

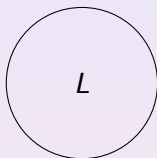
# An Operational Semantics for Trust Policies

Karl Krukow

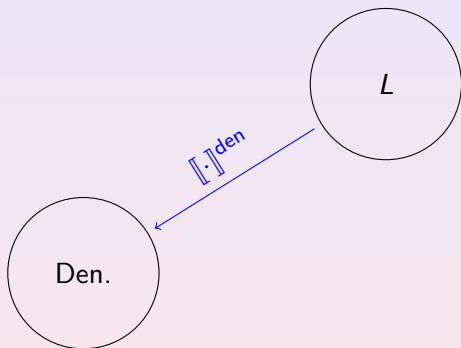
Department of Computer Science  
University of Aarhus, Denmark  


Workshop on Issues in the Theory of Security, 2006

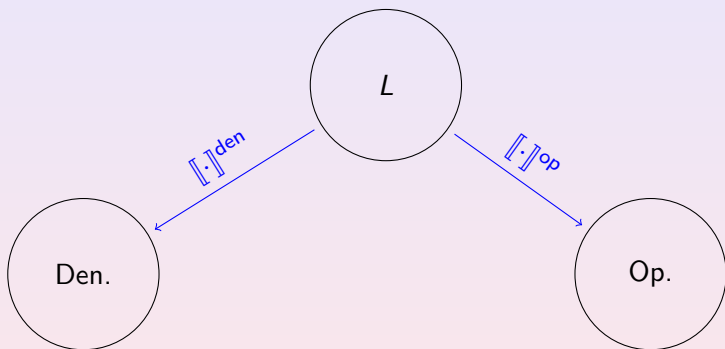
# A Familiar Picture



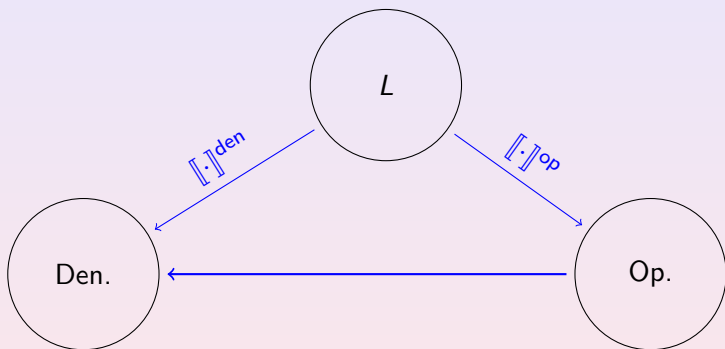
# A Familiar Picture



# A Familiar Picture



# A Familiar Picture



# Domain

## Trust Management Systems

- The domain of this work is *trust management* (TM) systems.
  - ▶ Supports security decision-making in open, large-scale distributed applications.
- Key Concept:
  - ▶ Security Policy.
  - ▶ A corresponding notion of compliance with security policy.
- Foundations of TM has been the subject of sophisticated theoretical work.
  - ▶ e.g., Mitchell; Li; Feigenbaum; Weeks; Shmatikov & Talcott; Carbone, Nielsen & Sassone; others ...

# Domain

## Trust Management Systems

- Our work extends the trust-structure framework: a general semantic model for trust.
  - ▶ M. Carbone, M. Nielsen and V. Sassone.  
*A Formal Model for Trust in Dynamic Networks* (2003).
- In line with the original ideas of TM.
  - ▶ Based on distributed ‘trust-policies.’
  - ▶ Flexible.
  - ▶ Separates mechanism from policy (infact, **there is no mechanism at all!**).
- This last point is exactly the subject of this work.
  - ▶ Complement the framework’s denotational semantics with a formal **operational semantics**.

# Domain

## Trust Management Systems

- Our work extends the trust-structure framework: a general semantic model for trust.
  - ▶ M. Carbone, M. Nielsen and V. Sassone.  
*A Formal Model for Trust in Dynamic Networks* (2003).
- In line with the original ideas of TM.
  - ▶ Based on distributed ‘trust-policies.’
  - ▶ Flexible.
  - ▶ Separates mechanism from policy (infact, **there is no mechanism at all!**).
- This last point is exactly the subject of this work.
  - ▶ Complement the framework’s denotational semantics with a formal **operational semantics**.

# Outline

## 1 Trust Structures

- Introduction
- The Formal Model
- Motivation: The Operational Problem

## 2 Our Contribution

- A Distributed Algorithm for Computing Least Fixed-Points
- Formalization using I/O Automata
- Correspondence result

# Outline

## 1 Trust Structures

- Introduction
- The Formal Model
- Motivation: The Operational Problem

## 2 Our Contribution

- A Distributed Algorithm for Computing Least Fixed-Points
- Formalization using I/O Automata
- Correspondence result

# Trust Structures: An Introduction

- Provides a generic mathematical framework, formalizing and solving the following problem.
  - ▶ Given a set  $\mathcal{P}$  of principal identities, each specifying a local trust-policy, define a unique *global trust-state* compatible with those policies.
- A *global trust-state*?
  - ▶ formally, a *global trust state* is a function  $\mathbf{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D = \mathbf{GTS}$  for some set  $D$  of “degrees of trust” (called trust values).
  - ▶ for each  $p, q \in \mathcal{P}$  answer: “to what degree does  $p$  trust  $q$ ?” as  $\mathbf{gts}(p)(q)$ .
- A *generic model*?
  - ▶ Different applications have different requirements for trust-information.
  - ▶ Obtained by choosing set  $D$  of trust degrees, and by choosing appropriate trust policies.
  - ▶ Example instances, KeyNote, SPKI, SECURE Trust model, ...

# Trust Structures: An Introduction

- Provides a generic mathematical framework, formalizing and solving the following problem.
  - ▶ Given a set  $\mathcal{P}$  of principal identities, each specifying a local trust-policy, define a unique *global trust-state* compatible with those policies.
- A *global trust-state*?
  - ▶ formally, a *global trust state* is a function  $\mathbf{gts} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow D = \mathbf{GTS}$  for some set  $D$  of “degrees of trust” (called trust values).
  - ▶ for each  $p, q \in \mathcal{P}$  answer: “to what degree does  $p$  trust  $q$ ?” as  $\mathbf{gts}(p)(q)$ .
- Trust policies?
  - ▶ Specifies how a principal defines its trust in others.
  - ▶  $\pi_p$  : “my trust in Alice is **high** and for anyone else, it is the minimum of what Bob and Carl think.”

# Trust Structures: Trust Policies

## A Simple Language

$$\pi ::= \star : \tau$$
$$| p : \tau, \pi$$

$$\tau ::= d$$
$$| p?q$$
$$| \text{op}_n^i(\tau_1, \tau_2, \dots, \tau_n)$$

E.g., the policy  $\pi_p$  of principal  $p$  could be

$$A : \text{high}, \star : (B?\star) \wedge (C?\star)$$

# Trust Structures: Trust Policies

## A Simple Language

$$\pi ::= \star : \tau$$
$$| p : \tau, \pi$$

E.g., the policy  $\pi_p$  of principal  $p$  could be

$$\tau ::= d$$
$$| p?q$$
$$| \text{op}_n^i(\tau_1, \tau_2, \dots, \tau_n)$$

$A : \text{high}, \star : (B?\star) \wedge (C?\star)$

Denotationally, simply functions mapping *global* trust-states to *local* trust states,  $\llbracket \pi_p \rrbracket^{\text{den}} : \mathbf{GTS} \rightarrow \mathbf{LTS}$ , i.e.,  $\llbracket \pi_p \rrbracket^{\text{den}} : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow D) \rightarrow \mathcal{P} \rightarrow D$ .

# Trust Structures: The Formal Model (1/3)

- Refining the main objective of the model.
  - ▶ Fix a set  $\mathcal{P}$  of principal identities.
  - ▶ Each principal  $p \in \mathcal{P}$  specifying a local trust-policy  $\pi_p$ .  
Let  $\Pi = (\pi_p \mid p \in \mathcal{P})$  be the (global) collection of all these policies,  
 $\llbracket \pi_p \rrbracket^{\text{den}} : \mathbf{GTS} \rightarrow \mathbf{LTS}$ .
  - ▶ Goal is to define a suitable **unique** *global trust-state*, denoted  $\llbracket \Pi \rrbracket^{\text{den}}$ , compatible with all the policies  $\Pi$ .
- But... policies may have *cyclic references*.
  - ▶ Example:  $\pi_p = \star : q? \star$ ,  $\pi_q = \star : p? \star$ .
  - ▶ In this example, the policies contain *no information*: this is distinct from explicitly specifying *untrusted*.
- This example suggests that the set  $D$  of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - ▶ trust / privilege ( $\preceq$ ), e.g. low  $\preceq$  high.
  - ▶ information content ( $\sqsubseteq$ ), e.g. unknown  $\sqsubseteq$  low.

# Trust Structures: The Formal Model (1/3)

- Refining the main objective of the model.
  - ▶ Fix a set  $\mathcal{P}$  of principal identities.
  - ▶ Each principal  $p \in \mathcal{P}$  specifying a local trust-policy  $\pi_p$ .  
Let  $\Pi = (\pi_p \mid p \in \mathcal{P})$  be the (global) collection of all these policies,  
 $\llbracket \pi_p \rrbracket^{\text{den}} : \mathbf{GTS} \rightarrow \mathbf{LTS}$ .
  - ▶ Goal is to define a suitable **unique** *global trust-state*, denoted  $\llbracket \Pi \rrbracket^{\text{den}}$ , compatible with all the policies  $\Pi$ .
- But... policies may have cyclic *references*.
  - ▶ Example:  $\pi_p = \star : q? \star$ ,  $\pi_q = \star : p? \star$ .
  - ▶ In this example, the policies contain *no information*: this is distinct from explicitly specifying *untrusted*.
- This example suggests that the set  $D$  of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - ▶ trust / privilege ( $\preceq$ ), e.g. low  $\preceq$  high.
  - ▶ information content ( $\sqsubseteq$ ), e.g. unknown  $\sqsubseteq$  low.

# Trust Structures: The Formal Model (1/3)

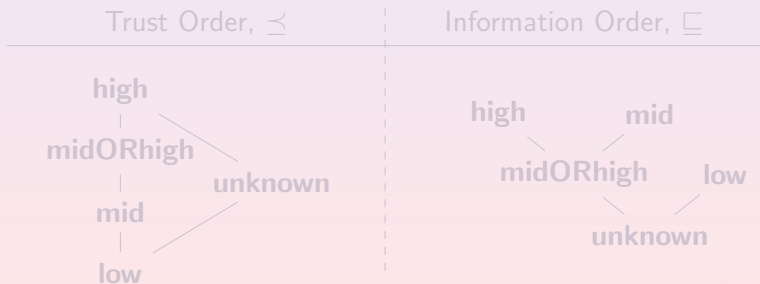
- Refining the main objective of the model.
  - ▶ Fix a set  $\mathcal{P}$  of principal identities.
  - ▶ Each principal  $p \in \mathcal{P}$  specifying a local trust-policy  $\pi_p$ .  
Let  $\Pi = (\pi_p \mid p \in \mathcal{P})$  be the (global) collection of all these policies,  
 $\llbracket \pi_p \rrbracket^{\text{den}} : \mathbf{GTS} \rightarrow \mathbf{LTS}$ .
  - ▶ Goal is to define a suitable **unique** *global trust-state*, denoted  $\llbracket \Pi \rrbracket^{\text{den}}$ , compatible with all the policies  $\Pi$ .
- But... policies may have *cyclic references*.
  - ▶ Example:  $\pi_p = \star : q? \star$ ,  $\pi_q = \star : p? \star$ .
  - ▶ In this example, the policies contain *no information*: this is distinct from explicitly specifying *untrusted*.
- This example suggests that the set  $D$  of trust-values can be ordered in two fundamentally distinct ways: with respect to
  - ▶ trust / privilege ( $\preceq$ ), e.g. low  $\preceq$  high.
  - ▶ information content ( $\sqsubseteq$ ), e.g. unknown  $\sqsubseteq$  low.

# Trust Structures: The Formal Model (2/3)

## Definition [Trust Structure]

A *trust structure* is a triple  $T = (D, \preceq, \sqsubseteq)$ , consisting of a set  $D$  and two orderings on  $D$ , called the *trust ordering* ( $\preceq$ ) and the *information ordering* ( $\sqsubseteq$ ). The trust ordering preorders  $D$  and the information order makes  $(D, \sqsubseteq)$  a complete partial order.

- Example  $D = \{\text{low}, \text{mid}, \text{high}, \text{unknown}, \text{midORhigh}\}$ .

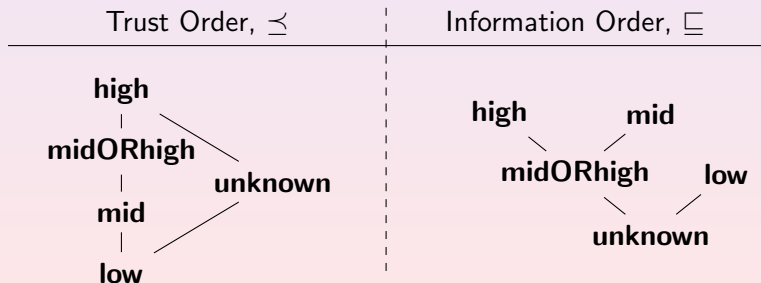


# Trust Structures: The Formal Model (2/3)

## Definition [Trust Structure]

A *trust structure* is a triple  $T = (D, \preceq, \sqsubseteq)$ , consisting of a set  $D$  and two orderings on  $D$ , called the *trust ordering* ( $\preceq$ ) and the *information ordering* ( $\sqsubseteq$ ). The trust ordering preorders  $D$  and the information order makes  $(D, \sqsubseteq)$  a complete partial order.

- Example  $D = \{\text{low}, \text{mid}, \text{high}, \text{unknown}, \text{midORhigh}\}$ .



## Trust Structures: The Formal Model (3/3)

- Let  $T = (D, \preceq, \sqsubseteq)$  be a trust structure, and  $\Pi = (\pi_p \mid p \in \mathcal{P})$  be a collection of trust policies.
- A *fixed point* of  $\Pi$  is a global trust-state **gts** so that for all  $p \in \mathcal{P}$  we have  $\pi_p(\mathbf{gts}) = \mathbf{gts}(p)$ .
  - ▶ Any fixed point of  $\Pi$  is *consistent with all policies*.
- Assuming that all policies are *continuous with respect to*  $\sqsubseteq$ .

### Definition $[[\Pi]]^{\text{den}}$

Define the global trust-state of  $\Pi$  in  $T$  as the  $\sqsubseteq$ -least fixed-point of  $\Pi$ .

$$[[\Pi]]^{\text{den}} \stackrel{(\text{def})}{=} \text{lfp}_{\sqsubseteq} \Pi$$

## Trust Structures: The Formal Model (3/3)

- Let  $T = (D, \preceq, \sqsubseteq)$  be a trust structure, and  $\Pi = (\pi_p \mid p \in \mathcal{P})$  be a collection of trust policies.
- A *fixed point* of  $\Pi$  is a global trust-state **gts** so that for all  $p \in \mathcal{P}$  we have  $\pi_p(\mathbf{gts}) = \mathbf{gts}(p)$ .
  - ▶ Any fixed point of  $\Pi$  is *consistent with all policies*.
- Assuming that all policies are *continuous with respect to*  $\sqsubseteq$ .

### Definition $[[\Pi]]^{\text{den}}$

Define the global trust-state of  $\Pi$  in  $T$  as the  $\sqsubseteq$ -least fixed-point of  $\Pi$ .

$$[[\Pi]]^{\text{den}} \stackrel{(\text{def})}{=} \text{lfp}_{\sqsubseteq} \Pi$$

# The Operational Problem (1/2)

- Suppose principal  $p$  has to make a *trust-based* access-control decision about another principal  $q$ .
- A *trust-based security policy* is a function of the trust in the requestor.
  - ▶  $\sigma : D \rightarrow \{\top, \perp\}$
  - ▶ A monotonic security policy satisfies  $d \preceq d' \Rightarrow \sigma(d) \Rightarrow \sigma(d')$ .
  - ▶ E.g., threshold policies: Allow access to  $q$  if my trust in  $q$  is trust-wise above threshold  $t \in D$ . (e.g.,  $t = \mathbf{high}$ )
- So one *mechanism* for deciding a request could be
  - ▶ compute  $x := \llbracket \Pi \rrbracket^{\text{den}}(p)(q)$
  - ▶ feed this value to policy,  $\sigma(x) = \top?$

# The Operational Problem (2/2)

- A **denotational** model:
  - ▶  $\llbracket \Pi \rrbracket^{\text{den}}$  is a well defined, unique mathematical object.
- But there is “no recipe for getting there”?
  - ▶ One might say there is no mechanism for evaluating policies.
- How do we actually *compute* the trust values, when
  - ▶  $\Pi$  is distributed.
  - ▶  $|\mathcal{P}|$  is large.
  - ▶ no centralized authority.
  - ▶ policy updates ...
- The standard technique for fixed point computation:  
 $\perp_{\subseteq} \subseteq \Pi(\perp_{\subseteq}) \subseteq \Pi^2(\perp_{\subseteq}) \dots ?$ 
  - ▶ Inadequate!

# The Operational Problem (2/2)

- A **denotational** model:
  - ▶  $\llbracket \Pi \rrbracket^{\text{den}}$  is a well defined, unique mathematical object.
- But there is “no recipe for getting there”?
  - ▶ One might say there is no mechanism for evaluating policies.
- How do we actually *compute* the trust values, when
  - ▶  $\Pi$  is distributed.
  - ▶  $|\mathcal{P}|$  is large.
  - ▶ no centralized authority.
  - ▶ policy updates ...
- The standard technique for fixed point computation:  
 $\perp_{\subseteq} \subseteq \Pi(\perp_{\subseteq}) \subseteq \Pi^2(\perp_{\subseteq}) \dots ?$ 
  - ▶ Inadequate!

# Outline

## 1 Trust Structures

- Introduction
- The Formal Model
- Motivation: The Operational Problem

## 2 Our Contribution

- A Distributed Algorithm for Computing Least Fixed-Points
- Formalization using I/O Automata
- Correspondence result

# Overview: operational techniques

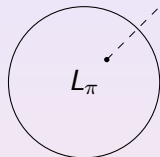
- An asynchronous distributed algorithm for computing  $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$  for arbitrary but *fixed*  $p, q \in \mathcal{P}$ .
  - ▶ Only nodes necessary for computing  $p$ 's trust in  $q$  are involved.
- Approximation Algorithms.
  - ▶ 'Proof-carrying' requests (client presents proof to speed-up access decision).
  - ▶ Snapshot-based approximation.
  - ▶ Generalized Protocol (combination of both).
- Techniques described previously (no rigorous formalization) (Krukow and Twigg, 2005).
- Our contribution here is a **formalization and proof** of correctness of the asynchronous algorithm, using I/O Automata.
- See the full paper (Krukow and Nielsen, 2006) for the complete formalization.

# Overview: operational techniques

- An asynchronous distributed algorithm for computing  $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$  for arbitrary but *fixed*  $p, q \in \mathcal{P}$ .
  - ▶ Only nodes necessary for computing  $p$ 's trust in  $q$  are involved.
- Approximation Algorithms.
  - ▶ 'Proof-carrying' requests (client presents proof to speed-up access decision).
  - ▶ Snapshot-based approximation.
  - ▶ Generalized Protocol (combination of both).
- Techniques described previously (no rigorous formalization) (Krukow and Twigg, 2005).
- Our contribution here is a **formalization and proof** of correctness of the asynchronous algorithm, using I/O Automata.
- See the full paper (Krukow and Nielsen, 2006) for the complete formalization.

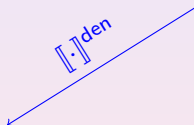
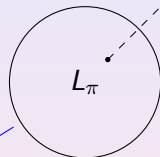
# Overview

$\pi = A : \text{high}, \star : B? \star \wedge C? \star$



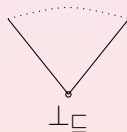
# Overview

$$\pi = A : \text{high}, \star : B? \star \wedge C? \star$$



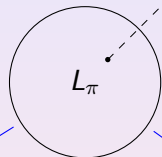
World of Mogens: Ideas

$$f : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow D, \sqsubseteq) \rightarrow (\mathcal{P} \rightarrow D, \sqsubseteq)$$



# Overview

$$\pi = A : \text{high}, \star : B? \star \wedge C? \star$$



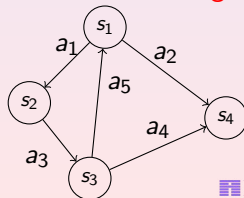
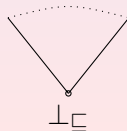
$[\cdot]_{\text{den}}$

World of Mogens: Ideas

$[\cdot]_{\text{op}}$

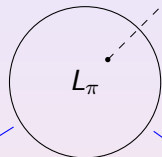
World of Karl: Pragmatics

$$f : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow D, \sqsubseteq) \rightarrow (\mathcal{P} \rightarrow D, \sqsubseteq)$$



# Overview

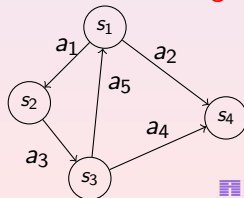
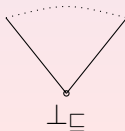
$$\pi = A : \text{high}, \star : B? \star \wedge C? \star$$



World of Mogens: Ideas

World of Karl: Pragmatics

$$f : (\mathcal{P} \rightarrow \mathcal{P} \rightarrow D, \sqsubseteq) \rightarrow (\mathcal{P} \rightarrow D, \sqsubseteq)$$



# A Distributed Asynchronous Algorithm

- Scenario.
  - ▶ Principal  $p$  needs to compute its trust in  $q$ , i.e., the value  $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$ .
  - ▶ Principals are network nodes with computational and communication capacity.
    - ★ Asynchronous, but reliable network.
- Basic observation.
  - ▶ Compute the **local** value  $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$  directly, rather than computing the global state  $\llbracket \Pi \rrbracket^{\text{den}}$  and then looking up “entry”  $(p, q)$ .
  - ▶ Excludes principals that are not relevant for the specific computation.
- Algorithm: two-step computation.
  - ▶ Dependency analysis distributedly computes a sub-graph  $G_{(p,q)}$  of the dependency graph  $G$  for the policies  $\Pi$ .
  - ▶ Asynchronous fixed-point algorithm in dependency graph.

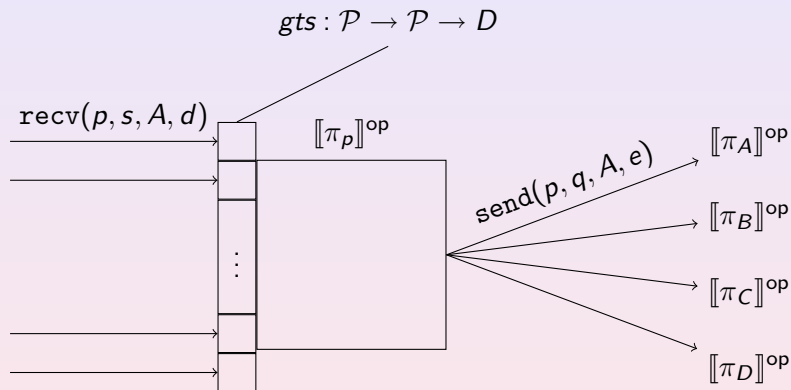
# A Least-Fixed-Point Algorithm (1/2)

- Very simple algorithm; instance of a framework for asynchronous fixed-point algorithms (Bertsekas and Tsitsiklis, 1989).
- Tailored for local **least** fixed-points ( $\llbracket \Pi \rrbracket^{\text{den}}(p)(q)$ ).
- Essentially a distributed, asynchronous version of the synchronous iteration:

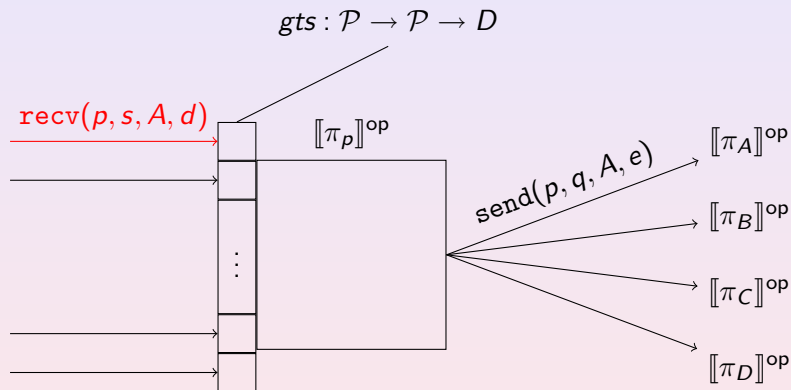
$$\perp_{\square} \sqsubseteq \Pi(\perp_{\square}) \sqsubseteq \dots \sqsubseteq \Pi^i(\perp_{\square})$$

- Proved correct using the **Asynchronous Convergence Theorem** of Bertsekas; the following invariant is essentially maintained:
  - ▶ Node  $p$ , is able to compute a sequence of values  $\perp_{\square} \sqsubseteq t_0 \sqsubseteq \dots \sqsubseteq t_n = \llbracket \Pi \rrbracket^{\text{den}}(p)(q)$  converging towards its local fixed point value.
  - ▶ Similar convergence holds at other nodes.

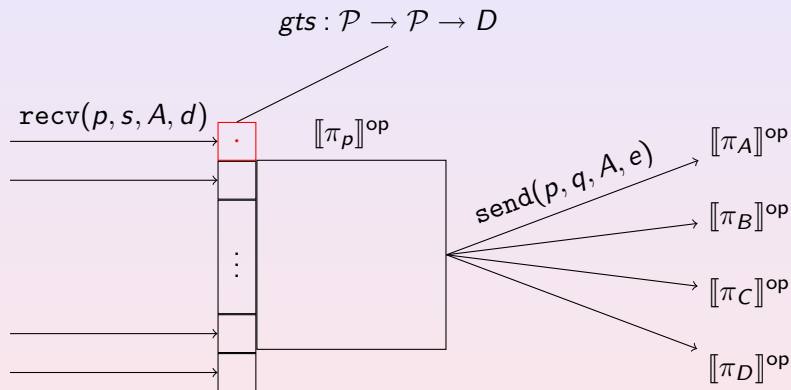
# A Least-Fixed-Point Algorithm (2/2)



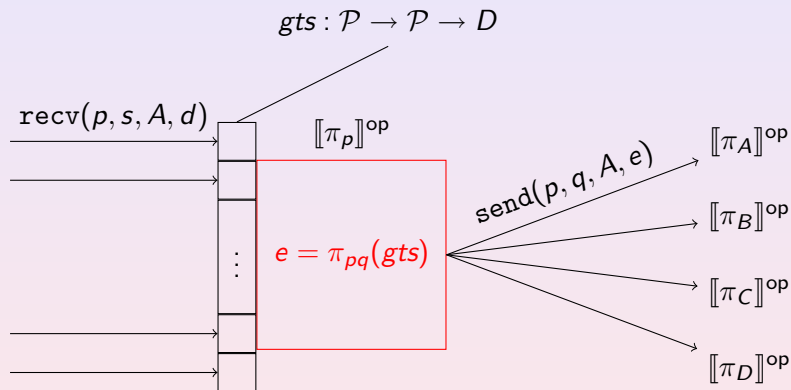
# A Least-Fixed-Point Algorithm (2/2)



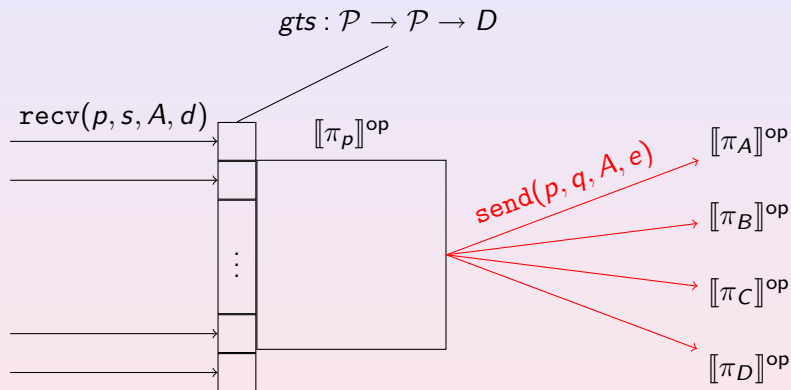
# A Least-Fixed-Point Algorithm (2/2)



# A Least-Fixed-Point Algorithm (2/2)

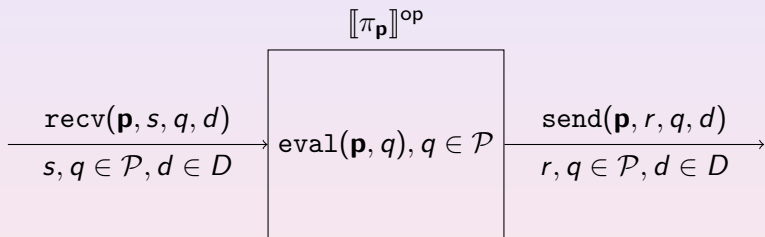


# A Least-Fixed-Point Algorithm (2/2)



# Formalization

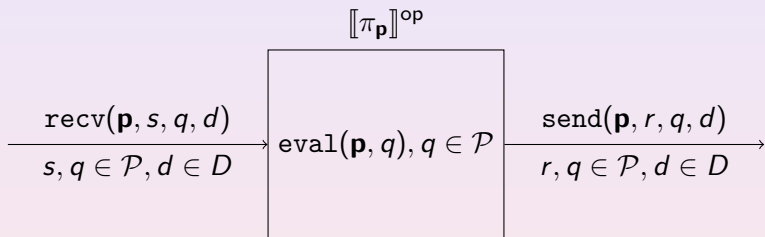
Each policy  $\pi_{\mathbf{p}}$  of each principal  $\mathbf{p} \in \mathcal{P}$  is mapped to an I/O automaton, modelling a node in the algorithm.



A web of policies,  $\Pi = (\pi_p \mid p \in \mathcal{P})$  is mapped to a composition of its parts,  $\llbracket \pi_p \rrbracket^{\text{op}}$ .

# Formalization

Each policy  $\pi_{\mathbf{p}}$  of each principal  $\mathbf{p} \in \mathcal{P}$  is mapped to an I/O automaton, modelling a node in the algorithm.



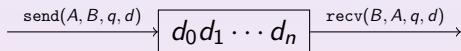
A web of policies,  $\Pi = (\pi_p \mid p \in \mathcal{P})$  is mapped to a composition of its parts,  $\llbracket \pi_p \rrbracket^{\text{op}}$ .

# Formalization: Composition

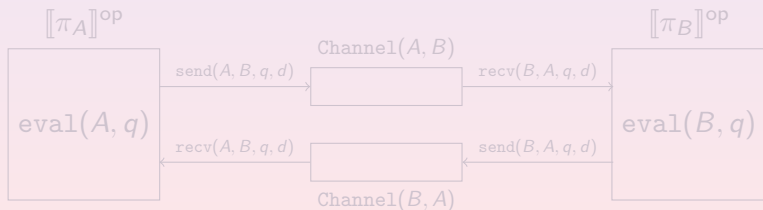
## Example: two principals

For a collection, say,  $\Pi = [A \mapsto \pi_A, B \mapsto \pi_B]$ .

Define  $\llbracket \Pi \rrbracket^{\text{op}}$  as a composition of the policy-automata, e.g.,  $\llbracket \pi_A \rrbracket^{\text{op}}$ , and channel automata e.g.,  $\text{Channel}(A, B)$ :



$\llbracket \Pi \rrbracket^{\text{op}}$  is the following composition:

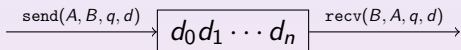


# Formalization: Composition

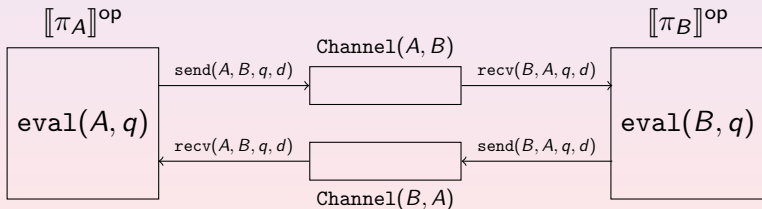
## Example: two principals

For a collection, say,  $\Pi = [A \mapsto \pi_A, B \mapsto \pi_B]$ .

Define  $\llbracket \Pi \rrbracket^{\text{op}}$  as a composition of the policy-automata, e.g.,  $\llbracket \pi_A \rrbracket^{\text{op}}$ , and channel automata e.g.,  $\text{Channel}(A, B)$ :

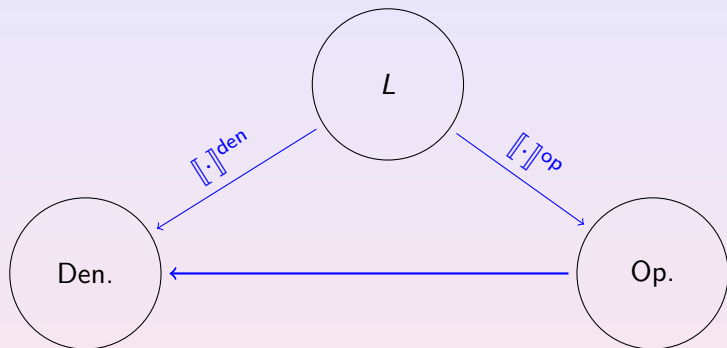


$\llbracket \Pi \rrbracket^{\text{op}}$  is the following composition:



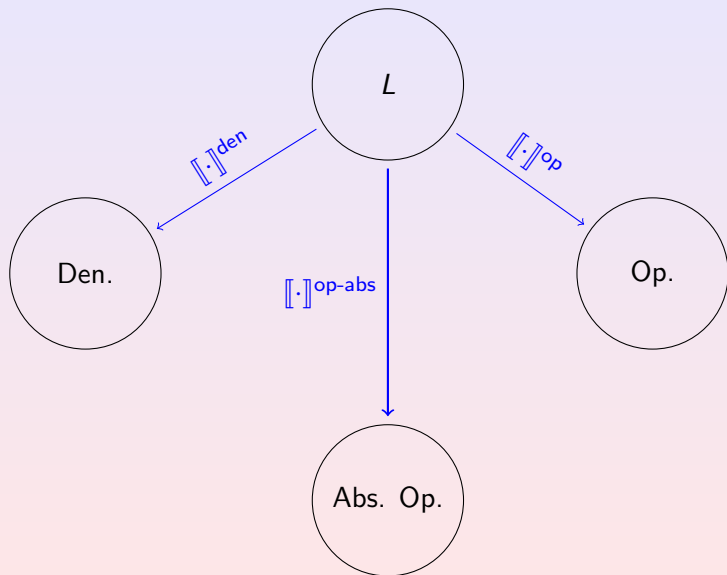
# Correspondence of Semantics

## Proof Structure



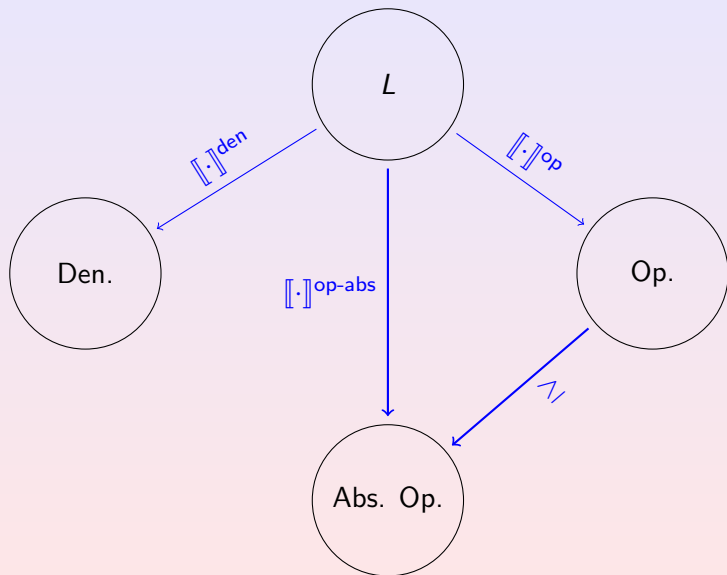
# Correspondence of Semantics

## Proof Structure



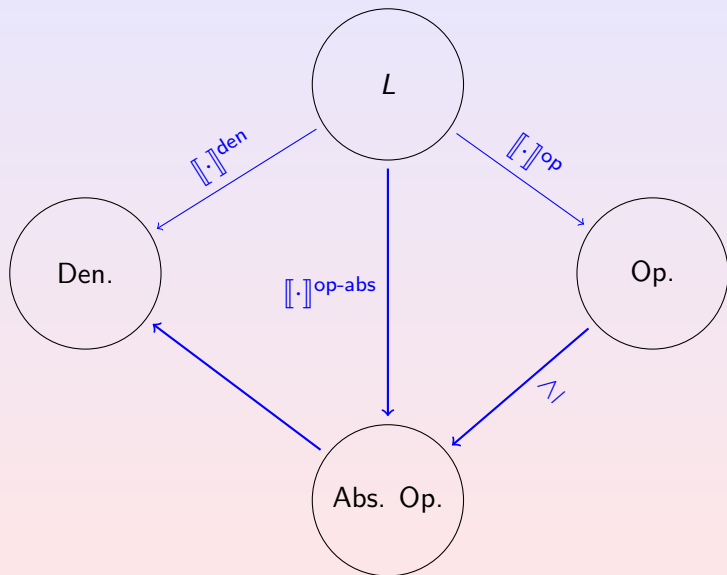
# Correspondence of Semantics

## Proof Structure



# Correspondence of Semantics

## Proof Structure



# Summary

- The trust-structure framework is a denotational semantic model for trust-management systems.
- We have provided a formal operational semantics for a general language of policies.
- Operational Semantics is proven to converge to the denotational semantics.

## For Further Reading

- Bertsekas, D. P. and Tsitsiklis, J. N.: 1989,  
*Parallel and Distributed Computation: Numerical Methods*,  
Prentice-Hall International Editions, Prentice-Hall, Inc.
- Carbone, M., Nielsen, M., and Sassone, V.: 2003,  
in *Proceedings from Software Engineering and Formal Methods (SEFM'03)*, IEEE Computer Society Press
- Krukow, K. and Nielsen, M.: 2006,  
*Distributed Trust Management: Denotational and Operational Semantics*,  
Submitted. Available online <http://www.brics.dk/~krukow>
- Krukow, K. and Twigg, A.: 2005,  
in *Proceedings from the 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, pp 805–814, IEEE