Monitoring and Enforcing Data-Usage Policies

Felix Klaedtke

Institute of Information Security, ETH Zurich
Modern Problems

financial information
medical data
social network data
calendar documents emails...

regulatory compliance
privacy policies
corporate governance
conflict of interest...

How do we address such concerns?
Modern Problems

How do we address such concerns?
At the Core

- Controlling access
  “My medical data should only be accessible to my care givers”

- Controlling usage
  “…and used for intended purpose, e.g., improving my health care.”

- Implement controls to reduce risks

- For IT systems:
  processes that monitor and control how other processes access and use data
At the Core

- Controlling access
  “My medical data should only be accessible to my care givers”

- Controlling usage
  “...and used for intended purpose, e.g., improving my health care.”

- Implement controls to reduce risks

- For IT systems:
  processes that monitor and control how other processes access and use data

Core problems are theoretically interesting! So are system aspects!
  Challenges: generality, scalability, automation, usability, ...
Setting: policies stipulating data usage and agent behavior
HIPAA, SOX, separation of duty, etc.

Objective: prevent / detect policy violations
Setting: policies stipulating data usage and agent behavior
HIPAA, SOX, separation of duty, etc.

Objective: prevent / detect policy violations
Overview

Enforcement  Policy Specification  Monitoring

Temporal Reasoning, Mathematical Logic
Overview

Enforcement

Policy Specification

Monitoring

Temporal Reasoning, Mathematical Logic
Security Policies Come in all Shapes and Sizes

History-based Access Control
Chinese Wall
Separation of Duty
Information Flow
Business Regulations
Data Usage
Privacy

... Which are the enforceable ones?
Enforcement by Execution Monitoring

Enforceable Security Policies
Fred B. Schneider, ACM Trans. Inf. Syst. Sec., 2000

Abstract Setting

- System iteratively executes actions
- Enforcement mechanism intercepts them (prior to their execution)
- Enforcement mechanism terminates system in case of violation

system

allowed action?

enforcement mechanism
Enforcement by Execution Monitoring

Enforceable Security Policies
Fred B. Schneider, ACM Trans. Inf. Syst. Sec., 2000

Abstract Setting

- System iteratively executes actions
- Enforcement mechanism intercepts them (prior to their execution)
- Enforcement mechanism terminates system in case of violation

Still limited understanding which classes of security policies are enforceable

enforceable policy $\iff$ safety property
Follow-Up Work

- SASI enforcement of security policies
  Ú. Erlingsson and F. Schneider, NSPW’99

- IRM enforcement of Java stack inspection
  Ú. Erlingsson and F. Schneider, S&P’00

- Access control by tracking shallow execution history
  P. Fong, S&P’04

- Edit automata: enforcement mechanisms for run-time security properties

- Computability classes for enforcement mechanisms

- Run-time enforcement of nonsafety policies

- A theory of runtime enforcement, with results
  J. Ligatti and S. Reddy, ESORICS’10

- Do you really mean what you actually enforced?
  N. Bielova and F. Massacci, Int. J. Inf. Secur., 2011

- Runtime enforcement monitors: composition, synthesis and enforcement abilities

- Service automata
  R. Gay, H. Mantel, and B. Sprick, FAST’11

- Cost-aware runtime enforcement of security policies
  P. Drábik, F. Martinelli, and C. Morisset, STM’12

- ...
# Refined Abstract Setting

<table>
<thead>
<tr>
<th>Actions</th>
<th>Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Set of actions</strong> $\Sigma = \mathcal{O} \cup \mathcal{C}$:</td>
<td><strong>Trace universe</strong> $\mathcal{U} \subseteq \Sigma^\infty$:</td>
</tr>
<tr>
<td>$\mathcal{O} = {\text{observable actions}}$</td>
<td>$\mathcal{U} \neq \emptyset$</td>
</tr>
<tr>
<td>$\mathcal{C} = {\text{controllable actions}}$</td>
<td>$\mathcal{U}$ prefix-closed</td>
</tr>
</tbody>
</table>

Example: request \cdot tick \cdot deliver \cdot tick \cdot tick \cdot request \cdot deliver \cdot tick \ldots \in \mathcal{U}
### Refined Abstract Setting

#### Actions

Set of actions $\Sigma = O \cup C$:

- $O = \{\text{observable actions}\}$
- $C = \{\text{controllable actions}\}$

#### Traces

Trace universe $U \subseteq \Sigma^\infty$:

- $U \neq \emptyset$
- $U$ prefix-closed

Example: $\text{request} \cdot \text{tick} \cdot \text{deliver} \cdot \text{tick} \cdot \text{tick} \cdot \text{request} \cdot \text{deliver} \cdot \text{tick} \ldots \in U$

### Requirements (on an Enforcement Mechanism)

- **Soundness**: prevent policy-violating traces
- **Transparency**: allow policy-compliant traces
- **Computability**: make decisions
Definition

\[ P \subseteq (O \cup C) \\infty \text{ is enforceable in } U \text{ def } \iff \exists \text{ DTM } M \text{ with } \]

1. \( \epsilon \in L(M) \) "M accepts the empty trace"
2. \( M \) halts on inputs in \( \text{trunc}(L(M)) \cdot (O \cup C) \) "M either permits or denies an intercepted action"
3. \( M \) accepts inputs in \( \text{trunc}(L(M)) \cdot O \) "M permits an intercepted observable action"
4. \( \text{limitclosure}(\text{trunc}(L(M))) \cap U = P \cap U \) "soundness (\( \subseteq \)) and transparency (\( \supseteq \))"
Formalization

**Definition**

\[ P \subseteq (O \cup C)^\infty \text{ is enforceable in } U \Downarrow \iff \exists \text{ DTM } M \]

1. \( \epsilon \in L(M) \) "M accepts the empty trace"
2. \( M \) halts on inputs in \((\text{trunc}(L(M)) \cdot (O \cup C)) \cap U \) "M either permits or denies an intercepted action"
3. \( M \) accepts inputs in \((\text{trunc}(L(M)) \cdot O) \cap U \) "M permits an intercepted observable action"
4. \( \text{limitclosure}(\text{trunc}(L(M))) \cap U = P \cap U \) "soundness (\subseteq) and transparency (\supseteq)"
Formalization

**Definition**

\[ P \subseteq (O \cup C)^\infty \text{ is enforceable in } U \iff \exists \text{ DTM } M \]

1. \( \varepsilon \in L(M) \)  
   "M accepts the empty trace"

2. \( M \) halts on inputs in \( \text{trunc}(L(M)) \cdot (O \cup C) \cap U \)  
   "M either permits or denies an intercepted action"

3. \( M \) accepts inputs in \( \text{trunc}(L(M)) \cdot O \cap U \)  
   "M permits an intercepted observable action"

4. \( \text{limitclosure}(\text{trunc}(L(M))) \cap U = P \cap U \)  
   "soundness (\( \subseteq \)) and transparency (\( \supseteq \))"
Formalization

system \rightarrow \text{action} \rightarrow a_{n+1}

\begin{center}
\begin{array}{c}
\text{enforcement mechanism} \\
\hline \\
a_1 & a_2 & \ldots & a_{n-1} & a_n & a_{n+1} \\
\hline \\
\text{DTM}
\end{array}
\end{center}

Definition

$P \subseteq (O \cup C)^\infty$ is enforceable in $U$ \iff exists DTM $M$ with

1. $\varepsilon \in L(M)$
   
   “$M$ accepts the empty trace”

2. $M$ halts on inputs in \( (\text{trunc}(L(M)) \cdot (O \cup C)) \cap U \)
   
   “$M$ either permits or denies an intercepted action”

3. $M$ accepts inputs in \( (\text{trunc}(L(M)) \cdot O) \cap U \)
   
   “$M$ permits an intercepted observable action”

4. $\text{limitclosure}(\text{trunc}(L(M))) \cap U = P \cap U$
   
   “soundness ($\subseteq$) and transparency ($\supseteq$)”
Examples

Setting

- Controllable actions: \( C = \{ \text{login, request, deliver} \} \)
- Observable actions: \( O = \{ \text{tick, fail} \} \)
- Set of actions: \( \Sigma = C \cup O \)
- Trace universe: \( U = \Sigma^* \cup (\Sigma^* \cdot \{ \text{tick} \})^\omega \)

Policies

1. “login must not happen within 3 time units after a fail”
   \( \Rightarrow \) enforceable
2. “each request must be followed by a deliver within 3 time units”
   \( \Rightarrow \) not enforceable
The Evolution of Safety

- L. Lamport, 1977: “A safety property is one which states that something bad will not happen.”
The Evolution of Safety

L. Lamport, 1977: “A safety property is one which states that something bad will not happen.”

B. Alpern and F. Schneider, 1985: A property \( P \subseteq \Sigma^\omega \) is \( \omega \)-safety if

\[
\forall \sigma \in \Sigma^\omega. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega. \sigma^{<i} \cdot \tau \notin P
\]
The Evolution of Safety

L. Lamport, 1977: “A safety property is one which states that something bad will not happen.”

B. Alpern and F. Schneider, 1985: A property $P \subseteq \Sigma^\omega$ is $\omega$-safety if
\[
\forall \sigma \in \Sigma^\omega. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega. \sigma^<i \cdot \tau \notin P
\]

Folklore: A property $P \subseteq \Sigma^\infty$ is $\infty$-safety if
\[
\forall \sigma \in \Sigma^\infty. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\infty. \sigma^<i \cdot \tau \notin P
\]
The Evolution of Safety

L. Lamport, 1977: “A safety property is one which states that something bad will not happen.”

B. Alpern and F. Schneider, 1985: A property $P \subseteq \Sigma^\omega$ is $\omega$-safety if

$$\forall \sigma \in \Sigma^\omega. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega. \sigma <^i \cdot \tau \notin P$$

Folklore: A property $P \subseteq \Sigma^\infty$ is $\infty$-safety if

$$\forall \sigma \in \Sigma^\infty. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\infty. \sigma <^i \cdot \tau \notin P$$

T. Henzinger, 1992: A property $P \subseteq \Sigma^\omega$ is safety in $U \subseteq \Sigma^\omega$

$$\forall \sigma \in U. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \forall \tau \in \Sigma^\omega. \sigma <^i \cdot \tau \notin P \cap U$$
Safety
(with Universe and Observables)

Intuition: “$P$ is safety in $U$ and bad things are not caused by an $O$”

Formalization: A property $P \subseteq \Sigma^\infty$ is $(U,O)$-safety if

$$\forall \sigma \in U. \sigma \notin P \rightarrow \exists i \in \mathbb{N}. \sigma <^i \notin \Sigma^* \cdot O \land \forall \tau \in \Sigma^\infty. \sigma <^i \cdot \tau \notin P \cap U$$
Safety
(with Universe and Observables)

Intuition: “$P$ is safety in $U$ and bad things are not caused by an $O$”

Formalization: A property $P \subseteq \Sigma^\infty$ is $(U,O)$-safety if
\[
\forall \sigma \in U. \sigma \notin P \implies \exists i \in \mathbb{N}. \sigma^{<i} \notin \Sigma^* \cdot O \land \forall \tau \in \Sigma^\infty. \sigma^{<i} \cdot \tau \notin P \cap U
\]

Liveness also generalizes to this setting
(“something good can happen in $U$ after actions not in $O$”)

Safety and Enforceability

Theorem

Let \( P \) be a property and \( U \) a trace universe with \( U \cap \Sigma^* \) decidable.

\( P \) is \((U, O)\)-enforceable \( \iff \)

1. \( P \) is \((U, O)\)-safety

2. \( \text{pre}_*(P \cap U) \) is a decidable set

3. \( \varepsilon \in P \)

Compare to Schneider’s work: only \( \implies \) for (1)

where \( U = \Sigma^\infty \) and \( O = \emptyset \)
Realizability of Enforcement Mechanisms

Fundamental Algorithmic Problems

Given a specification of a policy.
- Is it enforceable?
- If yes, can we synthesize an enforcement mechanism for it?
- With what complexity can we do so?

Some Results

Deciding if $P$ is $(U, O)$-enforceable when both $U$ and $P$ are given as
- PDAs is **undecidable**.
- FSAs is **PSPACE-complete**.
- LTL formulas is **PSPACE-complete**.
- MLTL formulas is **EXPSPACE-complete**.
Checking Enforceability and Safety
(PDA and FSA)

Checking Enforceability
Let $U$ and $P$ be given as PDAs or FSAs $A_U$ and $A_P$.

1. $\text{pre}^*(L(A_P) \cap L(A_U))$ is known to be decidable
2. check whether $\varepsilon \in L(A_P)$
3. check whether $L(A_P)$ is $(L(A_U), O)$-safety

Checking Safety
Let $U$ and $P$ be given as PDAs or FSAs $A_U$ and $A_P$.

- PDAs: undecidable in general
- FSAs: generalization of standard techniques
Checking Enforceability and Safety
(LTL and MLTL)

Checking Enforceability
Let $U$ and $P$ be given as LTL or MLTL formulas $\varphi_U$ and $\varphi_P$.
1. pre$_*$$(L(\varphi_P) \cap L(\varphi_U))$ is known to be decidable
2. check whether $\varepsilon \in L(\varphi_P)$
3. check whether $L(\varphi_P)$ is $(L(\varphi_U), O)$-safety

Checking Safety
Let $U$ and $P$ be given as LTL or MLTL formulas $\varphi_U$ and $\varphi_P$.
1. translate $\varphi_U$ and $\varphi_P$ into FSAs $A_U$ and $A_P$
2. use the results of the previous slide on $A_U$ and $A_P$
3. perform all these calculations on-the-fly
Beyond a Yes-No Answer

- If yes . . .
  synthesize an enforcement mechanism from $A_P$ and $A_U$

- If no . . .
  return a witness that illustrates why $P$ is not $(U, O)$-enforceable

- If no . . .
  return the maximal trace universe $V$ in which $P$ is $(V, O)$-enforceable
Overview

Enforcement

Policy Specification

Monitoring

Temporal Reasoning, Mathematical Logic
Example

Consider a financial or research institute
* Employees write and publish reports
* Reports may contain confidential data

Report-must-be-approved policy

1. Reports must be approved before they are published.
2. Approvals must happen at most 10 days before publication.
3. The employees’ managers must approve the reports.

IT system logs events

<table>
<thead>
<tr>
<th>Date</th>
<th>Event</th>
<th>Author</th>
<th>Report ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-03-03</td>
<td>publish_report</td>
<td>Charlie</td>
<td>#234</td>
</tr>
<tr>
<td>2013-03-04</td>
<td>archive_report</td>
<td>Alice</td>
<td>#104</td>
</tr>
<tr>
<td>2013-03-09</td>
<td>approve_report</td>
<td>Alice</td>
<td>#248</td>
</tr>
<tr>
<td>2013-03-13</td>
<td>publish_report</td>
<td>Bob</td>
<td>#248</td>
</tr>
</tbody>
</table>

Is system trace policy compliant?
Policy Elements

1. Reports must be approved before they are published.
2. Approvals must happen at most 10 days before publication.
3. The employees’ managers must approve the reports.
Policy Elements

Subjects
- reports and employees
- unbounded over time

1. **Reports must be approved before they are published.**

2. **Approvals must happen at most 10 days before publication.**

3. **The employees’ managers must approve the reports.**
Policy Elements

Subjects
- reports and employees
- unbounded over time

1. Reports must be approved before they are published.

2. Approvals must happen at most 10 days before publication.

3. The employees’ managers must approve the reports.

Temporal aspects
- qualitative: before and always
- quantitative: at most 10 days
Policy Elements

Subjects
- reports and employees
- unbounded over time

Event predicates
- approving and publishing a report
- happen at a point in time
- logged with time-stamp

1. Reports must be approved before they are published.
2. Approvals must happen at most 10 days before publication.
3. The employees’ managers must approve the reports.

Temporal aspects
- qualitative: before and always
- quantitative: at most 10 days
Policy Elements

Subjects
- reports and employees
- unbounded over time

Event predicates
- approving and publishing a report
- happen at a point in time
- logged with time-stamp

1. Reports must be approved before they are published.

2. Approvals must happen at most 10 days before publication.

3. The employees’ managers must approve the reports.

Temporal aspects
- qualitative: before and always
- quantitative: at most 10 days

State predicates
- being someone’s manager
- have a duration
Widely used for reasoning about behavior of nonterminating system
  * Amir Pnueli: *The Temporal Logic of Programs*, FOCS 1977

Zoo of temporal logics: LTL, CTL, CTL*, PSL, ITL, MTL, TPTL, ...
  * Underlying time models differ
    See [Alur & Henzinger, 1992] for an overview and comparison
  * Dedicated temporal operators
  * Restricts temporal reasoning to a few cases
Linear-Time Temporal Logic

At each time point $i \in \mathbb{N}$, a proposition $P$ is either true or false.

Previous and Next

Once and Eventually (including present)

Historically and Generally (including present)
Linear-Time Temporal Logic

At each time point $i \in \mathbb{N}$, a proposition $P$ is either true or false
At each time point \( i \in \mathbb{N} \), a proposition \( P \) is either true or false

- **Previous** and **Next**

\[
\begin{align*}
\bullet P & \quad P \\
\circ P & \quad P
\end{align*}
\]
Linear-Time Temporal Logic

- At each time point $i \in \mathbb{N}$, a proposition $P$ is either true or false
- **Previous** and **Next**
  - $\bullet P \rightarrow P$
  - $\bigcirc P \rightarrow P$
- **Once** and **Eventually** (including **present**)
  - $\blacklozenge P \rightarrow P$
  - $\Diamond P \rightarrow P$
Linear-Time Temporal Logic

At each time point $i \in \mathbb{N}$, a proposition $P$ is either true or false.

- **Previous** and **Next**
  - $\blacklozenge P$
  - $\diamondsuit P$

- **Once** and **Eventually** (including present)
  - $\blacklozenge P$
  - $\diamondsuit P$

- **Historically** and **Generally** (including present)
  - $\blacksquare P$
  - $\square P$

past \hspace{1cm} present \hspace{1cm} future
Temporal operators: **Since** and **Until**

**Example:**

\[
\square \text{access} \rightarrow \Box \text{login} \neg (\neg \text{login} \mathbin{\mathbf{U}} \text{access} \land \neg \text{login})
\]

\[
\square \text{access} \rightarrow \neg \text{logout} \quad \text{S} \text{login}
\]

"A user is not allowed to access a file before he has not logged in"
Since and Until

Temporal operators: Since and Until

Examples:

\[ \Box access \rightarrow \lozenge login \]
\[ \Box access \rightarrow \neg logout \mathbf{S} login \]

\[ \neg (\neg login \mathbf{U} access \wedge \neg login) \]

“a user is not allowed to access a file before he has not logged in”
Metric Temporal Operators

- Each time point $i \in \mathbb{N}$ is **timestamped** $\tau_i \in \mathbb{N}$
  - *monotonically increasing*: for all $i \in \mathbb{N}$, $\tau_i \leq \tau_{i+1}$
  - *progressing*: for every $\kappa \in \mathbb{N}$, there is some $i \in \mathbb{N}$ such that $\tau_i > \kappa$

- Attach timing constraints to temporal operators

$$\text{past} \quad \text{present} \quad \text{future}$$

$$\tau_0 \quad \tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \quad \tau_5 \quad \tau_6 \quad \tau_7 \quad \tau_8 \quad \tau_9 \quad \tau_{10} \quad \ldots$$

$$\tau \quad 0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \quad 9 \quad 10 \quad \ldots$$

- **$\leq 10$** $P$

$$\tau_0 \quad \tau_1 \quad \tau_2 \quad \tau_3 \quad \tau_4 \quad \tau_5$$

$$\tau_4 - \tau_1 \leq 10$$
Propositional MTL

**Syntax:**  
\[ \phi ::= P \mid \neg \phi \mid \phi \lor \psi \mid \bigcirc I \phi \mid \bigcirc I \phi \mid \phi S_I \phi \mid \phi U_I \phi \]

**Semantics:**  
\[ \bar{D} = (D_0, D_1, \ldots) \text{ with } D_0, \ldots \subseteq AP, \bar{\tau} = (\tau_0, \tau_1, \ldots), \text{ and } i \in \mathbb{N} \]

\[
\begin{align*}
(\bar{D}, \bar{\tau}, i) \models P & \quad \text{iff} \quad P \in D_i \\
(\bar{D}, \bar{\tau}, i) \models \neg \phi & \quad \text{iff} \quad (\bar{D}, \bar{\tau}, i) \not\models \phi \\
(\bar{D}, \bar{\tau}, i) \models \phi \lor \psi & \quad \text{iff} \quad (\bar{D}, \bar{\tau}, i) \models \phi \text{ or } (\bar{D}, \bar{\tau}, i) \models \psi \\
(\bar{D}, \bar{\tau}, i) \models \bigcirc I \phi & \quad \text{iff} \quad i > 0, \tau_i - \tau_{i-1} \in I, \text{ and } (\bar{D}, \bar{\tau}, i-1) \models \phi \\
(\bar{D}, \bar{\tau}, i) \models \bigcirc I \phi & \quad \text{iff} \quad \tau_{i+1} - \tau_i \in I \text{ and } (\bar{D}, \bar{\tau}, i+1) \models \phi \\
(\bar{D}, \bar{\tau}, i) \models \phi S_I \psi & \quad \text{iff} \quad \text{there is some } j \leq i \text{ with } \tau_i - \tau_j \in I, \quad (\bar{D}, \bar{\tau}, j) \models \psi, \quad \text{and } (\bar{D}, \bar{\tau}, k) \models \phi, \text{ for all } k \text{ with } j < k \leq i \\
(\bar{D}, \bar{\tau}, i) \models \phi U_I \psi & \quad \text{iff} \quad \text{there is some } j \geq i \text{ with } \tau_j - \tau_i \in I, \quad (\bar{D}, \bar{\tau}, j) \models \psi, \quad \text{and } (\bar{D}, \bar{\tau}, k) \models \phi, \text{ for all } k \text{ with } i \leq k < i
\end{align*}
\]

**Syntactic Sugar:**  
\[ \Diamond_I \phi := \text{true} S_I \phi, \quad \Box_I \phi := \neg \Diamond_I \neg \phi, \ldots \]
Remarks on Time Model

- Zoo of temporal logics: CTL, LTL, PSL, ITL, MTL, TPTL, ...
  - Underlying time models differ [Alur&Henzinger '92]

- Why time-points with time-stamps?
  - Event-based view
  - Temporal reasoning with points is “simpler” than with intervals
  - State predicates can often be mimicked with the $S$ operator

- Why a discrete time domain?
  - Clocks have limited precision
  - Minor impact on monitoring

- Linear time versus branching time
  - In monitoring, we observe a single trace
Policy Specification Language
Metric First-Order Temporal Logic [Koymans, 1990]

\[ \Box \forall e. \forall r. \text{publish\_report}(e, r) \rightarrow \]
\[ \Diamond_{\leq 10} \exists m. \text{manager}(m, e) \land \text{approve\_report}(m, r) \]

- **First-order** for expressing relations on data
- **Temporal operators** for reasoning about time
- **Metric information** adds timing constraints
Syntax

- A **signature** $S$ is a tuple $(C, R)$
  - $C$ is a finite set of **constant symbols** and $R$ is a finite set of **predicates**, each with an associated arity

- **(MFOTL) formulas** over a signature $S$ and set of variables $V$
  
  $\phi ::= t_1 \approx t_2 \mid t_1 \prec t_2 \mid r(t_1, \ldots, t_n) \mid \neg \phi \mid \phi \lor \phi \mid \exists x. \phi \mid \bigcirc l \phi \mid \bigodot l \phi \mid \phi S I \phi \mid \phi U I \phi$

  where $I$ is an **interval** of $\mathbb{N}$

- **Syntactic sugar** like $\square I \phi$ for $\neg (true U I \neg \phi)$ and $\square \phi$ for $\square_{[0,\infty)} \phi$
Semantics

A *temporal structure* (over $S$) is a pair $(\bar{D}, \bar{\tau})$.

- Sequence $\bar{\tau} = (\tau_0, \tau_1, \ldots)$ of timestamps, $\tau_i \in \mathbb{N}$ monotonically increasing and progressing
- Sequence of structures $\bar{D} = (D_0, D_1, \ldots)$ constant domains and rigid interpretation of constant symbols

$(\bar{D}, \bar{\tau}, \nu, i) \models \phi$ denotes *MFOTL satisfaction*

$(\bar{D}, \bar{\tau})$ is a temporal structure, $\nu$ a valuation, $i \in \mathbb{N}$, and $\phi$ a formula

Standard semantics for first-order part
Policy Revisited and Simplified

1. Reports must be approved before they are published.
2. Approvals must happen at most 10 days before publication.
3. The employees’ managers must approve the reports.

Publishing and approving events are logged with time-stamps

Simplified policy in MFOTL:

\[
\square \forall e. \forall r. \text{publish\_report}(e, r) \rightarrow \Diamond_{\leq 10} \exists m. \text{approve\_report}(m, r)
\]
Policy Revisited

1. Reports must be approved before they are published.
2. Approvals must happen at most 10 days before publication.
3. The employees’ managers must approve the reports.

- Being someone’s manager is a **state property**, with a duration
- Log events that mark **start** and **end** points

\[
\begin{align*}
\text{manager}_{\text{start}}(\text{Alice, Charlie}) & \quad \text{manager}_{\text{end}}(\text{Alice, Charlie}) \\
\text{manager}_{\text{start}}(\text{Alice, Bob}) &
\end{align*}
\]

- State predicate as syntactic sugar
  \[
  \text{manager}(m, e) = \neg \text{manager}_{\text{end}}(m, e) \land \text{manager}_{\text{start}}(m, e)
  \]

- Policy in MFOTL:
  \[
  \Box \forall e. \forall r. \text{publish\_report}(e, r) \rightarrow \\
  
  \Diamond_{\leq 10} \exists m. \text{manager}(m, e) \land \text{approve\_report}(m, r)
  \]
Transaction Requirements
Banking compliance à la Bank Secrecy or USA Patriot Act

- Requirements for monitoring, authorizing, and reporting large or suspicious transactions.

- Signature
  - Constant \( th \): a threshold “large” amount
  - \( \text{trans}(c, t, a) \): customer \( c \) carries out transaction \( t \) involving amount \( a \)
  - \( \text{auth}(e, t) \): employee \( e \) authorizes \( t \)
  - \( \text{report}(t) \): \( t \) is reported

- In general, signature determined by monitoring requirements and events that system actually provides.
Transaction Requirements (cont.)

- “Transactions of customers must be reported within 5 days when the transferred amount exceeds a given threshold”

\[ \square \forall c. \forall t. \forall a. \trans(c, t, a) \land th < a \rightarrow \Diamond \leq 5 \report(t) \]

- “Transactions exceeding the threshold must be authorized by employees (within 7 days) before execution”

\[ \square \forall c. \forall t. \forall a. \trans(c, t, a) \land th < a \rightarrow \lozenge \leq 7 \exists e. \auth(e, t) \]

- “Each transaction \( t \) of a customer \( c \), who has within the last 30 days been involved in a suspicious transaction \( t' \), must be reported as suspicious within 3 days”

\[ \square \forall c. \forall t. \forall a. \trans(c, t, a) \land (\lozenge \leq 30 \exists t'. \forall a'. \trans(c, t', a') \land \Diamond \leq 5 \report(t')) \rightarrow \Diamond \leq 3 \report(t) \]
Chinese Wall

- Policy to avoid conflict-of-interest situations

  "Subject s must not access object o when s has previously accessed another object in a different dataset than o and both datasets are in the same conflict-of-interest class"

- A possible formalization (with timing constraints):

  \[
  \Box \forall s. \forall o. \forall d. \forall d'. access(s, o) \land \text{dataset}(o, d) \land \\
  (\exists o'. (\bigtriangleup_{<4} access(s, o')) \land \text{dataset}(o', d')) \rightarrow \\
  \neg \text{conflict}(d, d')
  \]

  Assume that:
  
  * At each time point, \text{conflict} is irreflexive and symmetric
  * At each time point, \text{dataset} is a partial function from objects to datasets

- Different types of predicates:
  
  * Event predicate: accessing an object happens at a time point
  * State predicate: being in a dataset has a duration (start and finish)
  * Datasets and conflict-of-interest classes might change over time
Experience

MFOTL is well suited to formalize a wide range of policies

But:

- Precision must precede formalization
  - “Data must be securely stored.”

- Gap between high-level policies and system information
  - “Data must be deleted within 30 days.”
  - “Data should be used for statistical purposes only.”

- Not all policies are trace properties
  - “Average response time, over all executions, should be less than 10ms.”
  - “Actions of high users have no effect on observations of low users.”
Overview

Enforcement  Policy Specification  Monitoring

Temporal Reasoning, Mathematical Logic
Overview: Monitoring

I.) Monitoring Algorithm

II.) Case Study **NOKIA**

III.) Case Study **Google**
Monitoring Objective

- For a policy given as an **MFOTL formula** $\phi$

  $$\square \forall c. \forall t. \forall a. \text{trans}(c, t, a) \land th < a \rightarrow \diamond \leq_6 \text{report}(t)$$

- and a **prefix of a temporal structure** given by system events or logs

<table>
<thead>
<tr>
<th>trans</th>
<th>cID</th>
<th>tID</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>#34</td>
<td>#34</td>
<td>$100'000$</td>
</tr>
<tr>
<td>Eve</td>
<td>#37</td>
<td>#37</td>
<td>$1'000$</td>
</tr>
<tr>
<td>Eve</td>
<td>#45</td>
<td>#45</td>
<td>$999'999$</td>
</tr>
<tr>
<td>Mallory</td>
<td>#78</td>
<td>#78</td>
<td>$24$</td>
</tr>
<tr>
<td>Mallory</td>
<td>#99</td>
<td>#99</td>
<td>$333'333$</td>
</tr>
</tbody>
</table>

- monitor should **report all policy violations** (either online or offline)
Monitoring Objective

- For a policy given as an **MFOTL formula** $\phi$

\[
\square \forall c. \forall t. \forall a. \text{trans}(c, t, a) \land (\Diamond <31 \exists t'. \exists a'. t \not\approx t' \land \text{trans}(c, t', a') \land \Diamond <6 \text{report}(t')) \\
\rightarrow \\
\Diamond <3 \text{report}(t)
\]

- and a **prefix of a temporal structure** given by system events or logs

- monitor should **report all policy violations** (either online or offline)
For a policy given as an MFOTL formula $\phi$

\[ \square \forall c. \forall t. \forall a. \text{trans}(c, t, a) \land (\Diamond <_{31} \exists t'. \exists a'. t \not\approx t' \land \text{trans}(c, t', a') \land \Diamond <_{6} \text{report}(t')) \rightarrow \Diamond <_{3} \text{report}(t) \]

and a prefix of a temporal structure given by system events or logs

\[ \tau_0 \quad \tau_1 \quad \ldots \quad \tau_i \quad \ldots \]

<table>
<thead>
<tr>
<th>trans</th>
<th>cID</th>
<th>tID</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>#34</td>
<td>#34</td>
<td>$100'000</td>
</tr>
<tr>
<td>Eve</td>
<td>#37</td>
<td>#37</td>
<td>$1'000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>report</td>
<td>tID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>#34</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>trans</th>
<th>cID</th>
<th>tID</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eve</td>
<td>#45</td>
<td>#45</td>
<td>$999'999</td>
</tr>
<tr>
<td>Mallory</td>
<td>#78</td>
<td>#99</td>
<td>$24</td>
</tr>
<tr>
<td></td>
<td>#99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| report | tID  |      |            |
|        | #34  |      |            |

<table>
<thead>
<tr>
<th>trans</th>
<th>cID</th>
<th>tID</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>#78</td>
<td>#78</td>
<td>$333'333</td>
</tr>
<tr>
<td>Mallory</td>
<td>#99</td>
<td>#99</td>
<td>$24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| report | tID  |      |            |
|        |      |      |            |

monitor should report all policy violations (either online or offline)
Restrictions

Not every MFOTL-definable property can be effectively monitored on a temporal structure

- **Structures** $\mathcal{D}_0, \mathcal{D}_1, \ldots$ have only finite relations

- **Formula** $\phi$ must be of the form $\Box \phi'$
  * Temporal future operators in $\phi'$ only refer finitely into the future
    So $\phi$ describes a safety property [Lamport '77, Alpern&Schneider '85]
  * Further restrictions on $\phi'$ to guarantee finiteness of intermediate results

\[
\neg r(x) \land \neg q(x) \sim r(x) \land \neg \Box_{<7} q(x)
\]

Related to domain independence of database queries
(see, e.g., [Fagin, 1982])
Preprocessing: Negation and Rewriting

- Input formula $\phi$

$$\square \forall t. \forall c. \forall a. \text{trans}(t, c, a) \land (\Diamond \leq 31 \exists t'. \exists a'. t \not\approx t' \land \text{trans}(t', c, a') \land \Diamond \leq 6 \text{report}(t'))$$

$$\rightarrow$$

$$\Diamond \leq 3 report(t)$$
Preprocessing: Negation and Rewriting

- **Input formula** $\phi$

  $\square \forall t. \forall c. \forall a. \text{trans}(t, c, a) \land (\blacklozenge_{<31} \exists t'. \exists a'. t \not\approx t' \land \text{trans}(t', c, a') \land \blacklozenge_{<6} \text{report}(t'))$

  $\rightarrow$

  $\blacklozenge_{<3} \text{report}(t)$

- **Negate, rewrite, and drop outermost $\blacklozenge$ and $\exists$ quantifier(s), yielding $\psi$**

  $\neg \exists t. \exists c. \exists a. \text{trans}(t, c, a) \land (\blacklozenge_{<31} \exists t'. \exists a'. t \not\approx t' \land \text{trans}(t', c, a') \land \blacklozenge_{<6} \text{report}(t'))$

  $\land$

  $\neg \blacklozenge_{<3} \text{report}(t)$

For monitoring: for each $i \in \mathbb{N}$, determine elements satisfying $\psi$:

$\{\overline{a} \mid (\overline{D}, \overline{\tau}, v[\overline{x}/\overline{a}], i) = \psi\}$

These are the transactions that should have been reported at time-point $i$.41
Preprocessing: Negation and Rewriting

Input formula $\phi$

\[ \forall t. \forall c. \forall a. \text{trans}(t, c, a) \land (\lozenge_{<31} \exists t'. \exists a'. t \not\approx t' \land \text{trans}(t', c, a') \land \lozenge_{<6} \text{report}(t')) \rightarrow \lozenge_{<3} \text{report}(t) \]

Negate, rewrite, and drop outermost $\lozenge$ and $\exists$ quantifier(s), yielding $\psi$

\[ \neg \exists t. \exists c. \exists a. \text{trans}(t, c, a) \land (\lozenge_{<31} \exists t'. \exists a'. t \not\approx t' \land \text{trans}(t', c, a') \land \lozenge_{<6} \text{report}(t')) \land \neg \lozenge_{<3} \text{report}(t) \]

For monitoring: for each $i \in \mathbb{N}$, determine elements satisfying $\psi$:

\[ \{ \bar{a} \mid (\bar{D}, \bar{r}, v[\bar{x}/\bar{a}], i) \models \psi \} \]

These are the transactions that should have been reported at time-point $i$.
Preprocessing: Reduction to First-Order Queries

- For each temporal subformula $\alpha$ in $\psi$, introduce an auxiliary predicate $p_\alpha$

\[
\exists c. \exists a. \text{trans}(t, c, a) \land (\Diamond_{<3} \exists t'. \exists a'. \ldots \land \Diamond_{<6} \text{report}(t')) \land \neg \Diamond_{<3} \text{report}(t)
\]

- For each $i \in \mathbb{N}$, extend $D_i$ to $\hat{D}_i$, where for each temporal subformula $\alpha$:

\[
\hat{D}_i\alpha = \{ \overline{a} \mid (\hat{D}_i, \overline{\tau}, v[\overline{x}/\overline{a}], i) = \hat{\alpha} \}
\]
Preprocessing: Reduction to First-Order Queries

- For each temporal subformula $\alpha$ in $\psi$, introduce an auxiliary predicate $p_\alpha$

\[
\exists c. \exists a. \text{trans}(t, c, a) \land (\Diamond <_{31} \exists t'. \exists a' \ldots \land \Box <_{6} \text{report}(t')) \land \neg \Diamond <_{3} \text{report}(t)
\]

- Replace each $\alpha$ by a corresponding $p_\alpha$, yielding first-order formula $\hat{\psi}$

\[
\exists c. \exists a. \text{trans}(t, c, a) \land p_{\alpha_2}(c, t) \land \neg p_{\alpha_3}(t)
\]
Preprocessing: Reduction to First-Order Queries

- For each temporal subformula $\alpha$ in $\psi$, introduce an auxiliary predicate $p_\alpha$
  \[ \exists c. \exists a. \text{trans}(t, c, a) \land (\Box_{<3} \exists t'. \exists a' \ldots \land \Diamond_{<6} \text{report}(t')) \land \neg \Diamond_{<3} \text{report}(t) \]
  
  

- Replace each $\alpha$ by a corresponding $p_\alpha$, yielding first-order formula $\hat{\psi}$
  \[ \exists c. \exists a. \text{trans}(t, c, a) \land p_{\alpha_2}(c, t) \land \neg p_{\alpha_3}(t) \]

- For monitoring:
  - For each $i \in \mathbb{N}$, extend $D_i$ to $\hat{D}_i$, where for each temporal subformula $\alpha$
    \[ p_{\alpha_i}^{\hat{D}_i} = \{ \bar{a} \mid (\bar{x}, \bar{y}, v[\bar{x} / \bar{a}], i) \models \hat{\alpha} \} \]
  - For each $i \in \mathbb{N}$, query extended first-order structure $\hat{D}_i$
    \[ \{ \bar{a} \mid (\hat{D}_i, v[\bar{x} / \bar{a}]) \models \hat{\psi} \} \]
Preprocessing: Reduction to First-Order Queries

- For each temporal subformula $\alpha$ in $\psi$, introduce an auxiliary predicate $p_\alpha$

\[ \exists c. \exists a. \text{trans}(t, c, a) \land (\Box_{<31} \exists t'. \exists a'. \ldots \land \Diamond_{<6} \text{report}(t')) \land \neg \Diamond_{<3} \text{report}(t) \]

- Replace each $\alpha$ by a corresponding $p_\alpha$, yielding first-order formula $\hat{\psi}$

\[ \exists c. \exists a. \text{trans}(t, c, a) \land p_\alpha_2(c, t) \land \neg p_\alpha_3(t) \]

- For monitoring:
  * For each $i \in \mathbb{N}$, extend $D_i$ to $\hat{D}_i$, where for each temporal subformula $\alpha$

\[ p_{\hat{D}_i} = \{ \bar{a} \mid (\bar{D}, \bar{\tau}, \nu[\bar{x}/\bar{a}], i) \models \hat{\alpha} \} \]

  * For each $i \in \mathbb{N}$, query extended first-order structure $\hat{D}_i$

\[ \{ \bar{a} \mid (\hat{D}_i, \nu[\bar{x}/\bar{a}]) \models \hat{\psi} \} \]

Next: how to construct $p_{\hat{D}_i}$ for each $i \in \mathbb{N}$
Constructing the Auxiliary Relations

Construct auxiliary relations $p^{\hat{D}_i}_\alpha$ inductively over $\alpha$’s formula structure and using also relations from both previous and subsequent structures.

Case where $\alpha$ has form $\bullet_I \beta$:

$$p^{\hat{D}_i}_\alpha = \begin{cases} \hat{\beta}^{\hat{D}_{i-1}} & \text{if } i > 0 \text{ and } \tau_i - \tau_{i-1} \in I \\ \emptyset & \text{otherwise} \end{cases}$$

Case where $\alpha$ has form $\bigcirc_I \beta$:

$$p^{\hat{D}_i}_\alpha = \begin{cases} \hat{\beta}^{\hat{D}_{i+1}} & \text{if } \tau_{i+1} - \tau_i \in I \\ \emptyset & \text{otherwise} \end{cases}$$

* Construction depends on relations in $\hat{D}_{i+1}$ for which the predicates occur in $\hat{\beta}$
* Monitor constructs $p^{\hat{D}_i}_\alpha$ with a delay of at least one time step
Construction for $S_{[0,\infty)}$

- The construction for $\alpha = \beta S_{[0,\infty)} \gamma$ reflects the logical equivalence

$$\alpha \leftrightarrow \gamma \lor (\beta \land \Box \alpha)$$

- Assume that $\beta$ and $\gamma$ have the same free variables. Then

$$p_{\hat{\alpha}}^D_i = \gamma^D_i \cup \begin{cases} \emptyset & \text{if } i = 0 \\ \beta^D_i \cap p_{\hat{\alpha}}^{D_i-1} & \text{if } i > 0 \end{cases}$$

- Uses relations just for subformulas and previous time-point

- Constructions for metric $S_I$ and $U_I$ slightly more involved
Monitoring Algorithm

1: \( i \leftarrow 0 \) % lookahead index in sequence \((D_0, \tau_0), (D_1, \tau_1), \ldots \)
2: \( q \leftarrow 0 \) % index of next query evaluation in sequence \((D_0, \tau_0), (D_1, \tau_1), \ldots \)
3: \( Q \leftarrow \{(\alpha, 0, waitfor(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \} \)
4: loop
5: Carry over constants and relations of \( D_i \) to \( \hat{D}_i \).
6: for all \((\alpha, j, \emptyset) \in Q\) do % can build relation for \( \alpha \) in \( \hat{D}_j \)
7: Build auxiliary relation for \( \alpha \) in \( \hat{D}_j \).
8: Discard auxiliary relation for \( \alpha \) in \( \hat{D}_{j-1} \) if \( j - 1 \geq 0 \).
9: Discard relations \( p_{\delta}^{\hat{D}_j} \), where \( \delta \) is a temporal subformula of \( \alpha \).
10: while all relations \( p_{\alpha}^{\hat{D}_q} \) are built for \( \alpha \in tsub(\psi) \) do
11: Output violations \( \hat{\psi}^{\hat{D}_q} \) and time-stamp \( \tau_q \).
12: Discard structure \( \hat{D}_{q-1} \) if \( q > 0 \).
13: \( q \leftarrow q + 1 \)
14: \( Q \leftarrow \{(\alpha, i + 1, waitfor(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \} \cup \{(\alpha, j, \bigcup_{\alpha' \in update(S, \tau_{i+1} - \tau_i)} waitfor(\alpha')) \mid (\alpha, j, S) \in Q \text{ and } S \neq \emptyset \} \)
15: \( i \leftarrow i + 1 \) % process next element in input sequence \((D_{i+1}, \tau_{i+1})\)
16: end loop

Counters \( q \) (query) and \( i \) (lookahead) into input sequence
Monitoring Algorithm

1: $i \leftarrow 0$  \hspace{0.5cm} % lookahead index in sequence ($D_0, \tau_0), (D_1, \tau_1), \ldots$
2: $q \leftarrow 0$  \hspace{0.5cm} % index of next query evaluation in sequence ($D_0, \tau_0), (D_1, \tau_1), \ldots$
3: $Q \leftarrow \{(\alpha, 0, \text{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi\}$
4: loop
5: Carry over constants and relations of $D_i$ to $\hat{D}_i$.
6: for all $(\alpha, j, \emptyset) \in Q$ do  \hspace{0.5cm} % can build relation for $\alpha$ in $\hat{D}_j$
7: Build auxiliary relation for $\alpha$ in $\hat{D}_j$.
8: Discard auxiliary relation for $\alpha$ in $\hat{D}_{j-1}$ if $j - 1 \geq 0$.
9: Discard relations $p_{\hat{D}_j}^\delta$, where $\delta$ is a temporal subformula of $\alpha$.
10: while all relations $p_{\hat{D}_q}^\alpha$ are built for $\alpha \in \text{tsub}(\psi)$ do
11: Output violations $\hat{\psi}_{\hat{D}_q}$ and time-stamp $\tau_q$.
12: Discard structure $\hat{D}_{q-1}$ if $q > 0$.
13: $q \leftarrow q + 1$
14: $Q \leftarrow \{(\alpha, i + 1, \text{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi\} \cup$
   $\{(\alpha, j, \bigcup_{\alpha' \in \text{update}(S, \tau_{i+1} - \tau_i)} \text{waitfor}(\alpha')) \mid (\alpha, j, S) \in Q \text{ and } S \neq \emptyset\}$
15: $i \leftarrow i + 1$  \hspace{0.5cm} % process next element in input sequence ($D_{i+1}, \tau_{i+1}$)
16: end loop

$Q$ maintains list of unevaluated subformula $(\alpha, j, S)$ for past time-points
Monitoring Algorithm

1: \( i \leftarrow 0 \) \hspace{1cm} \% \text{ lookahead index in sequence } (D_0, \tau_0), (D_1, \tau_1), \ldots \n
2: \( q \leftarrow 0 \) \hspace{1cm} \% \text{ index of next query evaluation in sequence } (D_0, \tau_0), (D_1, \tau_1), \ldots \n
3: \( Q \leftarrow \{ (\alpha, 0, \text{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \} \n
4: \text{ loop} \n
5: \text{ Carry over constants and relations of } D_i \text{ to } \hat{D}_i. \n
6: \text{ for all } (\alpha, j, \emptyset) \in Q \text{ do} \hspace{1cm} \% \text{ can build relation for } \alpha \text{ in } \hat{D}_j \n
7: \text{ Build auxiliary relation for } \alpha \text{ in } \hat{D}_j. \n
8: \text{ Discard auxiliary relation for } \alpha \text{ in } \hat{D}_{j-1} \text{ if } j - 1 \geq 0. \n
9: \text{ Discard relations } p_{\delta}^{\hat{D}_j}, \text{ where } \delta \text{ is a temporal subformula of } \alpha. \n
10: \text{ while relations } p_{\alpha}^{\hat{D}_q} \text{ are built for all temporal subformulas } \alpha \text{ of } \psi \text{ do} \n
11: \text{ Output violations } \hat{\psi}^{\hat{D}_q} \text{ and time-stamp } \tau_q. \n
12: \text{ Discard structure } \hat{D}_{q-1} \text{ if } q > 0. \n
13: \( q \leftarrow q + 1 \n
14: \text{ } Q \leftarrow \{ (\alpha, i + 1, \text{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \} \cup \hspace{1cm} \% \text{ process next element in input sequence } (D_{i+1}, \tau_{i+1}) \n
15: \text{ } \{ (\alpha, j, \bigcup_{\alpha' \in \text{update}(S, \tau_{i+1}-\tau_i)} \text{waitfor}(\alpha')) \mid (\alpha, j, S) \in Q \text{ and } S \neq \emptyset \} \n
16: \text{ end loop} \n
Given relations for all temporal subformulas, output policy violations
Finite Relations

- In each iteration, monitor stores auxiliary relations

- **Problem**: must restrict negation and quantification
  - Consider the formula $p(x) \land \neg q(x)$
  - In $(i + 1)$st iteration, monitor constructs auxiliary relation $p \hat{D}_i \neg q(x)$

- **Solution**: rewrite to a formula so that auxiliary relations are finite
  - $p(x) \land \neg q(x)$ is rewritten to $p(x) \land (\neg q(x) \land \bigcirc p(x))$
  - Heuristic!
  - Related to domain independence of database queries, e.g., [Fagin ’82]
Finite Relations

In each iteration, monitor stores auxiliary relations

Problem: must restrict negation and quantification
- Consider the formula $p(x) \land \neg q(x)$
- In $(i + 1)$st iteration, monitor constructs auxiliary relation $p_i \hat{D} \neg q(x)$

Solution: rewrite to a formula so that auxiliary relations are finite
- $p(x) \land \neg q(x)$ is rewritten to $p(x) \land (\neg q(x) \land \bigcirc p(x))$
- Heuristic!
- Related to domain independence of database queries, e.g., [Fagin ’82]

Under reasonable assumptions, the size of the finite relations is polynomially bounded w.r.t. to input
Implementation of our monitoring algorithm for MFOTL

- Usage: monpoly -sig signature -formula policy -log logfile
- Output: policy violations

Open source, GNU public license

- Available at http://projects.developer.nokia.com/MonPoly
- Written in OCaml

Handles now also policies with aggregations:

$$\Box \forall u. \forall s. \left[ \text{SUM}_a a, t. \Diamond_{<31} \text{withdraw}(u, t, a) \right](s; u) \rightarrow s \leq 5000$$
Performance Evaluation

- Generated log files with different event rates for a fixed time span
- Monitoring performance for complex transaction-report policy:

- PostgreSQL does not scale to larger log files
Overview: Monitoring

I.) Monitoring Algorithm

II.) Case Study NOKIA

III.) Case Study Google
Nokia’s Data-Collection Campaign

- Phone data collected and propagated to databases:
  location, call and SMS info, accelerometer, ...

- Participants can view and delete their data

- Clear-text data used for personalized apps, e.g., location-history maps

- Anonymized data is used for research
1. Access-control rules restrict who accesses and modifies data in databases
   (A) Only user \textit{script2} may delete data from \textit{db2}
   (B) Databases \textit{db1} and \textit{db2} are accessed by \textit{script1} account only while \textit{script1} is running

2. Data changes are propagated between databases
   (C) Data deleted from \textit{db2} is deleted from \textit{db3} within 60 seconds
   (D) Data inserted into \textit{db1} is, within 30 hours, either inserted into \textit{db2} or deleted from \textit{db1}
Log entries are produced at multiple places

Need to combine logs

No total order on log entries

Compliance might depend on order
Intractability

- Instead of monitoring a single trace, we must monitor a set of traces

- Policy violation: some trace/all traces

- Even for a very restrictive setting, corresponding decision problems are intractable

Instance:
- propositional, past-only, non-metric linear-time temporal formula $\phi$
- prefixes $\bar{D}^1$ and $\bar{D}^2$ of length $n \geq 1$
  with $\bar{D}^i = (D^i_1, \tau_1)(D^i_1, \tau_1) \ldots (D^i_n, \tau_n)$, for $i \in \{1, 2\}$

Question **WEAK**: $(\bar{D}, 2n) \not\models \phi$, for some $\bar{D} \in \bar{D}^1 \parallel \bar{D}^2$ is NP-complete

Question **STRONG**: $(\bar{D}, 2n) \not\models \phi$, for all $\bar{D} \in \bar{D}^1 \parallel \bar{D}^2$ is coNP-complete
Collapsed Logs

Policies should not care about the ordering of events with equal time-stamps

\[ \square \forall u. \forall d. delete(u, db2, d) \to \Diamond <1s \, \Diamond <60s \, \exists u'. delete(u', db3, d) \]
Collapsed Logs

- Policies should not care about the ordering of events with equal time-stamps

\[ \square \forall u. \forall d. \, \text{delete}(u, db2, d) \rightarrow \bigtriangleup < 1s \bigdiamond < 60s \exists u'. \, \text{delete}(u', db3, d) \]

- Monitoring the log in which events with equal time-stamps are merged is sound and complete

- Checking if an MFOTL formula is order-independent is undecidable
  * Inductive reasoning over formula structure often sufficient
  * Approximation to order-independent properties possible
Results of Case Study

- **Performance:**
  * One year of logged data: 220 million log entries (8GB)
<table>
<thead>
<tr>
<th>policy</th>
<th>time</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>easiest</td>
<td>17 mins</td>
<td>14 MB</td>
</tr>
<tr>
<td>hardest</td>
<td>1 hour</td>
<td>3.3 GB   (mostly within 600 MB)</td>
</tr>
</tbody>
</table>
  * Processing times reasonable and space requirements manageable

- **Compliance:**
  * System users attempted unauthorized actions
  * Testing, debugging, and other improvement activities
  * Bugs in scripts and triggers

- **Value:**
  * Useful even in a benevolent environment where the enterprise is committed to policy compliance
  * Helpful to debug and sharpen controls
  * Can be used to support audits, both internal and external
Overview: Monitoring

I.) Monitoring Algorithm

II.) Case Study NOKIA

III.) Case Study Google
Scaling Up

- In larger IT systems billions of actions are executed each day
  That is, terabytes of logged data

- **Challenge:** a scalable monitoring approach

- **Solution:** use MapReduce to parallelize monitoring process

- See technical report for details:
  M. Harvan, D. Basin, G. Caronni, S. Ereth, F.K., H. Mantel
  *Checking System Compliance by Slicing and Monitoring Logs*
Monitoring with MapReduce in a Nutshell

- Framework that supports data-intensive distributed applications

![Diagram of MapReduce process]

- Allocates the tasks to different machines
- Minimizes the time of fetching the data
- Failed tasks are detected and restarted
Monitoring with MapReduce in a Nutshell

- Framework that supports data-intensive distributed applications

Advantages:
- Allocates the tasks to different machines
- Minimizes the time of fetching the data
- Failed tasks are detected and restarted
Case Study—Setting

Are the computers up-to-date when accessing the internal network?
* Each computer must regularly start an update tool
* It connects to a central server
* If necessary it downloads the latest centrally managed configuration
* It then attempts to reconfigure and update itself

Logged data of over 35,000 computers over a two year period
* 26 billion events at 77 million time-points (0.4 terabytes)
* Approximately 100 times bigger than Nokia case study

For monitoring: 1,000 computers and 10,000 slices
Case Study—Some Results

Monitor performance per slice

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>169</td>
<td>0:46</td>
<td>21.4</td>
<td>6.1</td>
<td>6.1</td>
</tr>
<tr>
<td>P2</td>
<td>170</td>
<td>0:51</td>
<td>21.4</td>
<td>6.1</td>
<td>10.3</td>
</tr>
<tr>
<td>P3</td>
<td>170</td>
<td>10:40</td>
<td>22.7</td>
<td>7.1</td>
<td>510.2</td>
</tr>
<tr>
<td>P5</td>
<td>169</td>
<td>1:06</td>
<td>21.3</td>
<td>9.2</td>
<td>13.1</td>
</tr>
<tr>
<td>P6</td>
<td>168</td>
<td>1:01</td>
<td>21.3</td>
<td>6.1</td>
<td>6.1</td>
</tr>
<tr>
<td>P7</td>
<td>168</td>
<td>0:48</td>
<td>21.1</td>
<td>6.1</td>
<td>7.1</td>
</tr>
</tbody>
</table>

Comments on P3:

* > 80% of the slices could be monitored in less than 4 hours
* a few slices were very big
Current and future directions

- **Distributed Monitoring and Enforcement**
  How to distributively monitor distributed systems online? What policies are enforceable in a concurrent setting?

- **Synthesis**
  How to effectively synthesize enforcement mechanisms from rich declarative policy specification languages?

- **Incomplete knowledge**
  How account for actions that are not logged (e.g., logging failures)? What if observations are contradictory?
  * Multi-valued logics, e.g., Belnap (t, f, ⊥, ⊤)
Conclusion

- Policy enforcement is a challenging and increasingly relevant topic. So is policy monitoring!

- Logical methods are well suited for reasoning about policies. Instance MFOTL: expressive, yet monitoring practically feasible

- Tool support publicly available at [http://projects.developer.nokia.com/MonPoly](http://projects.developer.nokia.com/MonPoly)

- No silver bullet
  - Not every policy can be formalized in MFOTL
  - Running times and space consumption is still (always will be!) an issue
Bibliography


5. David Basin, F. K., Srdjan Marinovic and Eugen Zălinescu. Monitoring compliance policies over incomplete and disagreeing logs. RV’12


10. David Basin, F. K., Samuel Müller, and Birgit Pfitzmann. Runtime monitoring of metric first-order temporal properties. FSTTCS’08