A Linear Logic of Authorization and Knowledge

Deepak Garg, Lujo Bauer, Kevin D. Bowers,
Frank Pfenning and Michael K. Reiter

Carnegie Mellon University

Why Logic?

- Express authorization questions
  - Does Deepak have a proof of the propostion

    Admin says may_access (Deepak, file)?
- Express policies
  - Admin says

    owns (Deepak, file) $\supset$ may_access(Deepak, file)
- Construct evidence (assemble proof)
- Verify evidence (verify proof)
- Reason from assumptions (given credentials)

Why logic … this paper

- Reason about state and state change
- Express private knowledge
- Prove properties of policies

- Verify policies
- Implement: logic programming

Authorization and Linearity

- Authorization
  - Someone wants to withdraw $10 from my account …
  - Should we allow it more than once?

- Logic
  - Is there a proof of <Deepak> check ($10)?
  - Is the proof linear?

Knowledge and Linearity

- Knowledge
  - foo.pdf is on the disk
  - Frank owns a check I signed
  - Can these facts be used more than once?

- Logic
  - [[disk]] file(foo.pdf)
    - (disk knows that foo.pdf is a file)
  - [Frank] <Deepak> check($10)
    - (Frank has a check for $10 signed by Deepak)
  - *The second fact should be usable only once!*

Main constructs in the logic

- Linearity
- Affirmation or intent

  <K> A (read “K asserts the truth of A”)
- Linear knowledge (resource possession)

  [K] A (read “K has resource A”)
- Non-linear knowledge

  [[K]] A (read “K knows fact A”)

Design emphasis

- Proof-theoretic, cut-elimination
- Intuitionistic logic
- Logical explanation of both intent (authorization) and knowledge (resource possession) with linearity
- Formal reasoning about state changes of systems

Example: Monetary systems

- Model checks, bank accounts and tradable items
- Use [K] A to model ownership of resources like checks and items
- Use <K> A to model intents expressed by principals (e.g. a check signed by K)

Cashing a Check

- "If Alice gives the bank a check for amount N signed by Bob , then the bank will transfer N dollars from Bob's account to Alice's account"

$$
\begin{aligned}
\texttt{ax1}: \quad & [\texttt{Alice}]\langle\texttt{Bob}\rangle\texttt{check}(\texttt{N}) \multimap \\
& [\texttt{bank}]\texttt{balance}(\texttt{Bob}, N_1) \multimap \\
& [\texttt{bank}]\texttt{balance}(\texttt{Alice}, N_2) \multimap \\
& \quad ([\texttt{bank}]\texttt{balance}(\texttt{Bob}, N_1 - N) \otimes \\
& \quad [\texttt{bank}]\texttt{balance}(\texttt{Alice}, N_2 + N))
\end{aligned}
$$

Trading Goods

- "If Alice has a check for amount N, and Bob has an item worth N, they may exchange the two."

$$
\begin{aligned}
\texttt{ax2}: \quad & [\texttt{Alice}]\langle\texttt{Charlie}\rangle\texttt{check(N)} \multimap \\
& [\texttt{Bob}]\texttt{item(N)} \multimap \\
& \quad ([\texttt{Alice}]\texttt{item(N)} \otimes \\
& \quad [\texttt{Bob}]\langle\texttt{Charlie}\rangle\texttt{check(N)})
\end{aligned}
$$

Writing Checks

- "Any principal Alice may sign a check for any amount"

$$\texttt{ax3}: \ [\texttt{Alice}]\langle\texttt{Alice}\rangle\texttt{check}(\texttt{N})$$

Example: State changes

$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Alice}, 1000),$$
$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Bob}, 500),$$
$$[\mathtt{Bob}]\mathtt{item}(200)$$

$\downarrow$ ax3 (Alice signs a check)

$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Alice}, 1000),$$
$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Bob}, 500),$$
$$[\mathtt{Bob}]\mathtt{item}(200),\ [\mathtt{Alice}]\langle\mathtt{Alice}\rangle\mathtt{check}(200)$$

$\downarrow$ ax2 (Bob gives item to Alice)

$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Alice}, 1000),$$
$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Bob}, 500),$$
$$[\mathtt{Alice}]\mathtt{item}(200),\ [\mathtt{Bob}]\langle\mathtt{Alice}\rangle\mathtt{check}(200)$$

Example: State changes (Contd.)

$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Alice}, 1000),$$
$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Bob}, 500),$$
$$[\mathtt{Alice}]\mathtt{item}(200),\ [\mathtt{Bob}]\langle\mathtt{Alice}\rangle\mathtt{check}(200)$$

$$\Big\downarrow \text{ ax1 (Bob deposits check)}$$

$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Alice}, 800),$$
$$[\mathtt{bank}]\mathtt{balance}(\mathtt{Bob}, 700),$$
$$[\mathtt{Alice}]\mathtt{item}(200)$$

Salient Points

- High level of abstraction
- Do not represent operational details
  - *Who* deducts amounts in bank accounts?
  - *How* is exchange of goods made?
- Atomicity in transactions is crucial
  - Significant burden in implementation
  - Necessary to prove correctness theorems

Atomicity in Steps

- Policy rules must be used atomically

Well-formed state

$$\Delta = [L]\langle L\rangle \mathtt{check}(200)$$

$$\mathtt{ax2}: \quad [L]\langle M\rangle \mathtt{check}(N) \multimap [K]\mathtt{item}(N) \multimap ([L]\mathtt{item}(N) \otimes [K]\langle M\rangle \mathtt{check}(N))$$

Partial Application

Bad state!

$$\Delta = \quad [K]\mathtt{item}(200) \multimap ([L]\mathtt{item}(200) \otimes [K]\langle L\rangle \mathtt{check}(200))$$

Logic Design with Judgments

- Judgments are the units of information
- Our judgments:
  - A true : resource A exists (linear)
  - K affirms A : K affirms the truth of A (linear)
  - K has A : K possesses resource A (linear)
  - A valid : proposition A holds (non-linear)
  - K knows A : K knows that fact A holds (non-linear)
- Deductions are evidence for judgments
- Connectives defined by right and left rules
- Right and left rules must match up
  - Cut elimination, identity

Hypothetical Judgments

$$\Gamma; \Delta \Longrightarrow A \text{ true}$$

$$\Gamma; \Delta \Longrightarrow K \text{ affirms } A$$

Unrestricted assumptions	Linear resources
A valid, K knows A	A true, K has A

We will ignore Γ for the rest of the talk

Init Rule

$$\frac{}{P \ \mathsf{true} \Longrightarrow P \ \mathsf{true}} \ (\texttt{init})$$

- P must be atomic
- No additional assumptions allowed
- Forces linearity

Implication

- Right rule

$$\frac{\Delta, A \ \mathsf{true} \Longrightarrow B \ \mathsf{true}}{\Delta \Longrightarrow A \multimap B \ \mathsf{true}} \ (\multimap R)$$

- Left rule

$$\frac{\Delta \Longrightarrow A \ \mathsf{true} \qquad \Delta', B \ \mathsf{true} \Longrightarrow \gamma}{\Delta, \Delta', A \multimap B \ \mathsf{true} \Longrightarrow \gamma} \ (\multimap L)$$

Affirmation: K affirms A

- K affirms A expresses *intent* (willingness to give money, permission to access a file, …)

- All principals are willing to affirm true statements

$$\frac{\Delta \Longrightarrow A \text{ true}}{\Delta \Longrightarrow K \text{ affirms } A} \text{ (affirms)}$$

The connective <K>A

- <K>A *internalizes* the judgment "affirms"
- Right rule

$$\frac{\Delta \Longrightarrow K \text{ affirms } A}{\Delta \Longrightarrow \langle K \rangle A \text{ true}} \; (\langle\rangle R)$$

- Left rule

$$\frac{\Delta, A \text{ true} \Longrightarrow K \text{ affirms } C}{\Delta, \langle K \rangle A \text{ true} \Longrightarrow K \text{ affirms } C} \; (\langle\rangle L)$$

Must match

<K>A as a Strong Monad

- Each <K> … is a strong monad
- Satisfies usual strong monad laws

$$A \multimap \langle K \rangle A$$
$$\langle K \rangle \langle K \rangle A \multimap \langle K \rangle A$$
$$(A \multimap B) \multimap \langle K \rangle A \multimap \langle K \rangle B$$

- Generalization of lax modality from lax logic

Resource possession: K has A

- K has A : principal K possesses resource A (K has $10, K has a check, …)

- Basic rule: If K has A is assumed, then resource A must exist

$$\frac{\Delta, A \ \mathsf{true} \Longrightarrow \gamma}{\Delta, K \ \mathsf{has} \ A \Longrightarrow \gamma} \ (\mathsf{has})$$

The connective [K] A

- [K] A internalizes (K has A)
- Left rule:

$$\frac{\Delta, K \text{ has } A \Longrightarrow \gamma}{\Delta, [K]A \text{ true} \Longrightarrow \gamma} \; ([]L)$$

- Right rule:

Δ restricted to assumptions of the form (K has A)

$$\frac{\Delta|_K \Longrightarrow A \text{ true}}{\Delta|_K \Longrightarrow [K]A \text{ true}} \; ([]R)$$

[K]A is a box modality

- Each [K] … is a constructive S4 □ modality

- Satisfies usual laws:

$$[K]A \multimap A$$
$$[K]A \multimap [K][K]A$$
$$[K](A \multimap B) \multimap [K]A \multimap [K]B$$

Cut-elimination

1. If $\Delta \Longrightarrow A$ true
   and Δ', A true $\Longrightarrow \gamma$,
   then $\Delta, \Delta' \Longrightarrow \gamma$.

2. If $\Delta \Longrightarrow K$ affirms A
   and Δ', A true $\Longrightarrow K$ affirms C,
   then $\Delta, \Delta' \Longrightarrow K$ affirms C.

3. If $\Delta|_K \Longrightarrow A$ true
   and Δ', K has $A \Longrightarrow \gamma$,
   then $\Delta|_K, \Delta' \Longrightarrow \gamma$

Identity

$$\overline{A \ \mathsf{true} \Longrightarrow A \ \mathsf{true}}$$

- Cut-elimination is a statement of soundness of the logic
- Identity is completeness

Consequences

- Important Properties:

$$\not\Longrightarrow \bot \ \mathsf{true}$$
$$\not\Longrightarrow (\langle K \rangle A \multimap A) \ \mathsf{true}$$
$$\not\Longrightarrow (A \multimap [K]A) \ \mathsf{true}$$

- Subformula property
- Independence: More connectives can be added through right and left rules

States of the system

- A state is a context Δ, with some suitable conditions imposed
- In the monetary example, Δ contains resources of the following forms only:

  [bank] balance(K, N)

  [K] item(N)

  [K] <L> check(N)
- Exactly one assumption [bank] balance(K, N) for each K

Policy rules as inference rules

- Need a mechanism to force atomicity of policy rules in the logic

$$\begin{aligned}\texttt{ax2}: \quad & [\texttt{Alice}]\langle\texttt{Charlie}\rangle\texttt{check(N)} \multimap \\ & [\texttt{Bob}]\texttt{item(N)} \multimap \\ & \quad ([\texttt{Alice}]\texttt{item(N)} \otimes \\ & \quad [\texttt{Bob}]\langle\texttt{Charlie}\rangle\texttt{check(N)})\end{aligned}$$

$$\Downarrow$$

$$\frac{\begin{array}{c}\Delta_1 \Longrightarrow [\texttt{Alice}]\langle\texttt{Charlie}\rangle\texttt{check(N)} \\ \Delta_2 \Longrightarrow [\texttt{Bob}]\texttt{item(N)} \\ \Delta_3, \texttt{Alice has item}(N), \texttt{Bob has } \langle\texttt{Charlie}\rangle\texttt{check(N)} \Longrightarrow \gamma\end{array}}{\Delta_1, \Delta_2, \Delta_3 \Longrightarrow \gamma}\,(\texttt{ax2})$$

Policy rules as inference rules

$$\frac{\begin{array}{c}\Delta_1 \Longrightarrow [\mathtt{Alice}]\langle\mathtt{Charlie}\rangle\mathtt{check(N)} \\ \Delta_2 \Longrightarrow [\mathtt{Bob}]\mathtt{item(N)} \\ \Delta_3, \mathtt{Alice\ has\ item}(N), \mathtt{Bob\ has}\ \langle\mathtt{Charlie}\rangle\mathtt{check(N)} \Longrightarrow \gamma\end{array}}{\Delta_1, \Delta_2, \Delta_3 \Longrightarrow \gamma}\ (\mathtt{ax2})$$

- Automatic enforcement of atomicity in logic (inference rule cannot be applied partially)
- Method can be generalized to arbitrary policies (focusing)
- *Implementation* still needs to deal with atomicity

Formal definition of step

$\Delta \longrightarrow \Delta'$ iff there is a derivation

$$\Delta' \Longrightarrow \gamma$$
$$\vdots$$
$$\Delta \Longrightarrow \gamma$$

that is parametric in γ.

- Derivations contain:
  - Logic's rules
  - Policy as rules

Correctness theorems

- Can prove (very formally in the logic) the following theorems
  - The total sum of money in assumptions of the form [bank] balance(K, N) remains invariant across steps (real money remains constant)
  - The net assets of every principal are invariant across steps (principals cannot be cheated)

Extensions

- Analysis of policies for unintended consequences
- Realistic implementation in a PCA style architecture
  - Whole logic hard to implement, need to find a good fragment
- Formalization of states and steps using a logic programming approach

Most Closely Related Work

- Authorization logics
  - [Abadi, Burrows, Lampson, Plotkin'93] propositional, rich calculus of principals
  - [De Treville'02] Binder

    Datalog fragment, decidable, logic programming, modality unclear
- Epistemic logics
  - "K-operator" is similar to [K] A

Most Closely Related Work

- Linear Logic
  - [Girard'87] Linear Logic

    Modeling resources in logic
  - [Andreoli'92] Focusing

    Chaining inference rules together, similar to atomicity here
  - [Cervesato'03] Multi-set rewriting

    Using linear logic to model security systems

Summary

- Novel integration of affirmation, linearity, knowledge
- Can model intent, resources, use-once authorizations and possession
- Simple proof theory, cut-elimination
- Formalization of system state and steps in the logic itself
- Correctness theorems for policy rules