

Towards a Theory of Trust for the Global Ubiquitous Computer

Karl Krukow



Department of Computer Science
University of Aarhus, Denmark
<http://www.brics.dk/~krukow>

PhD Defence, Århus, November 17, 2006

Ubiquitous Computing

the vision

- Billions of ubiquitous computational entities.
 - ▶ Memory, network and computational capabilities (though often limited).
 - ▶ Global connectivity.
 - ▶ Often mobile.
 - ▶ Autonomous, incomplete information.
 - ▶ Context-aware.
- Yet, they
 - ▶ Verifiably correct (purposeful).
 - ▶ Operate securely.
 - ▶ Timely.
 - ▶ High failure rate at component level, yet reliable at system level.

Ubiquitous Computing

the vision

- Billions of ubiquitous computational entities.
 - ▶ Memory, network and computational capabilities (though often limited).
 - ▶ Global connectivity.
 - ▶ Often mobile.
 - ▶ Autonomous, incomplete information.
 - ▶ Context-aware.
- Yet, they
 - ▶ Verifiably correct (purposeful).
 - ▶ Operate securely.
 - ▶ Timely.
 - ▶ High failure rate at component level, yet reliable at system level.

Role of Trust in Ubiquitous Computing

Traditional Authorization

- Traditional Authorization
 - ▶ Authentication + Access Control List.
 - ▶ Closed-world:
 - ★ Each identity has meaning.
- Issues in ubiquitous systems
 - ▶ Very large set of entities that can potentially make requests.
 - ▶ Open-world: Entities joining and leaving networks.
 - ★ What about requests from unknown identities?
 - ▶ Expressive power and flexibility.
 - ▶ Active decisions.

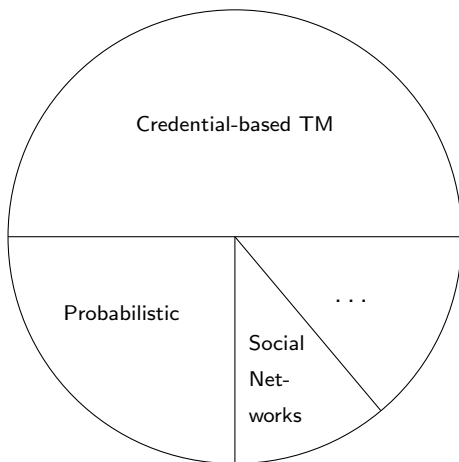
Role of Trust in Ubiquitous Computing

Traditional Authorization

- Traditional Authorization
 - ▶ Authentication + Access Control List.
 - ▶ Closed-world:
 - ★ Each identity has meaning.
- Issues in ubiquitous systems
 - ▶ Very large set of entities that can potentially make requests.
 - ▶ Open-world: Entities joining and leaving networks.
 - ★ What about requests from unknown identities?
 - ▶ Expressive power and flexibility.
 - ▶ Active decisions.

Trust Management

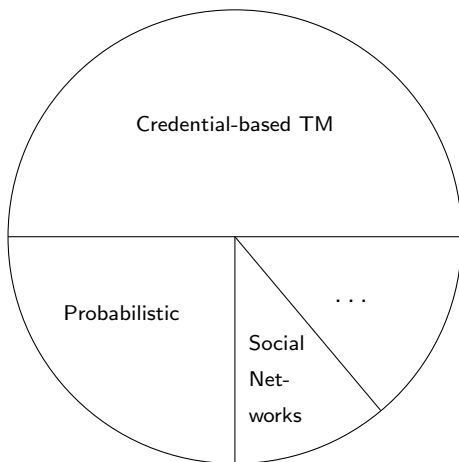
One alternative approach to decision-making



- Deals with *computer representations* of trust.
 - ▶ Decision making.
 - ▶ **Not** as a human emotion, or any kind of philosophical or sociological concept.

Trust Management

One alternative approach to decision-making



- Deals with *computer representations* of trust.
 - ▶ Decision making.
 - ▶ **Not** as a human emotion, or any kind of philosophical or sociological concept.

Security?

When information is about past behavior, what kinds of security properties can be established? Not the traditional:

- Only the identities of ACL get access to resources?
- If p gets access to resource r the authority A has authorized it.

Experience-based Security?

If interaction with entity p at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

Intended application is not access to a Bank account!

Security?

When information is about past behavior, what kinds of security properties can be established? Not the traditional:

- Only the identities of ACL get access to resources?
- If p gets access to resource r the authority A has authorized it.

Experience-based Security?

If interaction with entity p at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

Intended application is not access to a Bank account!

Security?

When information is about past behavior, what kinds of security properties can be established? Not the traditional:

- Only the identities of ACL get access to resources?
- If p gets access to resource r the authority A has authorized it.

Experience-based Security?

If interaction with entity p at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

Intended application is not access to a Bank account!

Example

History-based Access Control

- Program p can access resource r at time t only if p 's execution history up until time t satisfies ψ .
- Example: Edjlali *et al.*
 - ▶ Suppose you've downloaded what claims to be a new cool browser from some web page.
 - ▶ “allow program to connect to a remote site if-and-only-if it has neither tried to **open a local file that it has not created**, nor tried to **modify a file it has created**, nor tried to **create a sub-process**.”

Example

History-based Access Control

- Program p can access resource r at time t only if p 's execution history up until time t satisfies ψ .
- Example: Edjlali *et al.*
 - ▶ Suppose you've downloaded what claims to be a new cool browser from some web page.
 - ▶ “allow program to connect to a remote site if-and-only-if it has neither tried to **open a local file that it has not created**, nor tried to **modify a file it has created**, nor tried to **create a sub-process**.”

Outline

- 1 A Concrete Model of Behavioral Information
- 2 A Logical Approach to Experience-based Trust Management
 - Intermezzo: Short Demo of JavaHBAC
- 3 From Simulations to Theorems?
 - Probabilistic examples
 - Results for (simple) probabilistic systems
- 4 The Future?

What else is in the dissertation?

Overview

- Chapter 1 & 2: More on Ubiquitous Computing and the notion of Grand Challenges. Selected survey of TM.
- Chapter 3: Overview of results, outlook for future research.
- Chapter 4: The trust model of the SECURE project.
 - ▶ An event structure instance of trust structures.
 - ▶ Probabilistic model based on event structures and Bayesian analysis with Dirichlet priors. (Generalizes Beta systems)
- Chapter 5: An operational semantics for trust policies.
 - ▶ I/O-Automata-based operational semantics which is proved to agree with the denotational semantics of Carbone, Nielsen and Sassone (A Formal Model for Trust in Dynamic Networks).
- Chapter 6: A Logical Approach to Experience-based Trust Management.
 - ▶ Basic policy language and semantics. Algorithms based on finite automata and dynamic programming.

Outline

- 1 **A Concrete Model of Behavioral Information**
- 2 **A Logical Approach to Experience-based Trust Management**
 - Intermezzo: Short Demo of JavaHBAC
- 3 **From Simulations to Theorems?**
 - Probabilistic examples
 - Results for (simple) probabilistic systems
- 4 **The Future?**

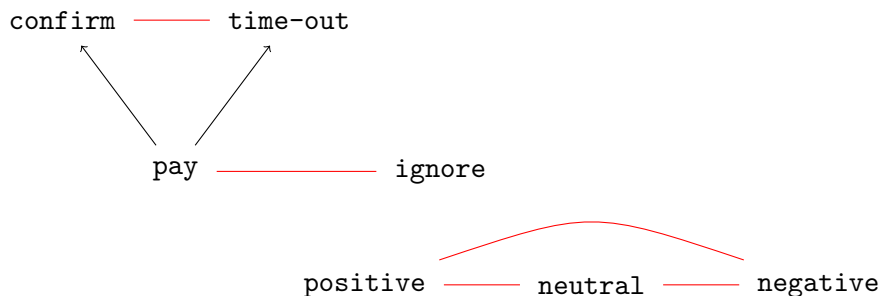
An event-structure model

- Primary target environment: concurrent systems.
 - ▶ Entities in a concurrent system interact following protocols.
 - ▶ Behavioral information is information about a number of past protocol-runs (sessions).
- We use event structures as an abstract model of protocols.
 - ▶ A protocol is often specified as a concurrent process.
 - ▶ Event structures were invented to give formal semantics to concurrent processes.
 - ▶ More formally, $ES = (E, \leq, \#)$, E a set of events, \leq and $\#$ (causality and conflict) relations on E .

Event structures

Example

Simple eBay example:

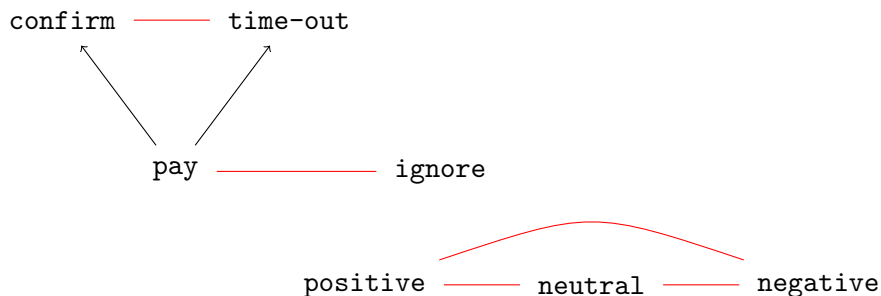


- Information about a session is a finite set of events $x \subseteq E$, called a **configuration** (which is always conflict free and causally closed).
- A history h , is a sequence of sessions, $h = x_1 x_2 \cdots x_n \in \mathcal{C}_{ES}^*$
 - E.g., $h = \{\text{pay}, \text{confirm}, \text{pos}\} \{\text{pay}, \text{confirm}, \text{neu}\} \{\text{pay}\}$

Event structures

Example

Simple eBay example:



- Information about a session is a finite set of events $x \subseteq E$, called a **configuration** (which is always conflict free and causally closed).
- A history h , is a sequence of sessions, $h = x_1 x_2 \cdots x_n \in \mathcal{C}_{ES}^*$
 - ▶ E.g., $h = \{\text{pay}, \text{confirm}, \text{pos}\} \{\text{pay}, \text{confirm}, \text{neu}\} \{\text{pay}\}$

Interface

- An entity may be involved in several concurrent protocol-runs.
- We use the following interface (where $h = x_1x_2 \cdots x_n$),
 - ▶ **update**(e, i), event e occurs in session i , where $e \in E$ and i is a session-index.
 - ★ $h.\mathbf{update}(e, i) = x_1x_2 \cdots (x_i \cup \{e\}) \cdots x_n$.
 - ▶ **new**(\cdot), creates a new session.
 - ★ $h.\mathbf{new}() = h\emptyset$.
- Hence, a stream of observations,

new().upd($e_1, 1$).new().upd($e_2, 1$).upd($e_1, 2$).upd($e_3, 1$).upd($e_4, 2$)

gets grouped into sessions, which becomes a history

$$h = \{e_1, e_2, e_3\}\{e_1, e_4\}$$

Outline

- 1 A Concrete Model of Behavioral Information
- 2 A Logical Approach to Experience-based Trust Management**
 - Intermezzo: Short Demo of JavaHBAC
- 3 From Simulations to Theorems?
 - Probabilistic examples
 - Results for (simple) probabilistic systems
- 4 The Future?

Experience-based Security

If interaction with entity p occurs at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

- Specification language.
 - ▶ How to specify requirements ψ declaratively and expressively.
- Dynamic verification problem.
 - ▶ Given h and ψ does h satisfy ψ ?
 - ▶ Information is provided incrementally via operations: **update**(e, i) and **new**(\cdot).

Experience-based Security

If interaction with entity p occurs at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

- Specification language.
 - ▶ How to specify requirements ψ declaratively and expressively.
- Dynamic verification problem.
 - ▶ Given h and ψ does h satisfy ψ ?
 - ▶ Information is provided incrementally via operations: **update**(e, i) and **new**().

Experience-based Security

If interaction with entity p occurs at time t , then the past behavior of p up *until* time t satisfies requirement ψ .

- Specification language.
 - ▶ How to specify requirements ψ declaratively and expressively.
- Dynamic verification problem.
 - ▶ Given h and ψ does h satisfy ψ ?
 - ▶ Information is provided incrementally via operations: **update**(e, i) and **new**().

Parametrized Events

- Recall example property: “. . . [never] open a local file that it has not created”
 - ▶ $\text{open}(f)$ and $\text{create}(f)$ could be events.
 - ▶ But infinitely many possible file-names
- Need a notion of *parametrized* event structure.
 - ▶ Events occur with parameters from (infinite) parameter sets.
 - ▶ Otherwise as usual event structures.

Parametrized Events

- Recall example property: "... [never] open a local file that it has not created ..."
 - ▶ $\text{open}(f)$ and $\text{create}(f)$ could be events.
 - ▶ But infinitely many possible file-names
- Need a notion of *parametrized* event structure.
 - ▶ Events occur with parameters from (infinite) parameter sets.
 - ▶ Otherwise as usual event structures.

Quantified Pure-Past Linear Temporal Logic

- Syntax.

$$\psi ::= e(x) \mid \diamond e(x) \mid \psi_0 \wedge \psi_1 \mid \psi_0 \vee \psi_1 \mid \neg \psi \mid X^{-1} \psi \mid \psi_0 S \psi_1 \mid Qx.\psi$$

(Q is \forall or \exists) (F^{-1} and G^{-1} encoded using S)

- Semantics: relation $(h, i) \models^\sigma \psi(\mathbf{x})$

- Semantics by example ;-)

- ▶ $(\{a, b(3)\} \cdot \{c\}, 1) \models b(3) \wedge \exists x.b(x)$ but not $b(4) \vee \forall x.b(x)$.

- ▶ $(\{a(2)\} \cdot \{a(3)\} \cdot \{b, c('secret.txt')\}, 3) \models^{[x \mapsto 2]} F^{-1}(a(x))$

- ▶ $(\{a(1)\} \cdot \{a(2)\} \cdot \{b, a(3)\}, 3) \models G^{-1}(\exists x.a(x))$

- ▶ $(\{b(2)\} \cdot \{a(2), b(4)\} \cdot \{a(4)\}, 3) \models G^{-1}\forall x. [a(x) \rightarrow F^{-1}b(x)]$.

Quantified Pure-Past Linear Temporal Logic

- Syntax.

$$\psi ::= e(x) \mid \diamond e(x) \mid \psi_0 \wedge \psi_1 \mid \psi_0 \vee \psi_1 \mid \neg \psi \mid X^{-1} \psi \mid \psi_0 S \psi_1 \mid Qx.\psi$$

(Q is \forall or \exists) (F^{-1} and G^{-1} encoded using S)

- Semantics: relation $(h, i) \models^\sigma \psi(\mathbf{x})$

- Semantics by example ;-)

- ▶ $(\{a, b(3)\} \cdot \{c\}, 1) \models b(3) \wedge \exists x.b(x)$ but not $b(4) \vee \forall x.b(x)$.

- ▶ $(\{a(2)\} \cdot \{a(3)\} \cdot \{b, c('secret.txt')\}, 3) \models^{[x \mapsto 2]} F^{-1}(a(x))$

- ▶ $(\{a(1)\} \cdot \{a(2)\} \cdot \{b, a(3)\}, 3) \models G^{-1}(\exists x.a(x))$

- ▶ $(\{b(2)\} \cdot \{a(2), b(4)\} \cdot \{a(4)\}, 3) \models G^{-1}\forall x. [a(x) \rightarrow F^{-1}b(x)]$.

Quantified Pure-Past Linear Temporal Logic

- Syntax.

$$\psi ::= e(x) \mid \diamond e(x) \mid \psi_0 \wedge \psi_1 \mid \psi_0 \vee \psi_1 \mid \neg \psi \mid X^{-1} \psi \mid \psi_0 S \psi_1 \mid Qx.\psi$$

(Q is \forall or \exists) (F^{-1} and G^{-1} encoded using S)

- Semantics: relation $(h, i) \models^\sigma \psi(\mathbf{x})$

- Semantics by example ;-)

- ▶ $(\{a, b(3)\} \cdot \{c\}, 1) \models b(3) \wedge \exists x.b(x)$ but not $b(4) \vee \forall x.b(x)$.

- ▶ $(\{a(2)\} \cdot \{a(3)\} \cdot \{b, c('secret.txt')\}, 3) \models^{[x \mapsto 2]} F^{-1}(a(x))$

- ▶ $(\{a(1)\} \cdot \{a(2)\} \cdot \{b, a(3)\}, 3) \models G^{-1}(\exists x.a(x))$

- ▶ $(\{b(2)\} \cdot \{a(2), b(4)\} \cdot \{a(4)\}, 3) \models G^{-1} \forall x. [a(x) \rightarrow F^{-1} b(x)]$.

Example

- Recall example property: "... [never] open a local file that it has not created ..."
 - ▶ We want *for any* file f "if $\text{open}(f)$ then $F^{-1}\text{create}(f)$."
 - ▶ But infinitely many possible file-names
- Specify property as

$$G^{-1} (\forall x. [\text{open}(x) \rightarrow F^{-1}(\text{create}(x))])$$

Example

- Recall example property: "... [never] open a local file that it has not created ..."
 - ▶ We want *for any* file f "if $\text{open}(f)$ then $F^{-1}\text{create}(f)$."
 - ▶ But infinitely many possible file-names
- Specify property as

$$G^{-1} (\forall x. [\text{open}(x) \rightarrow F^{-1}(\text{create}(x))])$$

Verifying quantified policies

- Verification problem.
 - ▶ Given history h and closed quantified formula ψ does $h \models \psi$ hold?
- Unfortunately, the verification problem is *PSPACE* complete even in single-element models.
- Decidable although we are quantifying over infinitely many values.

Verifying quantified policies

- Verification problem.
 - ▶ Given history h and closed quantified formula ψ does $h \models \psi$ hold?
- Unfortunately, the verification problem is *PSPACE* complete even in single-element models.
- Decidable although we are quantifying over infinitely many values.

Verifying quantified policies

- Verification problem.
 - ▶ Given history h and closed quantified formula ψ does $h \models \psi$ hold?
- Unfortunately, the verification problem is *PSPACE* complete even in single-element models.
- Decidable although we are quantifying over infinitely many values.

Verifying quantified policies

A Complex Dynamic Programming Algorithm in One Slide

x_1	x_2	\dots	x_{i-1}	x_i	\dots	x_m
s_1	s_2	\dots	s_{i-1}	s_i	\dots	s_m

- The states s_j are arrays of constraints (one entry for each subformula).
- Think of constraints as sets of substitutions.

Invariant

$$\forall \sigma. (h, k) \models^\sigma \psi_i \iff \sigma \in s_k[i]$$

Verifying quantified policies

A Complex Dynamic Programming Algorithm in One Slide

x_1	x_2	\dots	x_{i-1}	x_i	\dots	x_m
s_1	s_2	\dots	s_{i-1}	s_i	\dots	s_m

- The states s_i are arrays of constraints (one entry for each subformula).
- Think of constraints as sets of substitutions.

Invariant

$$\forall \sigma. (h, k) \models^\sigma \psi_i \iff \sigma \in s_k[i]$$

Verifying quantified policies

the good news...

- We have a result which bounds the worst-case running-time of our algorithm.
 - ▶ **update** is “only” exponential in the number of free-variables of formula ψ , with the base of the exponential being “ O -of” the number of occurred distinct parameters.
 - ▶ In practice, if policies do not have too many variables, we are OK.
- Constraints can be efficiently represented and manipulated using Binary Decision Diagrams (BDDs).
 - ▶ Need to evaluate efficiency in practice.

Short Demo

`https://sourceforge.net/projects/javahbac`

Outline

- 1 A Concrete Model of Behavioral Information
- 2 A Logical Approach to Experience-based Trust Management
 - Intermezzo: Short Demo of JavaHBAC
- 3 From Simulations to Theorems?**
 - Probabilistic examples
 - Results for (simple) probabilistic systems
- 4 The Future?

Simple Probabilistic Systems

The model:

- Assume that the behavior of each principal p is so that there is a fixed parameter such that at each interaction we have, *independently of anything we know about other interactions*, the probability θ_p for a 'success' and therefore probability $1 - \theta_p$ for 'failure.'

The specification:

- **Interface** (Trust computation algorithm, \mathcal{A}):
 - ▶ Input: A sequence $h = x_1x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
 - ▶ Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.
- **Goal:**
 - ▶ When the input h is obtained by interacting n times with principal p , output π should approximate $(\theta_p, 1 - \theta_p)$ as good as possible.

Simple Probabilistic Systems

The model:

- Assume that the behavior of each principal p is so that there is a fixed parameter such that at each interaction we have, *independently of anything we know about other interactions*, the probability θ_p for a 'success' and therefore probability $1 - \theta_p$ for 'failure.'

The specification:

- **Interface** (Trust computation algorithm, \mathcal{A}):
 - ▶ Input: A sequence $h = x_1x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
 - ▶ Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.
- **Goal**:
 - ▶ When the input h is obtained by interacting n times with principal p , output π should approximate $(\theta_p, 1 - \theta_p)$ as good as possible.

Beta-Based (Mui et al.)

The specification:

- Input: A sequence $h = x_1 x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
- Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.

The Algorithm: Bayesian Analysis (uniform prior)

$$\mathcal{A}_1(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h) + 1}{|h| + 2}, \quad \mathcal{A}_1(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h) + 1}{|h| + 2}$$

($N_x(h)$ = “number of x 's in h ”)

Properties:

- Well defined semantics: $\mathcal{A}_1(\mathbf{s} \mid h)$ is interpreted as a *probability of success* in the next interaction.
- Results: Mui: Chernoff Bound.

Beta-Based (Mui et al.)

The specification:

- Input: A sequence $h = x_1 x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
- Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.

The Algorithm: Bayesian Analysis (uniform prior)

$$\mathcal{A}_1(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h) + 1}{|h| + 2}, \quad \mathcal{A}_1(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h) + 1}{|h| + 2}$$

($N_x(h)$ = “number of x 's in h ”)

Properties:

- Well defined semantics: $\mathcal{A}_1(\mathbf{s} \mid h)$ is interpreted as a *probability* of success in the next interaction.
- Results: Mui: Chernoff Bound.

Maximum Likelihood

(Aberer and Despotovic)

The specification:

- Input: A sequence $h = x_1 x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
- Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.

The Algorithm: Max-Likelihood

$$\mathcal{A}_0(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h)}{|h|}, \quad \mathcal{A}_0(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h)}{|h|}$$

Properties:

- Well defined semantics: $\mathcal{A}_0(\mathbf{s} \mid h)$ is interpreted as a *probability* of success in the next interaction.
- Result, $\frac{N_{\mathbf{s}}(h)}{|h|} \rightarrow \theta_p$ as $n \rightarrow \infty$.

Maximum Likelihood

(Aberer and Despotovic)

The specification:

- Input: A sequence $h = x_1 x_2 \cdots x_n$ for $n \geq 0$ and $x_i \in \{\mathbf{s}, \mathbf{f}\}$.
- Output: A probability distribution $\pi : \{\mathbf{s}, \mathbf{f}\} \rightarrow [0, 1]$.

The Algorithm: Max-Likelihood

$$\mathcal{A}_0(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h)}{|h|}, \quad \mathcal{A}_0(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h)}{|h|}$$

Properties:

- Well defined semantics: $\mathcal{A}_0(\mathbf{s} \mid h)$ is interpreted as a *probability* of success in the next interaction.
- Result, $\frac{N_{\mathbf{s}}(h)}{|h|} \rightarrow \theta_p$ as $n \rightarrow \infty$.

Comparing Systems

Question

Which algorithm, Aberer et al. or Mui et al., best approximates $(\theta_p, 1 - \theta_p)$?

Simulation Answer (traditional)

Set up a simulation environment: Repeat the following a number of times and average the results: Choose a parameter θ uniformly from $[0, 1]$. Sample histories $h = x_1 \cdots x_n$ of various sizes. Compare $\mathcal{A}_0(\cdot | h)$ with $\mathcal{A}_1(\cdot | h)$, e.g., “mean absolute error.”

It is not clear how much this depends on: the number of repetitions, the outcomes of θ -choices, the sizes of histories, the measure (e.g., “mean absolute error”).

Comparing Systems

Question

Which algorithm, Aberer et al. or Mui et al., best approximates $(\theta_p, 1 - \theta_p)$?

Simulation Answer (traditional)

Set up a simulation environment: Repeat the following a number of times and average the results: Choose a parameter θ uniformly from $[0, 1]$. Sample histories $h = x_1 \cdots x_n$ of various sizes. Compare $\mathcal{A}_0(\cdot | h)$ with $\mathcal{A}_1(\cdot | h)$, e.g., “mean absolute error.”

It is not clear how much this depends on: the number of repetitions, the outcomes of θ -choices, the sizes of histories, the measure (e.g., “mean absolute error”).

Comparing Systems

Question

Which algorithm, Aberer et al. or Mui et al., best approximates $(\theta_p, 1 - \theta_p)$?

Simulation Answer (traditional)

Set up a simulation environment: Repeat the following a number of times and average the results: Choose a parameter θ uniformly from $[0, 1]$. Sample histories $h = x_1 \cdots x_n$ of various sizes. Compare $\mathcal{A}_0(\cdot | h)$ with $\mathcal{A}_1(\cdot | h)$, e.g., “mean absolute error.”

It is not clear how much this depends on: **the number of repetitions, the outcomes of θ -choices, the sizes of histories, the measure** (e.g., “mean absolute error”).

Comparing Systems – II

Question

Which algorithm, Aberer et al. or Mui et al., best approximates $(\theta_p, 1 - \theta_p)$?

But in some sense the problem is not precisely specified: what does it mean to

“(...) approximate $(\theta_p, 1 - \theta_p)$ as good as possible.”

Comparing Systems – II

Question

Which algorithm, Aberer et al. or Mui et al., best approximates $(\theta_p, 1 - \theta_p)$?

But in some sense the problem is not precisely specified: what does it mean to

“(...) approximate $(\theta_p, 1 - \theta_p)$ as good as possible.”

Simple Probabilistic Systems

Let us define a simple probabilistic system for computational trust as given by the following:

- A (finite) set of possible outcomes of each interaction
 $O = \{o_1, o_2, \dots, o_m\}$.
- A *probabilistic model*, λ , of principal behavior. Model λ defines the following probabilities. For any $\mathbf{X} \in O^n$ and any $o_i \in O$:

$P(o_i | \mathbf{X}\lambda)$: The probability of o_i in the next interaction given a past history of \mathbf{X} ; and,

$P(\mathbf{X} | \lambda)$: The a priori probability of observing sequence \mathbf{X} in the model.

- An algorithm \mathcal{A} which takes as input a history $\mathbf{X} \in O^n$ (for any n) and outputs a distribution on O ,

$$\mathcal{A}(o_i | \mathbf{X}) \in [0, 1] \text{ (for all } i), \quad \sum_{i=1}^m \mathcal{A}(o_i | \mathbf{X}) = 1.$$

Simple Probabilistic Systems

Let us define a simple probabilistic system for computational trust as given by the following:

- A (finite) set of possible outcomes of each interaction
 $O = \{o_1, o_2, \dots, o_m\}$.
- A *probabilistic model*, λ , of principal behavior. Model λ defines the following probabilities. For any $\mathbf{X} \in O^n$ and any $o_i \in O$:

$P(o_i \mid \mathbf{X}\lambda)$: The probability of o_i in the next interaction given a past history of \mathbf{X} ; and,

$P(\mathbf{X} \mid \lambda)$: The a priori probability of observing sequence \mathbf{X} in the model.

- An algorithm \mathcal{A} which takes as input a history $\mathbf{X} \in O^n$ (for any n) and outputs a distribution on O ,

$$\mathcal{A}(o_i \mid \mathbf{X}) \in [0, 1] \text{ (for all } i), \quad \sum_{i=1}^m \mathcal{A}(o_i \mid \mathbf{X}) = 1.$$

Simple Probabilistic Systems

Let us define a simple probabilistic system for computational trust as given by the following:

- A (finite) set of possible outcomes of each interaction
 $O = \{o_1, o_2, \dots, o_m\}$.
- A *probabilistic model*, λ , of principal behavior. Model λ defines the following probabilities. For any $\mathbf{X} \in O^n$ and any $o_i \in O$:

$P(o_i \mid \mathbf{X}\lambda)$: The probability of o_i in the next interaction given a past history of \mathbf{X} ; and,

$P(\mathbf{X} \mid \lambda)$: The a priori probability of observing sequence \mathbf{X} in the model.

- An algorithm \mathcal{A} which takes as input a history $\mathbf{X} \in O^n$ (for any n) and outputs a distribution on O ,

$$\mathcal{A}(o_i \mid \mathbf{X}) \in [0, 1] \text{ (for all } i), \quad \sum_{i=1}^m \mathcal{A}(o_i \mid \mathbf{X}) = 1.$$

The Kullback-Leibler Divergence

A “pseudo-distance” on probability distributions

Consider the following binary function D_{KL} , the Kullback-Leibler divergence, on two distribution $\hat{p}, \hat{q} : \mathcal{O} \rightarrow [0, 1]$.

$$D_{\text{KL}}(\hat{p} \parallel \hat{q}) = \sum_{i=1}^m p_i \log_2\left(\frac{p_i}{q_i}\right)$$

- Established measure in statistics for comparing the “distance” from a “true” distribution to an “approximation” of that distribution.
- Information-theoretic interpretation: The amount of information one would gain if one would know \hat{p} instead of \hat{q} .

The Kullback-Leibler Divergence

A “pseudo-distance” on probability distributions

Consider the following binary function D_{KL} , the Kullback-Leibler divergence, on two distribution $\hat{p}, \hat{q} : \mathcal{O} \rightarrow [0, 1]$.

$$D_{\text{KL}}(\hat{p} \parallel \hat{q}) = \sum_{i=1}^m p_i \log_2\left(\frac{p_i}{q_i}\right)$$

- Established measure in statistics for comparing the “distance” from a “true” distribution to an “approximation” of that distribution.
- Information-theoretic interpretation: The amount of information one would gain if one would know \hat{p} instead of \hat{q} .

The Expected Kullback-Leibler Divergence

A measure on probabilistic trust algorithms

- The goal of a probabilistic trust-based algorithm, say \mathcal{A} , is to best approximate a distribution on the outcomes (o_1, \dots, o_m) given a history \mathbf{X} .
- “to best approximates” now means to minimize the Kullback-Leibler Divergence.
- However, different inputs \mathbf{X} result in different output distributions $\mathcal{A}(\cdot | \mathbf{X})$.

Define the n 'th **expected** Kullback-Leibler divergence from λ to \mathcal{A}

$$ED_{\text{KL}}^n(\lambda \parallel \mathcal{A}) = \sum_{\mathbf{X} \in \mathcal{O}^n} P(\mathbf{X} | \lambda) \cdot D_{\text{KL}}(P(\cdot | \mathbf{X}\lambda) \parallel \mathcal{A}(\cdot | \mathbf{X}))$$

The Expected Kullback-Leibler Divergence

A measure on probabilistic trust algorithms

- The goal of a probabilistic trust-based algorithm, say \mathcal{A} , is to best approximate a distribution on the outcomes (o_1, \dots, o_m) given a history \mathbf{X} .
- “to best approximates” now means to minimize the Kullback-Leibler Divergence.
- However, different inputs \mathbf{X} result in different output distributions $\mathcal{A}(\cdot | \mathbf{X})$.

Define the n 'th **expected** Kullback-Leibler divergence from λ to \mathcal{A}

$$ED_{\text{KL}}^n(\lambda \parallel \mathcal{A}) = \sum_{\mathbf{X} \in \mathcal{O}^n} P(\mathbf{X} | \lambda) \cdot D_{\text{KL}}(P(\cdot | \mathbf{X}\lambda) \parallel \mathcal{A}(\cdot | \mathbf{X}))$$

A Range of Algorithms

What if we consider an approximation \mathcal{A}_ϵ of \mathcal{A}_0 given by

$$\mathcal{A}_\epsilon(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h) + \epsilon}{|h| + 2\epsilon}, \quad \mathcal{A}_\epsilon(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h) + \epsilon}{|h| + 2\epsilon}$$

(for some (typically) small $\epsilon \in [0, 1]$).


Corollary

Consider the $\lambda_{\mathbf{B}}$ model with $\theta \in [\frac{1}{2} - \frac{1}{\sqrt{12}}, \frac{1}{2} + \frac{1}{\sqrt{12}}]$. For any $n \geq 0$ we have

$$\text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_1) < \text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$$

For all $\epsilon \in [0, 1)$.

What does this mean?

If $\theta \in [.215, .785]$ then (on average) the algorithm of Mui et al. computes a better approximation of θ than any algorithm \mathcal{A}_ϵ for $\epsilon \in [0, 1)$. 

A Range of Algorithms

What if we consider an approximation \mathcal{A}_ϵ of \mathcal{A}_0 given by

$$\mathcal{A}_\epsilon(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h) + \epsilon}{|h| + 2\epsilon}, \quad \mathcal{A}_\epsilon(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h) + \epsilon}{|h| + 2\epsilon}$$

(for some (typically) small $\epsilon \in [0, 1]$).


Corollary

Consider the $\lambda_{\mathbf{B}}$ model with $\theta \in [\frac{1}{2} - \frac{1}{\sqrt{12}}, \frac{1}{2} + \frac{1}{\sqrt{12}}]$. For any $n \geq 0$ we have

$$\text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_1) < \text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$$

For all $\epsilon \in [0, 1]$.

What does this mean?

If $\theta \in [.215, .785]$ then (on average) the algorithm of Mui et al. computes a better approximation of θ than any algorithm \mathcal{A}_ϵ for $\epsilon \in [0, 1]$. 

A Range of Algorithms

What if we consider an approximation \mathcal{A}_ϵ of \mathcal{A}_0 given by

$$\mathcal{A}_\epsilon(\mathbf{s} \mid h) = \frac{N_{\mathbf{s}}(h) + \epsilon}{|h| + 2\epsilon}, \quad \mathcal{A}_\epsilon(\mathbf{f} \mid h) = \frac{N_{\mathbf{f}}(h) + \epsilon}{|h| + 2\epsilon}$$

(for some (typically) small $\epsilon \in [0, 1]$).


Corollary

Consider the $\lambda_{\mathbf{B}}$ model with $\theta \in [\frac{1}{2} - \frac{1}{\sqrt{12}}, \frac{1}{2} + \frac{1}{\sqrt{12}}]$. For any $n \geq 0$ we have

$$\text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_1) < \text{ED}_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$$

For all $\epsilon \in [0, 1]$.

What does this mean?

If $\theta \in [.215, .785]$ then (on average) the algorithm of Mui et al. computes a better approximation of θ than any algorithm \mathcal{A}_ϵ for $\epsilon \in [0, 1]$. 

More...

- Algorithms \mathcal{A}_1 and \mathcal{A}_0 have been proposed in the literature.
- Why not \mathcal{A}_ϵ for other positive reals ϵ ? What is best? How does this depend on θ and on n ?

Theorem

For any $\theta \in [0, 1]$ with $\theta \neq \frac{1}{2}$ there exists an $\bar{\epsilon} \in [0, \infty)$ that minimizes $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ (simultaneously for all n) as an expression in ϵ . Further, $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ is decreasing as a function of ϵ on the interval $(0, \bar{\epsilon})$, and increasing on $(\bar{\epsilon}, \infty)$.

What does this mean? Unless behavior is completely random, then for each behavior θ there is a unique best \mathcal{A}_ϵ algorithm, which outperforms all other \mathcal{A}_ϵ algorithms (for that θ). Further, this is independent of n .

For $\theta = \frac{1}{2}$ the larger the ϵ the better.

More...

- Algorithms \mathcal{A}_1 and \mathcal{A}_0 have been proposed in the literature.
- Why not \mathcal{A}_ϵ for other positive reals ϵ ? What is best? How does this depend on θ and on n ?

Theorem

For any $\theta \in [0, 1]$ with $\theta \neq \frac{1}{2}$ there exists an $\bar{\epsilon} \in [0, \infty)$ that minimizes $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ (simultaneously for all n) as an expression in ϵ . Further, $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ is decreasing as a function of ϵ on the interval $(0, \bar{\epsilon})$, and increasing on $(\bar{\epsilon}, \infty)$.

What does this mean? Unless behavior is completely random, then for each behavior θ there is a unique best \mathcal{A}_ϵ algorithm, which outperforms all other \mathcal{A}_ϵ algorithms (for that θ). Further, this is independent of n .

For $\theta = \frac{1}{2}$ the larger the ϵ the better.

More...

- Algorithms \mathcal{A}_1 and \mathcal{A}_0 have been proposed in the literature.
- Why not \mathcal{A}_ϵ for other positive reals ϵ ? What is best? How does this depend on θ and on n ?

Theorem

For any $\theta \in [0, 1]$ with $\theta \neq \frac{1}{2}$ there exists an $\bar{\epsilon} \in [0, \infty)$ that minimizes $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ (simultaneously for all n) as an expression in ϵ . Further, $ED_{\text{KL}}^n(\lambda_{\mathbf{B}} \parallel \mathcal{A}_\epsilon)$ is decreasing as a function of ϵ on the interval $(0, \bar{\epsilon})$, and increasing on $(\bar{\epsilon}, \infty)$.

What does this mean? Unless behavior is completely random, then for each behavior θ there is a unique best \mathcal{A}_ϵ algorithm, which outperforms all other \mathcal{A}_ϵ algorithms (for that θ). Further, this is independent of n .

For $\theta = \frac{1}{2}$ the larger the ϵ the better.

Further

- We know what $\bar{\epsilon}$ looks like: $\bar{\epsilon} = 2 \frac{\theta(\theta-1)}{(2\theta-1)^2}$.
- Algorithm \mathcal{A}_ϵ is optimal (among such algorithms) for values $\theta = \frac{1}{2} \pm \frac{1}{2\sqrt{2\epsilon+1}}$.
- Algorithm \mathcal{A}_1 is optimal for $\theta = \frac{1}{2} - \frac{1}{\sqrt{12}}$ and for $\theta = \frac{1}{2} + \frac{1}{\sqrt{12}}$.
- Algorithm \mathcal{A}_0 is optimal for $\theta = 0$ and for $\theta = 1$.

Further

- We know what $\bar{\epsilon}$ looks like: $\bar{\epsilon} = 2 \frac{\theta(\theta-1)}{(2\theta-1)^2}$.
- Algorithm \mathcal{A}_ϵ is optimal (among such algorithms) for values $\theta = \frac{1}{2} \pm \frac{1}{2\sqrt{2\epsilon+1}}$.
- Algorithm \mathcal{A}_1 is optimal for $\theta = \frac{1}{2} - \frac{1}{\sqrt{12}}$ and for $\theta = \frac{1}{2} + \frac{1}{\sqrt{12}}$.
- Algorithm \mathcal{A}_0 is optimal for $\theta = 0$ and for $\theta = 1$.

Further

- We know what $\bar{\epsilon}$ looks like: $\bar{\epsilon} = 2 \frac{\theta(\theta-1)}{(2\theta-1)^2}$.
- Algorithm \mathcal{A}_ϵ is optimal (among such algorithms) for values $\theta = \frac{1}{2} \pm \frac{1}{2\sqrt{2\epsilon+1}}$.
- Algorithm \mathcal{A}_1 is optimal for $\theta = \frac{1}{2} - \frac{1}{\sqrt{12}}$ and for $\theta = \frac{1}{2} + \frac{1}{\sqrt{12}}$.
- Algorithm \mathcal{A}_0 is optimal for $\theta = 0$ and for $\theta = 1$.

Outline

- 1 A Concrete Model of Behavioral Information
- 2 A Logical Approach to Experience-based Trust Management
 - Intermezzo: Short Demo of JavaHBAC
- 3 From Simulations to Theorems?
 - Probabilistic examples
 - Results for (simple) probabilistic systems
- 4 The Future?

Future Work?

- Recall $\lambda_{\mathbf{B}}$ (behavior of principals is fixed once and for all, given by $\theta \in [0, 1]$).
 - ▶ Unrealistic in some scenarios: No dynamic behavior-change.
- Consider the algorithm of Mui et al., which remembers two parameters $\alpha =$ “no. of s” and $\beta =$ “no. of f”.
- To cope with dynamic behavior, it has been proposed to exponentially “decay” α and β each time they are updated.
- To cope with dynamic behavior, it has been proposed to store only the past K interactions (finite memory).

Question

Which algorithm, exponential decay or finite memory, best approximates various formalizations of extensions of $\lambda_{\mathbf{B}}$ to dynamic principal behavior?

Future Work?

- Recall $\lambda_{\mathbf{B}}$ (behavior of principals is fixed once and for all, given by $\theta \in [0, 1]$).
 - ▶ Unrealistic in some scenarios: No dynamic behavior-change.
- Consider the algorithm of Mui et al., which remembers two parameters $\alpha =$ “no. of \mathbf{s} ” and $\beta =$ “no. of \mathbf{f} ”.
- To cope with dynamic behavior, it has been proposed to exponentially “decay” α and β each time they are updated.
- To cope with dynamic behavior, it has been proposed to store only the past K interactions (finite memory).

Question

Which algorithm, exponential decay or finite memory, best approximates various formalizations of extensions of $\lambda_{\mathbf{B}}$ to dynamic principal behavior?

Future Work?

- Recall $\lambda_{\mathbf{B}}$ (behavior of principals is fixed once and for all, given by $\theta \in [0, 1]$).
 - ▶ Unrealistic in some scenarios: No dynamic behavior-change.
- Consider the algorithm of Mui et al., which remembers two parameters $\alpha =$ “no. of \mathbf{s} ” and $\beta =$ “no. of \mathbf{f} ”.
- To cope with dynamic behavior, it has been proposed to exponentially “decay” α and β each time they are updated.
- To cope with dynamic behavior, it has been proposed to store only the past K interactions (finite memory).

Question

Which algorithm, exponential decay or finite memory, best approximates various formalizations of extensions of $\lambda_{\mathbf{B}}$ to dynamic principal behavior?

Future Work?

- Recall $\lambda_{\mathbf{B}}$ (behavior of principals is fixed once and for all, given by $\theta \in [0, 1]$).
 - ▶ Unrealistic in some scenarios: No dynamic behavior-change.
- Consider the algorithm of Mui et al., which remembers two parameters $\alpha =$ “no. of \mathbf{s} ” and $\beta =$ “no. of \mathbf{f} ”.
- To cope with dynamic behavior, it has been proposed to exponentially “decay” α and β each time they are updated.
- To cope with dynamic behavior, it has been proposed to store only the past K interactions (finite memory).

Question

Which algorithm, exponential decay or finite memory, best approximates various formalizations of extensions of $\lambda_{\mathbf{B}}$ to dynamic principal behavior?

Future Work

- Our measure is parametric in the model, so it is possible to apply it to Hidden Markov Models.
- We would like to examine the question by considering

$$ED_{\text{KL}}^n(\lambda_{\text{HMM}} \parallel \mathcal{A}_{\text{decay}}) \quad \text{vs.} \quad ED_{\text{KL}}^n(\lambda_{\text{HMM}} \parallel \mathcal{A}_{\text{memory}})$$

(for various representative λ_{HMM} 's).

Dissemination

Key Publications

- Logical Approach to Experience-based Trust.
 - ▶ K. Krukow, M. Nielsen, V. Sassone. **A Logical Framework for Reputation Systems**. Submitted to Journal of Computer Security.
 - ▶ K. Krukow, M. Nielsen, V. Sassone. **A Framework for Concrete Reputation-Systems with Applications to History-based Access Control**. In proceedings from 12th ACM Conference on Computer and Communications Security (CCS'05), pages 260-269, ACM Press.
- Probabilistic approaches.
 - ▶ K. Krukow and M. Nielsen. From Simulations to Theorems: **A Position Paper on Research in the Field of Computational Trust**. In Proceedings from FAST 2006 (in preparation).
 - ▶ K. Krukow and M. Nielsen. **On the Formal Modelling of Trust in Reputation-based Systems**. in "Theory is Forever: Essays Dedicated to Arto Salomaa", Springer LNCS, 3113, pp. 192–204.
- Trust Structures: Operational and Denotational Semantics.
 - ▶ K. Krukow and M. Nielsen. **Trust Structures: Denotational and Operational Semantics**. Accepted for publication in the International Journal of Information Security.
 - ▶ K. Krukow and A. Twigg. **Distributed Approximation of Fixed-Points in Trust Structures**. In proceedings of the 25th International Conference on Distributed Computing Systems (ICDCS'05), pages 805–814, IEEE.