} \text{has}_L$$

$$\frac{\Gamma; \Delta, K \text{ has } A^- \vdash J}{\Gamma; \Delta, [K]A^- \vdash J} \boxed{L}$$

$$\frac{\Gamma|_K; \Delta|_K \vdash A^-}{\Gamma; \Delta|_K \vdash [[K]A^-]} \boxed{R}$$

$$\begin{aligned} (\cdot)|_K &= \cdot \\ (\Delta, A^+ \text{ hyp})|_K &= \Delta|_K \\ (\Delta, K \text{ has } A^-)|_K &= \Delta|_K, K \text{ has } A^- \\ (\Delta, L \text{ has } A^-)|_K &= \Delta|_K \text{ if } L \neq K \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} & \langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ & \llbracket fs \rrbracket contents^-(F, S) \otimes [fs] available(F) \multimap \\ & \{ \llbracket K \rrbracket contents^-(F, S) \otimes \\ & \quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} & \langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ & \llbracket fs \rrbracket contents^-(F, S) \otimes [fs]available(F) \multimap \\ & \{ \llbracket K \rrbracket contents^-(F, S) \otimes \\ & \quad [fs]available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} &\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ &[fs] available(F) \multimap \\ &\quad \{ \mathbf{1} \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\{ [[K]] contents^-(F, S) \otimes \\ &[fs] available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} &\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ &[fs] available(F) \multimap \\ &\{ \mathbf{1} \} \end{aligned}$$

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} & \langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ & \llbracket fs \rrbracket contents^-(F, S) \otimes [fs] available(F) \multimap \\ & \{ \llbracket K \rrbracket contents^-(F, S) \otimes \\ & \quad [fs] available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} & \langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ & [fs] available(F) \multimap \\ & \{ \mathbf{1} \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} & \langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ & \llbracket fs \rrbracket contents^-(F, S) \otimes [fs]available(F) \multimap \\ & \{ \llbracket K \rrbracket contents^-(F, S) \otimes \\ & \quad [fs]available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} & \langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ & [fs]available(F) \multimap \\ & \{ \mathbf{1} \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\{ [[K]] contents^-(F, S) \otimes \\ &[fs] available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} &\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ &[fs] available(F) \multimap \\ &\{ \mathbf{1} \} \end{aligned}$$

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} &\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ &[[fs]] contents^-(F, S) \otimes [fs] available(F) \multimap \\ &\quad \{ [[K]] contents^-(F, S) \otimes \\ &\quad [fs] available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} &\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ &[fs] available(F) \multimap \\ &\quad \{ \mathbf{1} \} \end{aligned}$$

Possession: Examples

- If granted read access to an available file F , a principal may learn F 's contents.

$$\begin{aligned} & \langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes \\ & \llbracket fs \rrbracket contents^-(F, S) \otimes \llbracket fs \rrbracket available(F) \multimap \\ & \{ \llbracket K \rrbracket contents^-(F, S) \otimes \\ & \llbracket fs \rrbracket available(F) \} \end{aligned}$$

- If granted delete rights to an available file F , a principal may make F unavailable.

$$\begin{aligned} & \langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes \\ & \llbracket fs \rrbracket available(F) \multimap \\ & \{ \mathbf{1} \} \end{aligned}$$

- Linear nature of possession is essential for deletes!
- $available(F)$ state is localized to fs , not global.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

Theorem (Admissibility of Cut)

- 1 If $\Gamma; \Delta \vdash A^-$ and $\Gamma; \Delta', [A^-] \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 2 If $\Gamma; \Delta \vdash [A^+]$ and $\Gamma; \Delta', A^+ \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.

Theorem (Admissibility of Cut)

- 1 If $\Gamma; \Delta \vdash A^-$ and $\Gamma; \Delta', [A^-] \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 2 If $\Gamma; \Delta \vdash [A^+]$ and $\Gamma; \Delta', A^+ \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.

Theorem (Identity)

- 1 For all Γ and A^- , $\Gamma; A^- \vdash A^-$.
- 2 For all Γ and A^+ , $\Gamma; A^+ \vdash A^+ \text{ lax}$.

Theorem (Admissibility of Cut)

- 1 If $\Gamma; \Delta \vdash A^-$ and $\Gamma; \Delta', [A^-] \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 2 If $\Gamma; \Delta \vdash [A^+]$ and $\Gamma; \Delta', A^+ \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.

Theorem (Identity)

- 1 For all Γ and A^- , $\Gamma; A^- \vdash A^-$.
- 2 For all Γ and A^+ , $\Gamma; A^+ \vdash A^+$ lax.

- Ensure that our sequent forms respect the defining properties of a hypothetical judgment.
- Can be useful in reasoning about policies and their semantics.

Theorem (Admissibility of Cut)

- 1 If $\Gamma; \Delta \vdash A^-$ and $\Gamma; \Delta', [A^-] \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 2 If $\Gamma; \Delta \vdash [A^+]$ and $\Gamma; \Delta', A^+ \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 3 If $\Gamma; \cdot \vdash A^-$ and $\Gamma, A^-; \Delta' \vdash J$, then $\Gamma; \Delta' \vdash J$.
- 4 If $\Gamma; \Delta \vdash A^+$ lax and $\Gamma; \Delta', A^+ \vdash C^+$ lax, then $\Gamma; \Delta', \Delta \vdash C^+$ lax.
- 5 If $\Gamma; \Delta \vdash K$ affirms A^+ and $\Gamma; \Delta', A^+ \vdash K$ affirms C^+ , then $\Gamma; \Delta', \Delta \vdash K$ affirms C^+ .
- 6 If $\Gamma|_K; \cdot \vdash A^-$ and Γ, K knows $A^-; \Delta' \vdash J$, then $\Gamma; \Delta' \vdash J$.
- 7 If $\Gamma|_K; \Delta|_K \vdash A^-$ and $\Gamma; \Delta', K$ has $A^- \vdash J$, then $\Gamma; \Delta', \Delta|_K \vdash J$.

Theorem (Admissibility of Cut)

- 1 If $\Gamma; \Delta \vdash A^-$ and $\Gamma; \Delta', [A^-] \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 2 If $\Gamma; \Delta \vdash [A^+]$ and $\Gamma; \Delta', A^+ \vdash J$, then $\Gamma; \Delta', \Delta \vdash J$.
- 3 If $\Gamma; \cdot \vdash A^-$ and $\Gamma, A^-; \Delta' \vdash J$, then $\Gamma; \Delta' \vdash J$.
- 4 If $\Gamma; \Delta \vdash A^+$ lax and $\Gamma; \Delta', A^+ \vdash C^+$ lax, then $\Gamma; \Delta', \Delta \vdash C^+$ lax.
- 5 If $\Gamma; \Delta \vdash K$ affirms A^+ and $\Gamma; \Delta', A^+ \vdash K$ affirms C^+ , then $\Gamma; \Delta', \Delta \vdash K$ affirms C^+ .
- 6 If $\Gamma|_K; \cdot \vdash A^-$ and Γ, K knows $A^-; \Delta' \vdash J$, then $\Gamma; \Delta' \vdash J$.
- 7 If $\Gamma|_K; \Delta|_K \vdash A^-$ and $\Gamma; \Delta', K$ has $A^- \vdash J$, then $\Gamma; \Delta', \Delta|_K \vdash J$.

- Ensures that our sequent forms respect the substitution property of a hypothetical judgment.

Theorem (η Expansion)

- 1 *If, by assuming a derivation of $\Gamma, \Gamma'; \Delta' \vdash [A^+]$, one can derive $\Gamma, \Gamma'; \Delta, \Delta' \vdash J$ parametrically in Γ' and Δ' , then $\Gamma; \Delta, A^+ \vdash J$.*
- 2 *If, by assuming a derivation of $\Gamma, \Gamma'; \Delta', [A^-] \vdash J$, one can derive $\Gamma, \Gamma'; \Delta', \Delta \vdash J$ parametrically in Γ' , Δ' , and J , then $\Gamma; \Delta \vdash A^-$.*

Theorem (η Expansion)

- 1** *If, by assuming a derivation of $\Gamma, \Gamma'; \Delta' \vdash [A^+]$, one can derive $\Gamma, \Gamma'; \Delta, \Delta' \vdash J$ parametrically in Γ' and Δ' , then $\Gamma; \Delta, A^+ \vdash J$.*
 - 2** *If, by assuming a derivation of $\Gamma, \Gamma'; \Delta', [A^-] \vdash J$, one can derive $\Gamma, \Gamma'; \Delta', \Delta \vdash J$ parametrically in Γ' , Δ' , and J , then $\Gamma; \Delta \vdash A^-$.*
- Ensures that our sequent forms respect the identity property of a hypothetical judgment.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

$\Gamma; \Delta$ is a generic state, if Δ is fully inverted.

Definition (Generic State)

A pair of contexts $\Gamma; \Delta$ is a *generic state* if and only if Δ fits the grammar:

$$\Delta ::= \cdot \mid \Delta, A^- \text{ hyp} \mid \Delta, p^+ \text{ hyp} \mid \Delta, K \text{ has } A^-$$

Definition (Step)

$\Gamma; \Delta \rightarrow \Gamma'; \Delta'$ if and only if there exists a derivation

$$\frac{\frac{\Gamma'; \Delta' \vdash C^+ \text{ lax}}{\vdots} \quad \frac{\Gamma; \Delta_2, A_2^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta_2, [\{A_2^+\}] \vdash C^+ \text{ lax}} \left. \vphantom{\frac{\Gamma; \Delta_2, A_2^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta_2, [\{A_2^+\}] \vdash C^+ \text{ lax}}} \right\} \!_L}{\frac{\Gamma; \Delta_1, [A_1^-] \vdash C^+ \text{ lax}}{\vdots} \quad \frac{\Gamma; \Delta_1, [A_1^-] \vdash C^+ \text{ lax}}{\Gamma; \Delta \vdash C^+ \text{ lax}} \!_*$$

parametrically in C^+ , where only invertible left rules are used above $\left. \vphantom{\frac{\Gamma; \Delta_2, A_2^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta_2, [\{A_2^+\}] \vdash C^+ \text{ lax}}} \right\} \!_L$ and both $\Gamma; \Delta$ and $\Gamma'; \Delta'$ are states.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 **File System Example**
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

File System Example

Our file system has insert, read, write, and delete actions.

Actions $A(\dots) ::= \textit{insert}() \mid \textit{read}(F) \mid \textit{write}(F, S) \mid \textit{delete}(F)$

File System Example

Our file system has insert, read, write, and delete actions.

Actions $A(\dots) ::= \text{insert}() \mid \text{read}(F) \mid \text{write}(F, S) \mid \text{delete}(F)$

$do^-(K, A(\dots))$	“ K does action $A(\dots)$.”
$may^-(K, A(\dots))$	“ K may do action $A(\dots)$.”
$user^-(K)$	“ K is a registered user.”
$owns^-(K, F)$	“ K is an owner of file F .”

File System Example

Our file system has insert, read, write, and delete actions.

Actions $A(\dots) ::= \text{insert}() \mid \text{read}(F) \mid \text{write}(F, S) \mid \text{delete}(F)$

$do^-(K, A(\dots))$	“ K does action $A(\dots)$.”
$may^-(K, A(\dots))$	“ K may do action $A(\dots)$.”
$user^-(K)$	“ K is a registered user.”
$owns^-(K, F)$	“ K is an owner of file F .”

Files in the system are tagged with version tags.

Writes create new versions of a file; old versions are saved.

$current^-(F, T)$ “ T is the current version of file F .”

File System Example

Our file system has insert, read, write, and delete actions.

Actions $A(\dots) ::= \text{insert}() \mid \text{read}(F) \mid \text{write}(F, S) \mid \text{delete}(F)$

$\text{do}^-(K, A(\dots))$	“ K does action $A(\dots)$.”
$\text{may}^-(K, A(\dots))$	“ K may do action $A(\dots)$.”
$\text{user}^-(K)$	“ K is a registered user.”
$\text{owns}^-(K, F)$	“ K is an owner of file F .”

Files in the system are tagged with version tags.

Writes create new versions of a file; old versions are saved.

$\text{current}^-(F, T)$	“ T is the current version of file F .”
$\text{contents}^-(F, T, S)$	“Version T of file F contains string S .”

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example**
 - **Specification of Policy and Action Rules**
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

- Any registered user may insert files.

`mayinsert` : $\langle \text{fs} \rangle (user^-(K) \multimap may^-(K, insert()))$

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

File System Example: Policy Rules

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

- A principal may read, write, or delete files he owns.

`owner` : $\langle fs \rangle (owns^-(K, F) \multimap may^-(K, A(F, \dots)))$

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

- A principal may read, write, or delete files he owns.

`owner` : $\langle fs \rangle (owns^-(K, F) \multimap may^-(K, A(F, \dots)))$

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- Any registered user may insert files.

`mayinsert` : $\langle fs \rangle (user^-(K) \multimap may^-(K, insert()))$

- A principal may read, write, or delete files he owns.

`owner` : $\langle fs \rangle (owns^-(K, F) \multimap may^-(K, A(F, \dots)))$

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- A principal may read, write, or delete a file if the owner says so.

$\text{delegate} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, A(F, \dots)) \multimap \text{may}^-(L, A(F, \dots)))$

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- A principal may read, write, or delete a file if the owner says so.

$\text{delegate} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, A(F, \dots))) \multimap \text{may}^-(L, A(F, \dots))$

File System Example: Policy Rules

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- A principal may read, write, or delete a file if the owner says so.

$\text{delegate} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, A(F, \dots))) \multimap \text{may}^-(L, A(F, \dots))$

File System Example: Policy Rules

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- A principal may read, write, or delete a file if the owner says so.

$\text{delegate} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, A(F, \dots))) \multimap \text{may}^-(L, A(F, \dots))$

- Any registered user may insert files.

$\text{mayinsert} : \langle \text{fs} \rangle (\text{user}^-(K) \multimap \text{may}^-(K, \text{insert}()))$

- A principal may read, write, or delete files he owns.

$\text{owner} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \multimap \text{may}^-(K, A(F, \dots)))$

- A principal may read, write, or delete a file if the owner says so.

$\text{delegate} : \langle \text{fs} \rangle (\text{owns}^-(K, F) \otimes \langle K \rangle \text{may}^-(L, A(F, \dots))) \multimap$
 $\text{may}^-(L, A(F, \dots))$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle fs \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. !\langle fs \rangle owns^-(K, f) \otimes \\ & \quad [fs] current^-(f, t) \otimes \\ & \quad \llbracket fs \rrbracket contents^-(f, t, \epsilon) \otimes \\ & \quad \llbracket K \rrbracket contents^-(f, t, \epsilon) \} \end{aligned}$$

$\text{insert} : \langle K \rangle \text{do}^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle \text{may}^-(K, \text{insert}()) \multimap$
 $\{ \exists f. \exists t. !\langle \text{fs} \rangle \text{owns}^-(K, f) \otimes$
 $\quad [\text{fs}] \text{current}^-(f, t) \otimes$
 $\quad \llbracket \text{fs} \rrbracket \text{contents}^-(f, t, \epsilon) \otimes$
 $\quad \llbracket K \rrbracket \text{contents}^-(f, t, \epsilon) \}$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. !\langle \text{fs} \rangle owns^-(K, f) \otimes \\ & \quad [\text{fs}] current^-(f, t) \otimes \\ & \quad \llbracket \text{fs} \rrbracket contents^-(f, t, \epsilon) \otimes \\ & \quad \llbracket K \rrbracket contents^-(f, t, \epsilon) \} \end{aligned}$$

$\text{insert} : \langle K \rangle \text{do}^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle \text{may}^-(K, \text{insert}()) \multimap$
 $\{ \exists f. \exists t. !\langle \text{fs} \rangle \text{owns}^-(K, f) \otimes$
 $\quad [\text{fs}] \text{current}^-(f, t) \otimes$
 $\quad \llbracket \text{fs} \rrbracket \text{contents}^-(f, t, \epsilon) \otimes$
 $\quad \llbracket K \rrbracket \text{contents}^-(f, t, \epsilon) \}$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle fs \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. \text{!} \langle fs \rangle owns^-(K, f) \otimes \\ & \quad [fs] current^-(f, t) \otimes \\ & \quad \llbracket fs \rrbracket contents^-(f, t, \epsilon) \otimes \\ & \quad \llbracket K \rrbracket contents^-(f, t, \epsilon) \} \end{aligned}$$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. !\langle \text{fs} \rangle owns^-(K, f) \otimes \\ & \quad [\text{fs}] \text{current}^-(f, t) \otimes \\ & \quad \llbracket \text{fs} \rrbracket contents^-(f, t, \epsilon) \otimes \\ & \quad \llbracket K \rrbracket contents^-(f, t, \epsilon) \} \end{aligned}$$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. !\langle \text{fs} \rangle owns^-(K, f) \otimes \\ & \quad [\text{fs}] current^-(f, t) \otimes \\ & \quad \color{red}[[\text{fs}]] contents^-(f, t, \epsilon) \otimes \\ & \quad \color{red}[[K]] contents^-(f, t, \epsilon) \} \end{aligned}$$

$$\begin{aligned} \text{insert} : & \langle K \rangle do^-(K, \text{insert}()) \otimes \langle fs \rangle may^-(K, \text{insert}()) \multimap \\ & \{ \exists f. \exists t. !\langle fs \rangle owns^-(K, f) \otimes \\ & \quad [fs] current^-(f, t) \otimes \\ & \quad \llbracket fs \rrbracket contents^-(f, t, \epsilon) \otimes \\ & \quad \llbracket K \rrbracket contents^-(f, t, \epsilon) \} \end{aligned}$$

$$\text{insert} : \langle K \rangle \text{do}^-(K, \text{insert}()) \otimes \langle \text{fs} \rangle \text{may}^-(K, \text{insert}()) \multimap$$
$$\{ \exists f. \exists t. !\langle \text{fs} \rangle \text{owns}^-(K, f) \otimes$$
$$[\text{fs}] \text{current}^-(f, t) \otimes$$
$$[[\text{fs}]] \text{contents}^-(f, t, \epsilon) \otimes$$
$$[[K]] \text{contents}^-(f, t, \epsilon) \}$$

- Monad around action's conclusion
- Actions appear only on the main branch (rewrite sequence)

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

delete : $\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \mathbf{1} \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

delete : $\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \mathbf{1} \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

delete : $\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \mathbf{1} \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

delete : $\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \mathbf{1} \}$

File System Example: Semantic Action Rules

read : $\langle K \rangle do^-(K, read(F)) \otimes \langle fs \rangle may^-(K, read(F)) \otimes$
 $[fs]current^-(F, T) \otimes \llbracket fs \rrbracket contents^-(F, T, S) \multimap$
 $\{ \llbracket K \rrbracket contents^-(F, T, S) \otimes$
 $[fs]current^-(F, T) \}$

write : $\langle K \rangle do^-(K, write(F, S)) \otimes \langle fs \rangle may^-(K, write(F, S)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \exists t. [fs]current^-(F, t) \otimes$
 $\llbracket fs \rrbracket contents^-(F, t, S) \otimes$
 $\llbracket K \rrbracket contents^-(F, t, S) \}$

delete : $\langle K \rangle do^-(K, delete(F)) \otimes \langle fs \rangle may^-(K, delete(F)) \otimes$
 $[fs]current^-(F, T) \multimap$
 $\{ \mathbf{1} \}$

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example**
 - Specification of Policy and Action Rules
 - File System State**
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

Definition (File System State)

$\Gamma; \Delta$ is a *file system state* if and only if:

Definition (File System State)

$\Gamma; \Delta$ is a *file system state* if and only if:

- 1 Each assumption in Γ is either:
 - a policy rule (*mayinsert*, *owner*, or *delegate*)
 - an action rule (*insert*, *read*, *write*, or *delete*)
 - *fs* knows $contents^-(F, T, S)$
 - *K* knows $contents^-(F, T, S)$
 - $\langle fs \rangle user^-(K)$
 - $\langle fs \rangle owns^-(K, F)$

Definition (File System State)

$\Gamma; \Delta$ is a *file system state* if and only if:

1 Each assumption in Γ is either:

- a policy rule (*mayinsert*, *owner*, or *delegate*)
- an action rule (*insert*, *read*, *write*, or *delete*)
- *fs* knows $contents^-(F, T, S)$
- *K* knows $contents^-(F, T, S)$
- $\langle fs \rangle user^-(K)$
- $\langle fs \rangle owns^-(K, F)$

2 Each assumption in Δ is either:

- *fs* has $current^-(F, T)$
- $\langle K \rangle do^-(K, A(\dots))$
- $\langle K \rangle may^-(L, A(F, \dots))$

Definition (File System State)

$\Gamma; \Delta$ is a *file system state* if and only if:

- 1 Each assumption in Γ is either:
 - a policy rule (*mayinsert*, *owner*, or *delegate*)
 - an action rule (*insert*, *read*, *write*, or *delete*)
 - *fs* knows $\text{contents}^-(F, T, S)$
 - *K* knows $\text{contents}^-(F, T, S)$
 - $\langle \text{fs} \rangle \text{user}^-(K)$
 - $\langle \text{fs} \rangle \text{owns}^-(K, F)$
- 2 Each assumption in Δ is either:
 - *fs* has $\text{current}^-(F, T)$
 - $\langle K \rangle \text{do}^-(K, A(\dots))$
 - $\langle K \rangle \text{may}^-(L, A(F, \dots))$
- 3 For each F , there is at most one T such that *fs* has $\text{current}^-(F, T) \in \Delta$.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example**
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps**
 - Proving Properties about the File System
- 5 Conclusion

File System Example: Rewrite Steps

We would like to enumerate all possible rewrite steps for the file system. Recall $\Gamma; \Delta \rightarrow \Gamma'; \Delta'$ if and only if

$$\frac{\Gamma'; \Delta' \vdash C^+ \text{ lax}}{\vdots} \\ \frac{\Gamma; \Delta_2, A_2^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta_2, [\{A_2^+\}] \vdash C^+ \text{ lax}} \left. \vphantom{\frac{\Gamma; \Delta_2, A_2^+ \vdash C^+ \text{ lax}}{\Gamma; \Delta_2, [\{A_2^+\}] \vdash C^+ \text{ lax}}} \right\} L \\ \vdots \\ \frac{\Gamma; \Delta_1, [A_1^-] \vdash C^+ \text{ lax}}{\Gamma; \Delta \vdash C^+ \text{ lax}} *$$

If $\Gamma; \Delta$ is a file system state, then by construction only the actions have monadic heads.

Construct derived rules for each left focusing phase on an action.

Theorem (Rewrite Step Schema)

Each rewrite step from a file system state is either:

- $\Gamma; \langle K \rangle do^-(K, write(F, S)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2$
 $\rightarrow \Gamma, fs \text{ knows } contents^-(F, t, S), K \text{ knows } contents^-(F, t, S);$
 $\Delta_2, fs \text{ has } current^-(F, t)$
such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, write(F, S))$ for some fresh tag t .
- $\Gamma; \langle K \rangle do^-(K, delete(F)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2 \rightarrow \Gamma; \Delta_2$
such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, delete(F))$.
- Similar rewrite steps for *insert* and *read* elided.*

Construct derived rules for each left focusing phase on an action.

Theorem (Rewrite Step Schema)

Each rewrite step from a file system state is either:

- $\Gamma; \langle K \rangle do^-(K, write(F, S)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2$
 $\rightarrow \Gamma, fs \text{ knows } contents^-(F, t, S), K \text{ knows } contents^-(F, t, S);$
 $\Delta_2, fs \text{ has } current^-(F, t)$
such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, write(F, S))$ for some fresh tag t .
- $\Gamma; \langle K \rangle do^-(K, delete(F)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2 \rightarrow \Gamma; \Delta_2$
such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, delete(F))$.
- Similar rewrite steps for insert and read elided.*

Proof.

Left focusing on

$$\text{delete} : \langle K \rangle \text{do}^-(K, \text{delete}(F)) \otimes \langle \text{fs} \rangle \text{may}^-(K, \text{delete}(F)) \otimes \\ [\text{fs}] \text{current}^-(F, T) \multimap \\ \{\mathbf{1}\}$$

Proof.

Left focusing on

$$\text{delete} : \langle K \rangle \text{do}^-(K, \text{delete}(F)) \otimes \langle \text{fs} \rangle \text{may}^-(K, \text{delete}(F)) \otimes \\ [\text{fs}] \text{current}^-(F, T) \multimap \\ \{\mathbf{1}\}$$

yields the derived rule

$$\frac{\Gamma; \Delta'_1 \vdash \langle K \rangle \text{do}^-(K, \text{delete}(F)) \quad \Gamma|_{\text{fs}}; \Delta'_2|_{\text{fs}} \vdash \text{current}^-(F, T) \quad \Gamma; \Delta_1 \vdash \langle \text{fs} \rangle \text{may}^-(K, \text{delete}(F)) \quad \Gamma; \Delta_2, \mathbf{1} \vdash C^+ \text{ lax}}{\Gamma; \Delta'_1, \Delta_1, \Delta'_2|_{\text{fs}}, \Delta_2 \vdash C^+ \text{ lax}}$$

Proof.

Left focusing on

$$\text{delete} : \langle K \rangle do^-(K, \text{delete}(F)) \otimes \langle fs \rangle may^-(K, \text{delete}(F)) \otimes [fs]current^-(F, T) \multimap \{\mathbf{1}\}$$

yields the derived rule

$$\frac{\Gamma; \Delta'_1 \vdash \langle K \rangle do^-(K, \text{delete}(F)) \quad \Gamma|_{fs}; \Delta'_2|_{fs} \vdash current^-(F, T) \quad \Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, \text{delete}(F)) \quad \Gamma; \Delta_2, \mathbf{1} \vdash C^+ lax}{\Gamma; \Delta'_1, \Delta_1, \Delta'_2|_{fs}, \Delta_2 \vdash C^+ lax}$$

The $\mathbf{1}_L$ rule is invertible. So

$$\Gamma; \langle K \rangle do^-(K, \text{delete}(F)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2 \rightarrow \Gamma; \Delta_2$$

such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, \text{delete}(F))$. □

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example**
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System**
- 5 Conclusion

Theorem (Knowledge Safety)

If $\Gamma; \Delta$ is a file system state such that

$$\Gamma; \Delta \rightarrow \Gamma', K \text{ knows contents}^-(F, T, S); \Delta'$$

then either $K \text{ knows contents}^-(F, T, S) \in \Gamma$ or the step was an insert, read, or write step for F triggered by K .

Proof.

By case analysis of the possible rewrite step schema. □

Theorem (Knowledge Safety)

If $\Gamma; \Delta$ is a file system state such that

$$\Gamma; \Delta \rightarrow \Gamma', K \text{ knows contents}^-(F, T, S); \Delta'$$

then either $K \text{ knows contents}^-(F, T, S) \in \Gamma$ or the step was an insert, read, or write step for F triggered by K .

Proof.

By case analysis of the possible rewrite step schema. □

Principals do not learn file contents unless permitted by the policy!

Corollary

If

- $\Gamma; \langle K \rangle do^-(K, delete(F)), \Delta_1, fs \text{ has current}^-(F, T), \Delta_2$
- $\rightarrow \Gamma; \Delta_2$
- $\rightarrow^* \Gamma', L \text{ knows contents}^-(F, T', S); \Delta'$

such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, delete(F))$, *then*
 $L \text{ knows contents}^-(F, T', S) \in \Gamma$.

Corollary

If

- $\Gamma; \langle K \rangle do^-(K, delete(F)), \Delta_1, fs \text{ has } current^-(F, T), \Delta_2$
- $\rightarrow \Gamma; \Delta_2$
- $\rightarrow^* \Gamma', L \text{ knows } contents^-(F, T', S); \Delta'$

such that $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, delete(F))$, *then*
 $L \text{ knows } contents^-(F, T', S) \in \Gamma$.

Principals cannot learn the contents of deleted files!

Theorem (Delete Safety)

If $\Gamma; \Delta$, fs has $\text{current}^-(F, T)$ is a file system state such that

$$\Gamma; \Delta, \text{fs has } \text{current}^-(F, T) \rightarrow \Gamma'; \Delta'$$

and there is no T' such that fs has $\text{current}^-(F, T') \in \Delta'$, then the step was a delete.

Proof.

By case analysis of the rewrite step schema. □

Theorem (Delete Safety)

If $\Gamma; \Delta$, fs has $\text{current}^-(F, T)$ is a file system state such that

$$\Gamma; \Delta, \text{fs has } \text{current}^-(F, T) \rightarrow \Gamma'; \Delta'$$

and there is no T' such that $\text{fs has } \text{current}^-(F, T') \in \Delta'$, then the step was a delete.

Proof.

By case analysis of the rewrite step schema. □

Files do not spontaneously disappear!

Recall the rewrite steps each have a condition:

$$\Gamma; \Delta_1 \vdash \langle \text{fs} \rangle \text{may}^-(K, A(\dots))$$

File System Example: Delete Safety

Recall the rewrite steps each have a condition:

$$\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, A(\dots))$$

In principle, fs has $current^-(F, T)$ might be in Δ_1 .

So, we prove:

Lemma (Possession Strengthening)

If $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, A(\dots))$, then Δ_1 does not contain fs has $current^-(\dots)$.

File System Example: Delete Safety

Recall the rewrite steps each have a condition:

$$\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, A(\dots))$$

In principle, fs has $current^-(F, T)$ might be in Δ_1 .

So, we prove:

Lemma (Possession Strengthening)

If $\Gamma; \Delta_1 \vdash \langle fs \rangle may^-(K, A(\dots))$, then Δ_1 does not contain fs has $current^-(\dots)$.

This lemma requires stratification of policy and semantics.

- Visible syntactically: semantics contain $\{-\}$; policies do not.
- Semantics confined to the monad, and no monads in policies.

- 1 Introduction
- 2 Logic
 - Metatheory
- 3 Notions of Generic State and Step
- 4 File System Example
 - Specification of Policy and Action Rules
 - File System State
 - File System Rewrite Steps
 - Proving Properties about the File System
- 5 Conclusion

Goal:

- Formally specify authorization policies and their semantic consequences, and reason about them.

Goal:

- Formally specify authorization policies and their semantic consequences, and reason about them.

Method:

- 1 Specify policies and semantics in a linear logic with security modalities.
 - Affirmation, knowledge, and possession modalities
 - Stratify policies and semantics.

Goal:

- Formally specify authorization policies and their semantic consequences, and reason about them.

Method:

- 1 Specify policies and semantics in a linear logic with security modalities.
 - Affirmation, knowledge, and possession modalities
 - Stratify policies and semantics.
- 2 Define a notion of state for the system.

Goal:

- Formally specify authorization policies and their semantic consequences, and reason about them.

Method:

- 1 Specify policies and semantics in a linear logic with security modalities.
 - Affirmation, knowledge, and possession modalities
 - Stratify policies and semantics.
- 2 Define a notion of state for the system.
- 3 Interpret semantic specifications as rewrite steps.
 - rewrite sequence = trace of semantic actions
 - Use focusing to enforce atomicity of actions.
 - Confine semantics, but not policies, to a monad.

Goal:

- Formally specify authorization policies and their semantic consequences, and reason about them.

Method:

- 1 Specify policies and semantics in a linear logic with security modalities.
 - Affirmation, knowledge, and possession modalities
 - Stratify policies and semantics.
- 2 Define a notion of state for the system.
- 3 Interpret semantic specifications as rewrite steps.
 - rewrite sequence = trace of semantic actions
 - Use focusing to enforce atomicity of actions.
 - Confine semantics, but not policies, to a monad.
- 4 Using the rewrite steps, prove properties about the system.
 - E.g., knowledge and delete safety theorems.

- Subordination for general strengthening result

- Subordination for general strengthening result
- Properties involving order of actions

- Subordination for general strengthening result
- Properties involving order of actions
- Obligations as semantic effects:
 - To check out a library book, you incur an obligation to return the book.
 - To access normally protected records in an emergency, you incur an obligation to document the necessity.

- Subordination for general strengthening result
- Properties involving order of actions
- Obligations as semantic effects:
 - To check out a library book, you incur an obligation to return the book.
 - To access normally protected records in an emergency, you incur an obligation to document the necessity.
- Implementation à la LolliMon [López+05] and MSR [Cervesato+]

- Subordination for general strengthening result
- Properties involving order of actions
- Obligations as semantic effects:
 - To check out a library book, you incur an obligation to return the book.
 - To access normally protected records in an emergency, you incur an obligation to document the necessity.
- Implementation à la LolliMon [López+05] and MSR [Cervesato+]
- Case studies of more complex/realistic systems: e.g., protocols

Thank you!

Questions?