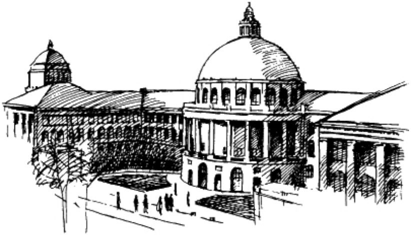
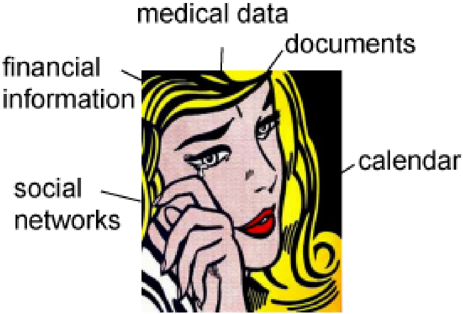
Policy Monitoring in First-order Temporal Logic

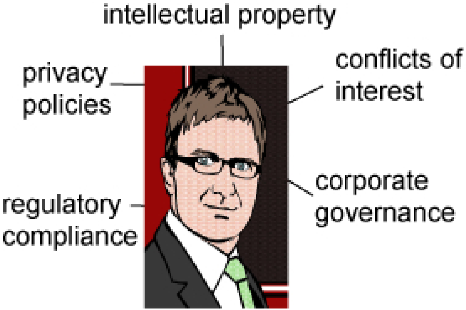
David Basin
ETH Zurich



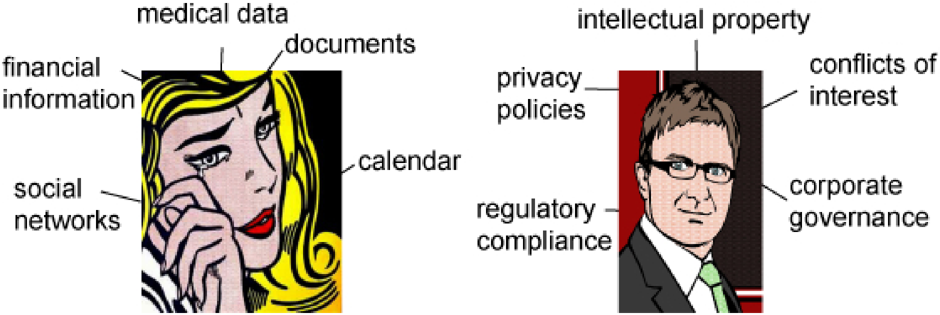
Joint work with Felix Klaedtke and Samuel Müller

Modern problems



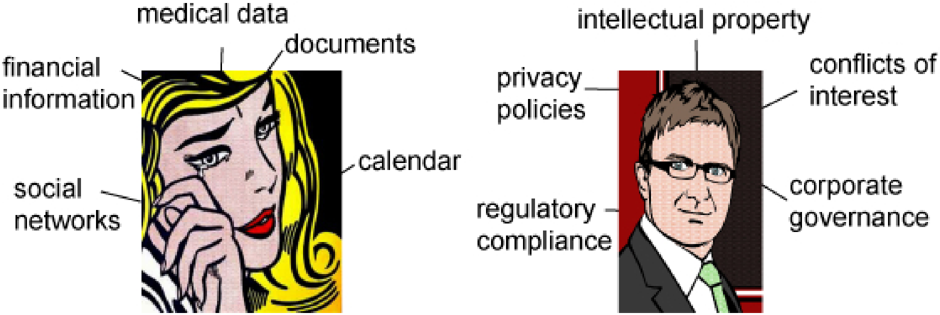


Modern problems



What do these topics have to do with each other?

Modern problems



What do these topics have to do with each other?
Are they theoretically interesting?

Technical issues

Processes to monitor and control proceses

- Controlling access
  My medical data should only be accessible to my care givers.
- Controlling usage
  ... and then used for intended purpose, e.g., improving healthcare
- Corporate governance and regulatory compliance
  Implement controls to reduce risks.

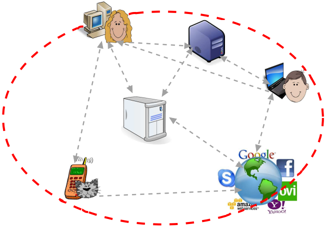
Technical issues

Processes to monitor and control proceses

- Controlling access
  My medical data should only be accessible to my care givers.
- Controlling usage
  ... and then used for intended purpose, e.g., improving healthcare
- Corporate governance and regulatory compliance
  Implement controls to reduce risks.

Core problems are theoretically interesting!

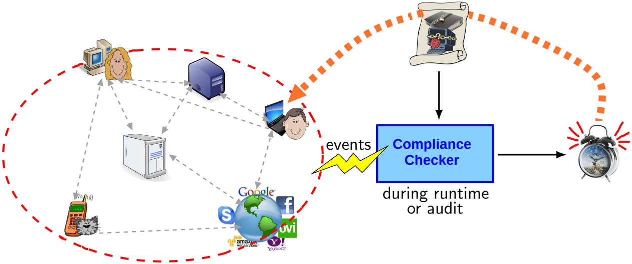
Focus



policies

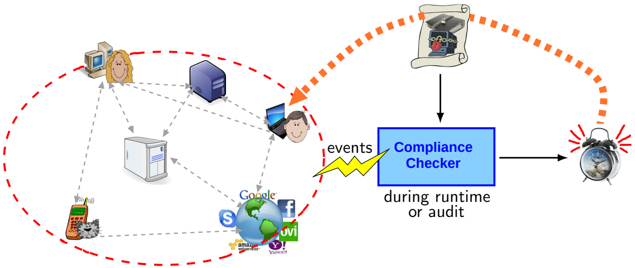
- Setting: security and compliance
  - Business processes
  - Policies regulating data and processes

Focus



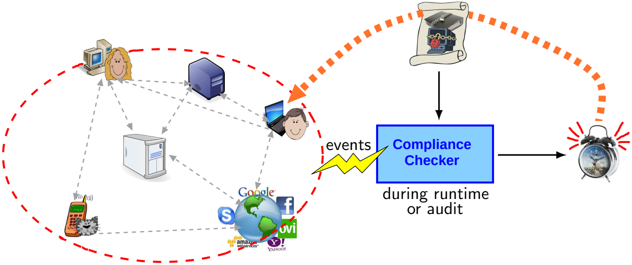
- Setting: security and compliance
  - Business processes
  - Policies regulating data and processes
- Monitoring ($\neq$ enforcement)

Focus



- Setting: security and compliance
  - Business processes
  - Policies regulating data and processes
- Monitoring ($\neq$ enforcement)
- General solution using metric first-order temporal logic and an associated monitoring algorithm

Focus



- Setting: security and compliance
  - Business processes
  - Policies regulating data and processes
- Monitoring ($\neq$ enforcement)
- General solution using metric first-order temporal logic and an associated monitoring algorithm
- Practical experience across a wide range of application areas

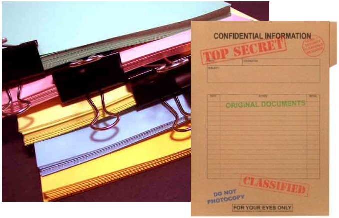
Road map

1. An example
2. Metric First-order Temporal Logic
3. Formalization examples
4. Monitoring
5. Performance
6. Conclusion

Road map

1. **An example**
2. Metric First-order Temporal Logic
3. Formalization examples
4. Monitoring
5. Performance
6. Conclusion

Example



- Consider a financial or research institute:
  - Employees write and publish reports
  - Reports may contain confidential data
- Report approval 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

```
2010-03-03    publish_report (Charlie, #234)
2010-03-04    archive_report (Alice, #104)
    ⋮               ⋮
2010-03-09    approve_report (Alice, #248)
2010-03-13    publish_report (Bob, #248)
    ⋮               ⋮
```

- Are executions policy conform?

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 time point
- logged with time stamps

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 time point
- logged with time stamps

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

These aspects can be formalized using MFOTL

$$\Box \forall e. \forall r.\, publish\_report(e, r) \rightarrow \\ \blacklozenge_{[0,11)} \exists m.\, \underline{manager}(m, e) \wedge approve\_report(m, r)$$

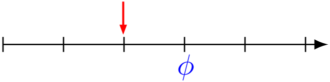
- **First-order** for expressing relations on system data.
- **Metric temporal operators** for expressing qualitative and quantitative timing information.
- Can represent both **event** and **state** predicates.

Let's look at this, starting with the temporal operators.

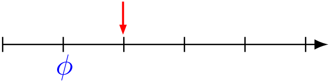
Standard linear temporal operators

- Primitive temporal operators

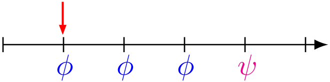
$\bigcirc\,\phi$



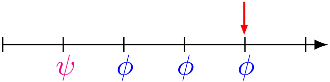
$\bullet\,\phi$



$\phi\ \mathbf{U}\ \psi$



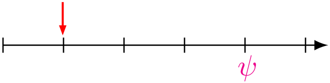
$\phi\ \mathbf{S}\ \psi$



- Derived temporal operators

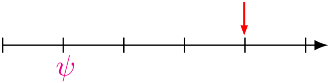
$\Diamond\,\psi$

true $\mathbf{U}\ \psi$



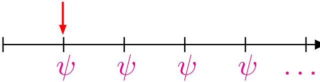
$\blacklozenge\,\psi$

true $\mathbf{S}\ \psi$



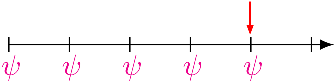
$\square\,\psi$

$\neg\Diamond\neg\psi$



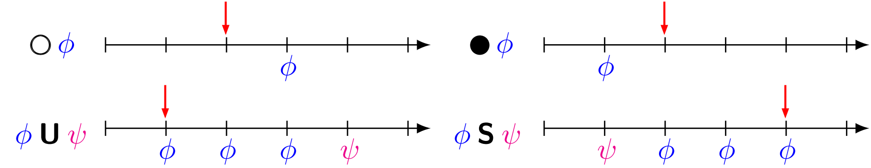
$\blacksquare\,\psi$

$\neg\blacklozenge\neg\psi$

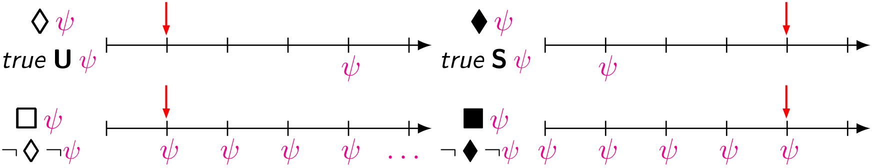


Metric temporal operators

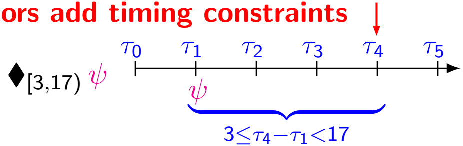
- Primitive temporal operators



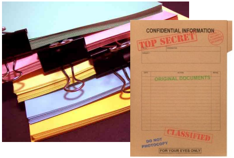
- Derived temporal operators



- **Metric operators add timing constraints**



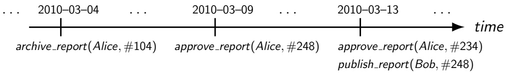
Policy revisited and simplified



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

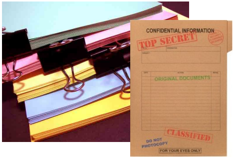
```
2010-03-04    archive_report (Alice, #104)
    ⋮                ⋮
2010-03-09    approve_report (Alice, #248)
    ⋮                ⋮
2010-03-13    approve_report (Alice, #234)
2010-03-13    publish_report (Bob, #248)
    ⋮                ⋮
```



- Simplified policy in MFOTL:

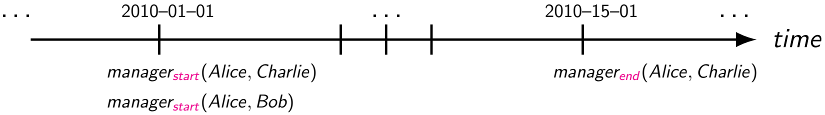
$$\Box\, \forall e.\, \forall r.\, \mathit{publish\_report}(e, r) \rightarrow \blacklozenge_{[0,11)}\, \exists m.\, \mathit{approve\_report}(m, r)$$

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
  - Also log events that mark start and end points



  - State predicate as syntactic sugar

$$\underline{manager}(m, e) := \neg manager_{end}(m, e) \;\mathbf{S}\; manager_{start}(m, e)$$

- Policy in MFOTL

$$\Box \forall e. \forall r. \, publish\_report(e, r) \rightarrow$$
$$\blacklozenge_{[0,11)} \exists m. \, \underline{manager}(m, e) \wedge approve\_report(m, r)$$

Road map

1. An example
2. **Metric First-order Temporal Logic**
   First-order variant of [Koymans 1990], [Alur/Henzinger 1990], ...
3. Formalization examples
4. Monitoring
5. Performance
6. Conclusion

Syntax

- Let $\mathbb{I}$ be the set of nonempty *intervals* over $\mathbb{N}$. Notation:
$[b, b') := \{a \in \mathbb{N} \mid b \leq a < b'\}$, for $b \in \mathbb{N}$, $b' \in \mathbb{N} \cup \{\infty\}$, and $b < b'$

- 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.

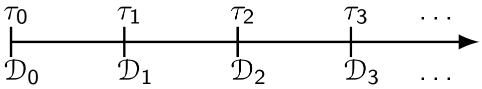
- *(MFOTL) formulas* over a signature S and set of variables V

$$\phi ::= t_1 \approx t_2 \mid t_1 \prec t_2 \mid r(t_1, \ldots, t_n) \mid \neg\phi \mid \phi \wedge \phi \mid \exists x.\phi \mid \\ \bullet_I \phi \mid \bigcirc_I \phi \mid \phi \, \mathbf{S}_I \, \phi \mid \phi \, \mathbf{U}_I \, \phi \, ,$$

  where t_i range over $V \cup C$ and r, x, I range over R, V, $\mathbb{I}$.

- Sugar like $\blacksquare_I \phi := \neg(\mathit{true} \, \mathbf{S}_I \, \neg\phi)$ and $\square_I \phi := \neg(\mathit{true} \, \mathbf{U}_I \, \neg\phi)$.
Non-metric operators like $\square \phi := \square_{[0,\infty)} \phi$

Semantics (I)



- A *temporal (first-order) structure* (over S) is a pair $(\bar{\mathcal{D}}, \bar{\tau})$.
  - Sequence of first-order structures $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \dots)$.
    Constant domains and rigid interpretation of constant symbols.
  - Sequence $\bar{\tau} = (\tau_0, \tau_1, \dots)$ of time stamps, $\tau_i \in \mathbb{N}$
    Monotonically increasing and progresses.
- $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$ denotes *MFOTL satisfaction*

  $(\bar{\mathcal{D}}, \bar{\tau})$ is a temporal structure, v a valuation, $i \in \mathbb{N}$, and ϕ a formula.
- Standard semantics for first-order fragment.

Semantics (II)

Metric temporal operators

A temporal formula is only satisfied at time point i if it is satisfied within the bounds given by interval I, relative to time stamp τ_i.

$$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \bigcirc_I \phi \quad \text{iff} \quad \tau_{i+1} - \tau_i \in I \text{ and } (\bar{\mathcal{D}}, \bar{\tau}, v, i+1) \models \phi$$

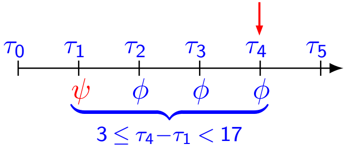
$$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \bullet_I \phi \quad \text{iff} \quad i > 0,\ \tau_i - \tau_{i-1} \in I, \text{ and } (\bar{\mathcal{D}}, \bar{\tau}, v, i-1) \models \phi$$

$$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \,\mathbf{U}_I\, \psi \quad \text{iff} \quad \text{for some } j \geq i,\ \tau_j - \tau_i \in I,\ (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi, \text{ and } (\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi, \text{ for all } k \in [i, j)$$

$$(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi \,\mathbf{S}_I\, \psi \quad \text{iff} \quad \text{for some } j \leq i,\ \tau_i - \tau_j \in I,\ (\bar{\mathcal{D}}, \bar{\tau}, v, j) \models \psi, \text{ and } (\bar{\mathcal{D}}, \bar{\tau}, v, k) \models \phi, \text{ for all } k \in [j+1, i+1)$$

Example

$\phi \,\mathbf{S}_{[3,17)}\, \psi$



Road map

1. An example
2. Metric First-order Temporal Logic
3. **Formalization examples**

   Examples illustrate typical compliance policies and their formalization in MFOTL.
4. Monitoring
5. Performance
6. Conclusion

Transaction requirements (I)

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.
  - $trans(c, t, a)$: customer c carries out transaction t involving fund amount a.
  - $auth(e, t)$: employee e authorizes t.
  - $report(t)$: t is reported.
- In general, signature determined by monitoring requirements and events that system actually provides.

Transaction requirements (II)

- Transactions t of any customers c must be reported within 5 days when the transferred amount a exceeds a given threshold th:

$$\Box \forall c. \forall t. \forall a. \mathit{trans}(c, t, a) \land \mathit{th} \prec a \rightarrow \Diamond_{[0,6)} \mathit{report}(t)$$

- Transactions exceeding the threshold must be authorized by an employee (e.g., 2-20 days) before execution:

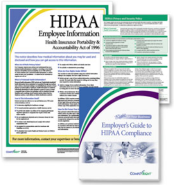
$$\Box \forall c. \forall t. \forall a. \mathit{trans}(c, t, a) \land \mathit{th} \prec a \rightarrow \blacklozenge_{[2,21)} \exists e. \mathit{auth}(e, t)$$

- 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 2 days:

$$\begin{aligned}&\Box \forall t. \forall c. \forall a. \mathit{trans}(c, t, a) \land \\ &\qquad \left(\blacklozenge_{[0,31)} \exists t'. \exists a'. \mathit{trans}(c, t', a') \land \Diamond_{[0,6)} \mathit{report}(t') \right) \rightarrow \\ &\qquad \Diamond_{[0,3)} \mathit{report}(t)\end{aligned}$$

Data retention requirements (I)

Health Insurance Portability and Accountability Act (HIPAA)



- Regulations address storage of health records.
  - Limited storage of sensitive records in the hospital's central database.
  - However, archiving is required for auditing and liability reasons.

- Signature
  - Constants *db* and *archive*: hospital's central and archive databases.
  - $hospitalize(p)$ and $release(p)$: patient p is hospitalized and released.
  - $delete(d, p)$: patient p's health record is deleted from the database d.
  - $copy(d, d', p)$: patient p's health record is copied from database d to d'.

Data retention requirements (II)

- A patient's health record must be deleted from hospital's database within 14 days after the patient is released from the hospital, unless the patient is readmitted to the hospital within this time window:

$$\Box\,\forall p.\, \mathit{release}(p) \rightarrow \Diamond_{[0,15)}\, \mathit{delete}(db, p) \vee \mathit{hospitalize}(p)\,.$$

- A health record is archived at most 7 days before it is deleted from the central database:

$$\Box\,\forall p.\, \mathit{delete}(db, p) \rightarrow \blacklozenge_{[0,8)}\, \mathit{copy}(db, \mathit{archive}, p)$$

- Archived data must be stored for at least 8 years:

$$\Box\,\forall p.\, \mathit{copy}(db, \mathit{archive}, p) \rightarrow \Box_{[0,9)}\, \neg \mathit{delete}(\mathit{archive}, p)$$

N.B. timestamps must distinguish time units, e.g., days versus years

Separation of duty requirements

Principle for preventing fraud and errors

- Requires involvement of multiple users in critical processes.
- Usually formulated on top of Role-Based Access Control.
  - Users are assigned to roles, which have associated permissions.
  - SoD constraints specified in terms of mutually exclusive roles.

Separation of duty requirements

Principle for preventing fraud and errors

- Requires involvement of multiple users in critical processes.
- Usually formulated on top of Role-Based Access Control.
  - Users are assigned to roles, which have associated permissions.
  - SoD constraints specified in terms of mutually exclusive roles.
- Signature (formalizing both RBAC and SoD)
  - $\underline{U}$, $\underline{R}$, $\underline{A}$, $\underline{O}$, and $\underline{S}$ represent the sets of users, roles, actions, objects, and sessions associated with a (RBAC) system
  - $\underline{UA}(u, r)$: user u assigned role r
  - $\underline{PA}(r, a, o)$: role r can carry out action a on object o
  - $\underline{roles}(s, r)$: role r is active in session s
  - $\underline{X}(r, r')$: roles r and r' are mutually exclusive
  - $exec(s, a, o)$: action a is executed on object o in session s

Formalizing SoD requirements

- *Static SoD*: no user may be assigned to two mutually exlusive roles

$$\Box \forall r. \forall r'. \underline{X}(r, r') \rightarrow \neg \exists u. \underline{UA}(u, r) \wedge \underline{UA}(u, r')$$

- *Simple dynamic SoD*: a user may be assigned to two exclusive roles provided he does not activate them both in the same session.

$$\begin{aligned} &\Box \forall r. \forall r'. \underline{X}(r, r') \rightarrow \\ &\qquad \neg \exists s. \underline{roles}(s, r) \wedge \left(\neg S_{end}(s) \,\mathbf{S}\, \underline{roles}(s, r')\right) . \end{aligned}$$

(Assumptions: session always associated with one user who remains constant over the session's lifetime, X is symmetric, ...)

SoD requirements (cont.)

- *Object-based SoD*: a user may be assigned to two exclusive roles and also activate them both in the same session, but he must not carry out actions on the same object through both.

$$
\begin{aligned}
&\Box \forall r. \forall r'. \underline{X}(r, r') \rightarrow \\
&\qquad \neg \exists s. \exists o. \big(\exists a.\, exec(s, a, o) \wedge \\
&\qquad\qquad \underline{roles}(s, r) \wedge \underline{PA}(r, a, o) \big) \wedge \\
&\qquad\qquad \big(\neg S_{end}(s) \; \mathbf{S} \; \exists a'.\, exec(s, a', o) \wedge \\
&\qquad\qquad\qquad \underline{roles}(s, r') \wedge \underline{PA}(r', a', o) \big)
\end{aligned}
$$

Experience with formalization in practice

Limitations and problems

- Precision must precede formalization.
  - "... must be securely stored."
- Not all requirements can be enforced by monitoring system traces.
  - "Information systems must be protected from intrusion."
  - "A contingency plan should be in place for responding to emergencies."
- Large gap between high-level policies and system information.
  - "Data should be use for statistical purposes only."
  - "... must be deleted ..."

Overcoming these problems is nontrivial.
MFOTL is a good fit afterwards.

Road map

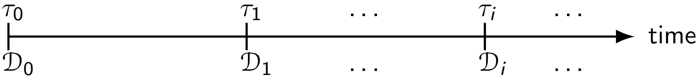
1. An example
2. Metric First-order Temporal Logic
3. Formalization examples
4. **Monitoring**
5. Performance
6. Conclusion

Monitoring objective

- Given a policy ϕ (example from transaction processing)

$$\Box\,\forall t.\,\forall c.\,\forall a.\,\mathit{trans}(c,t,a) \wedge \big(\blacklozenge_{[0,31)}\,\exists t'.\,\exists a'.\,\mathit{trans}(c,t',a') \wedge \Diamond_{[0,6)}\,\mathit{report}(t')\big) \rightarrow \Diamond_{[0,3)}\,\mathit{report}(t)$$

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



$\mathit{trans}^{\mathcal{D}_0}$	*tID*	*cID*
	t34	Bob
	t23	Bob

$\mathit{report}^{\mathcal{D}_0}$	*tID*	*cID*
	t13	Alice

$\mathit{trans}^{\mathcal{D}_1}$	*tID*	*cID*
	t22	Eve

$\mathit{report}^{\mathcal{D}_1}$	*tID*	*cID*
	t34	Bob
	t18	Joe

$\mathit{trans}^{\mathcal{D}_i}$	*tID*	*cID*
	t11	Bob
	t41	Mallory

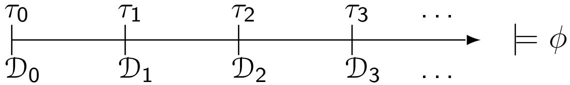
$\mathit{report}^{\mathcal{D}_i}$	*tID*	*cID*

- monitor should report all policy violations

Main ideas sketched here. Definitions and proofs in proceedings and FSTTCS 2008 paper and technical report.

Restrictions

Not all policies and log files can be effectively monitored



- MFOTL formula ϕ of form $\Box\, \phi'$, where ϕ' is bounded.
  - For all occurrences of operator $\mathbf{U}_{[c,d)}$ in ϕ', $d \neq \infty$
  - So ϕ describes a safety property

- Structures $\bar{\mathcal{D}} = \mathcal{D}_0, \mathcal{D}_1, \ldots$ Options:

  1. Each structure $\mathcal{D}_i$ is automatic

     Roughly, each $\mathcal{D}_i$ representable by a collection of finite automata.
     See, e.g. [Khoussainov & Nerode 1995] and [Blumensath & Grädel 2004]

  2. **or** all relations in $\mathcal{D}_i$ are finite (Special case of 1.)

Preprocessing: negation and rewriting

- Input formula ϕ

$$\Box\, \forall t.\, \forall c.\, \forall a.\, \mathit{trans}(c, t, a) \wedge \big(\blacklozenge_{[0,31)}\, \exists t'.\, \exists a'.\, \mathit{trans}(c, t', a') \wedge \Diamond_{[0,6)}\, \mathit{report}(t')\big) \\ \rightarrow \Diamond_{[0,3)}\, \mathit{report}(t)$$

Preprocessing: negation and rewriting

- Input formula ϕ

$$\Box \forall t.\, \forall c.\, \forall a.\, \mathit{trans}(c, t, a) \wedge \left(\blacklozenge_{[0,31)} \exists t'.\, \exists a'.\, \mathit{trans}(c, t', a') \wedge \Diamond_{[0,6)}\, \mathit{report}(t') \right) \\ \quad \rightarrow \Diamond_{[0,3)}\, \mathit{report}(t)$$

- Negate, rewrite, and drop outermost $\Diamond$ and $\exists$ quantifiers, yielding ψ

$$\Diamond \exists t.\, \exists c.\, \exists a.\, \mathit{trans}(c, t, a) \wedge \left(\blacklozenge_{[0,31)} \exists t'.\, \exists a'.\, \mathit{trans}(c, t', a') \wedge \Diamond_{[0,6)}\, \mathit{report}(t') \right) \\ \quad \wedge\, \Box_{[0,3)}\, \neg \mathit{report}(t)$$

Preprocessing: negation and rewriting

- Input formula ϕ

$$\Box\,\forall t.\,\forall c.\,\forall a.\,\mathit{trans}(c,t,a) \wedge \big(\blacklozenge_{[0,31)}\,\exists t'.\,\exists a'.\,\mathit{trans}(c,t',a') \wedge \Diamond_{[0,6)}\,\mathit{report}(t')\big) \\ \quad \rightarrow \Diamond_{[0,3)}\,\mathit{report}(t)$$

- Negate, rewrite, and drop outermost $\Diamond$ and $\exists$ quantifiers, yielding ψ

$$\Diamond\,\exists t.\,\exists c.\,\exists a.\,\mathit{trans}(c,t,a) \wedge \big(\blacklozenge_{[0,31)}\,\exists t'.\,\exists a'.\,\mathit{trans}(c,t',a') \wedge \Diamond_{[0,6)}\,\mathit{report}(t')\big) \\ \quad \wedge\ \Box_{[0,3)}\,\neg\mathit{report}(t)$$

- To monitor: for each $i \in \mathbb{N}$, determine elements satisfying ψ:

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

These are suspicious transactions that were not reported.

Preprocessing: reduction to first-order queries

- For each temporal subformula α in ψ, introduce an auxiliary predicate p_α

$$\mathit{trans}(c,t,a) \wedge \underbrace{\big(\blacklozenge_{[0,31)} \exists t'.\, \exists a'.\, \mathit{trans}(c,t',a') \wedge \underbrace{\lozenge_{[0,6)}\, \mathit{report}(t')\big)}_{p_{\alpha_1}}}_{p_{\alpha_3}} \wedge \underbrace{\square_{[0,3)} \neg \mathit{report}(t)}_{p_{\alpha_2}}$$

Preprocessing: reduction to first-order queries

- For each temporal subformula α in ψ, introduce an auxiliary predicate p_α

$$\mathit{trans}(c,t,a) \land \underbrace{\left(\blacklozenge_{[0,31)} \exists t'.\, \exists a'.\, \mathit{trans}(c,t',a') \land \underbrace{\lozenge_{[0,6)}\, \mathit{report}(t')}_{p_{\alpha_1}}\right)}_{p_{\alpha_3}} \land \underbrace{\Box_{[0,3)} \neg \mathit{report}(t)}_{p_{\alpha_2}}$$

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

$$\mathit{trans}(c,t,a) \land p_{\alpha_3}(c) \land p_{\alpha_2}(t)$$

Preprocessing: reduction to first-order queries

- For each temporal subformula α in ψ, introduce an auxiliary predicate p_α

$$\mathit{trans}(c,t,a) \land \big(\underbrace{\blacklozenge_{[0,31)}\, \exists t'.\, \exists a'.\, \mathit{trans}(c,t',a') \land \underbrace{\Diamond_{[0,6)}\, \mathit{report}(t')}_{p_{\alpha_1}}}_{p_{\alpha_3}}\big) \land \underbrace{\Box_{[0,3)}\, \neg \mathit{report}(t)}_{p_{\alpha_2}}$$

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

$$\mathit{trans}(c,t,a) \land p_{\alpha_3}(c) \land p_{\alpha_2}(t)$$

- Monitoring: for each $i \in \mathbb{N}$
  - Extend $\mathcal{D}_i$ to $\hat{\mathcal{D}}_i$, where for each temporal subformula α

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

  - Query extended first-order structure $\hat{\mathcal{D}}_i$

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

Preprocessing: reduction to first-order queries

- For each temporal subformula α in ψ, introduce an auxiliary predicate p_α

$$trans(c,t,a) \land \big(\underbrace{\blacklozenge_{[0,31)} \exists t'.\, \exists a'.\, trans(c,t',a') \land \underbrace{\lozenge_{[0,6)}\, report(t')}_{p_{\alpha_1}}}_{p_{\alpha_3}}\big) \land \underbrace{\square_{[0,3)} \neg report(t)}_{p_{\alpha_2}}$$

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

$$trans(c,t,a) \land p_{\alpha_3}(c) \land p_{\alpha_2}(t)$$

- Monitoring: for each $i \in \mathbb{N}$
  - Extend $\mathcal{D}_i$ to $\hat{\mathcal{D}}_i$, where for each temporal subformula α

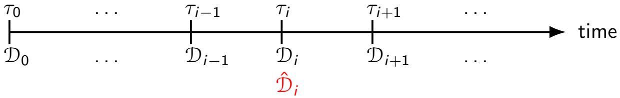
$$p_\alpha^{\hat{\mathcal{D}}_i} = \{\bar{a} \mid (\bar{\mathcal{D}}, \bar{\tau}, v[\bar{x}/\bar{a}], i) \models \alpha\}$$

  - Query extended first-order structure $\hat{\mathcal{D}}_i$

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

Next: how to incrementally build the auxiliary relations $p_\alpha^{\hat{\mathcal{D}}_i}$ for each $\hat{\mathcal{D}}_i$

Building the auxiliary relations



- Build auxiliary relations $p_\alpha^{\hat{\mathcal{D}}_i}$ in $\hat{\mathcal{D}}_i$ inductively over α's formula structure and using relations from both **previous** and **subsequent** structures.

- Example for $\alpha := \bullet_I \beta$

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

- Example for $\alpha := \bigcirc_I \beta$

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

Depends on the relations in D_{i+1} and auxiliary relations in $\hat{D}_{i+1}$.
Hence monitor instantiates $p_\alpha^{\hat{D}_i}$ with a delay of at least one time step.

Construction for $\mathbf{S}_{[b,b')}$

First consider the non-metric case $\alpha := \beta\,\mathbf{S}\,\gamma$

- For $\alpha := \beta\,\mathbf{S}\,\gamma$, construction reflects logical equivalence

$$\alpha \leftrightarrow \gamma \vee (\beta \wedge \bullet\,\alpha)$$

- Let $i \geq 0$ and assume that β and γ have the same free variables. Then

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

- Uses relations just for subformulas and (here) past time points.

Construction for $\mathsf{S}_{[b,b')}$

Metric case for $\alpha := \beta\, \mathsf{S}_{[b,b')}\, \gamma$

Recall (non-metric):

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

- Define additional auxiliary relation r_α for each $\mathcal{D}_i$ by

$$r_\alpha^{\hat{\mathcal{D}}_i} := (\hat{\gamma}^{\hat{\mathcal{D}}_i} \times \{0\}) \cup \begin{cases} \emptyset & \text{if } i = 0 \\ \left\{(\bar{a}, y) \;\middle|\; \bar{a} \in \hat{\beta}^{\hat{\mathcal{D}}_i},\ y < b', \text{ and } (\bar{a}, y + \tau_{i-1} - \tau_i) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}}\right\} & \text{if } i > 0 \end{cases}$$

- If $(\bar{a}, y) \in r_\alpha^{\hat{\mathcal{D}}_i}$, the age y expresses how long ago $\bar{a}$ satisfies α, independent of lower bound b
  - If $\bar{a}$ satisfies γ at i: add $\bar{a}$ to $r_\alpha^{\hat{\mathcal{D}}_i}$ with age 0.
  - If $i > 0$, $y < b'$ (not too old), and $\bar{a}$ satisfies β at i: add updated tuples by increasing the age of $(\bar{a}, y) \in r_\alpha^{\hat{\mathcal{D}}_{i-1}}$ by $\tau_i - \tau_{i-1}$.
- Obtain $p_\alpha^{\hat{\mathcal{D}}_i}$ from $r_\alpha^{\hat{\mathcal{D}}_i}$ by checking if age y of a tuple in $r_\alpha^{\hat{\mathcal{D}}_i}$ is old enough:

$$p_\alpha^{\hat{\mathcal{D}}_i} := \left\{\bar{a} \;\middle|\; (\bar{a}, y) \in r_\alpha^{\hat{\mathcal{D}}_i}, \text{ for some } y \geq b\right\}$$

Monitor $\mathcal{M}(\psi)$

$i \leftarrow 0$ *% lookahead index in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$
$q \leftarrow 0$ *% index of next query evaluation in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$
$Q \leftarrow \{ (\alpha, 0, \mathit{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \}$
loop
    Carry over constants and relations of $\mathcal{D}_i$ to $\hat{\mathcal{D}}_i$.
    for all $(\alpha, j, \emptyset) \in Q$ **do** *% can build relation for* α *in* $\hat{\mathcal{D}}_j$
        Build auxiliary relations for α in $\hat{\mathcal{D}}_j$.
        Discard auxiliary relations for α in $\hat{\mathcal{D}}_{j-1}$ if $j - 1 \geq 0$.
        Discard relations $p_\delta^{\hat{\mathcal{D}}_j}$, where δ is a temporal subformula of α.
    while all relations $p_\alpha^{\hat{\mathcal{D}}_q}$ are built for $\alpha \in \mathit{tsub}(\psi)$ **do**
        Output violations $\hat{\psi}^{\hat{\mathcal{D}}_q}$ and time stamp τ_q.
        Discard structure $\hat{\mathcal{D}}_{q-1}$ if $q > 0$.
        $q \leftarrow q + 1$
    $Q \leftarrow \{ (\alpha, i+1, \mathit{waitfor}(\alpha)) \mid \alpha \text{ temporal subformula of } \psi \} \cup$
        $\{ (\alpha, j, \bigcup_{\alpha' \in \mathit{update}(S, \tau_{i+1} - \tau_i)} \mathit{waitfor}(\alpha')) \mid (\alpha, j, S) \in Q \text{ and } S \neq \emptyset \}$
    $i \leftarrow i + 1$ *% process next element in input sequence* (D_{i+1}, τ_{i+1})
end loop

Counters q (query) and i (lookahead) into input sequence

Monitor $\mathcal{M}(\psi)$

$i \leftarrow 0$ *% lookahead index in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$
$q \leftarrow 0$ *% index of next query evaluation in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$
$Q \leftarrow \big\{ (\alpha, 0, \mathit{waitfor}(\alpha)) \,\big|\, \alpha$ temporal subformula of $\psi \big\}$
loop
    Carry over constants and relations of $\mathcal{D}_i$ to $\hat{\mathcal{D}}_i$.
    for all $(\alpha, j, \emptyset) \in Q$ **do**   *% can build relation for* α *in* $\hat{\mathcal{D}}_j$
        Build auxiliary relations for α in $\hat{\mathcal{D}}_j$.
        Discard auxiliary relations for α in $\hat{\mathcal{D}}_{j-1}$ if $j - 1 \geq 0$.
        Discard relations $p_\delta^{\hat{\mathcal{D}}_j}$, where δ is a temporal subformula of α.
    while all relations $p_\alpha^{\hat{\mathcal{D}}_q}$ are built for $\alpha \in \mathit{tsub}(\psi)$ **do**
        Output violations $\hat{\psi}^{\hat{\mathcal{D}}_q}$ and time stamp τ_q.
        Discard structure $\hat{\mathcal{D}}_{q-1}$ if $q > 0$.
        $q \leftarrow q + 1$
    $Q \leftarrow \big\{ (\alpha, i+1, \mathit{waitfor}(\alpha)) \,\big|\, \alpha$ temporal subformula of $\psi \big\} \cup$
        $\big\{ (\alpha, j, \bigcup_{\alpha' \in \mathit{update}(S, \tau_{i+1} - \tau_i)} \mathit{waitfor}(\alpha')) \,\big|\, (\alpha, j, S) \in Q$ and $S \neq \emptyset \big\}$
    $i \leftarrow i + 1$ *% process next element in input sequence* (D_{i+1}, τ_{i+1})
end loop

Q maintains list of unevaluated subformula (α, j, S) for past time points

Monitor $\mathcal{M}(\psi)$

$i \leftarrow 0$ *% lookahead index in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \dots$

$q \leftarrow 0$ *% index of next query evaluation in sequence* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \dots$

$Q \leftarrow \{ (\alpha, 0, \mathit{waitfor}(\alpha)) \mid \alpha$ temporal subformula of $\psi \}$

loop

  Carry over constants and relations of $\mathcal{D}_i$ to $\hat{\mathcal{D}}_i$.

  for all $(\alpha, j, \emptyset) \in Q$ **do** *% can build relation for* α *in* $\hat{\mathcal{D}}_j$

    Build auxiliary relations for α in $\hat{\mathcal{D}}_j$.

    Discard auxiliary relations for α in $\hat{\mathcal{D}}_{j-1}$ if $j - 1 \geq 0$.

    Discard relations $p_\delta^{\hat{\mathcal{D}}_j}$, where δ is a temporal subformula of α.

  while all relations $p_\alpha^{\hat{\mathcal{D}}_q}$ are built for $\alpha \in \mathit{tsub}(\psi)$ **do**

    Output violations $\hat{\psi}^{\hat{\mathcal{D}}_q}$ and time stamp τ_q.

    Discard structure $\hat{\mathcal{D}}_{q-1}$ if $q > 0$.

    $q \leftarrow q + 1$

  $Q \leftarrow \{ (\alpha, i+1, \mathit{waitfor}(\alpha)) \mid \alpha$ temporal subformula of $\psi \} \cup$
  $\{ (\alpha, j, \bigcup_{\alpha' \in \mathit{update}(S, \tau_{i+1} - \tau_i)} \mathit{waitfor}(\alpha')) \mid (\alpha, j, S) \in Q$ and $S \neq \emptyset \}$

  $i \leftarrow i + 1$ *% process next element in input sequence* (D_{i+1}, τ_{i+1})

end loop

Given relations for all temporal subformulas, output policy violations

Recall restriction on structures

1. Each structure is automatic.
2. **or** all relations in every structure are finite. (Special case of 1)

Let's look briefly at each case.

Monitoring with automatic structures

- For simplicity, fix structure's domain as $\mathbb{N}$. Encode tuples in $\mathbb{N}^k$ as words, using a binary representation and convolution.

  $$(5,3) \rightsquigarrow (101,11) \rightsquigarrow (1,1)(0,1)(1,\#)$$

  Thus each relation corresponds to languages.

- An automatic structure is one where the structure's domain, equality, and all relations are representable as regular languages.

- Theorem: If the structures $\mathcal{D}_i$ are automatic then so are the $\hat{\mathcal{D}}_i$, i.e. all auxiliary relations can be represented by automata. So can $\hat{\psi}^{\hat{\mathcal{D}}_i}$.

  Proof uses closure properties of regular languages and that basic arithmetic relations are first-order definable in $(\mathbb{N},<)$ and thus regular. E.g.

  $$\{(x,y) \in \mathbb{N}^2 \mid y = x+1\} \text{ and } \{(x,y) \in \mathbb{N}^2 \mid x+d \leq y\} \text{ for any } d \in \mathbb{N}$$

Monitoring with finite relations

- If all relations are finite, databases are an efficient alternative to automata for implementing monitoring algorithm.

- Problem: must restrict negation and quantification. Consider:

$$r(x) \wedge \bigcirc \neg q(x)$$

At each $i \in \mathbb{N}$, monitor stores $p_{\bigcirc \neg q(x)}^{\mathcal{D}_i}$, which is infinite.

Monitoring with finite relations

- If all relations are finite, databases are an efficient alternative to automata for implementing monitoring algorithm.
- Problem: must restrict negation and quantification. Consider:

  $$r(x) \wedge \bigcirc \neg q(x)$$

  At each $i \in \mathbb{N}$, monitor stores $p^{\mathcal{D}_i}_{\bigcirc \neg q(x)}$, which is infinite.
- Solution: rewrite to equivalent formula where stored relations finite.

  $$r(x) \wedge \bigcirc \left(\neg q(x) \wedge \bullet\, r(x)\right)$$

- Solution is a heuristic: rewrite into a syntactically defined form.

  N.B.: related to problem of *(temporal subformula) domain independence*.
  [Fagin 1982], [Chomicki 1995], [Chomicki, Toman, Böhlen, 2001]

Road map

1. An example
2. Metric First-order Temporal Logic
3. Formalization examples
4. Monitoring
5. **Performance**
   When monitoring with finite relations
6. Conclusion

Analysis of space consumption of $\mathcal{M}(\psi)$

- Assumptions
  - Relations are finite and ψ is monitorable
  - Number of equal time stamps is bounded
- Let the active domain be the set of data elements occurring in the relations in a prefix of a timed temporal structure.
- Theorem: At each time point, space $\mathcal{M}(\psi)$ needs to store auxiliary relations is polynomially bounded by cardinality of the active domain.
- In practice, space requirements often modest.

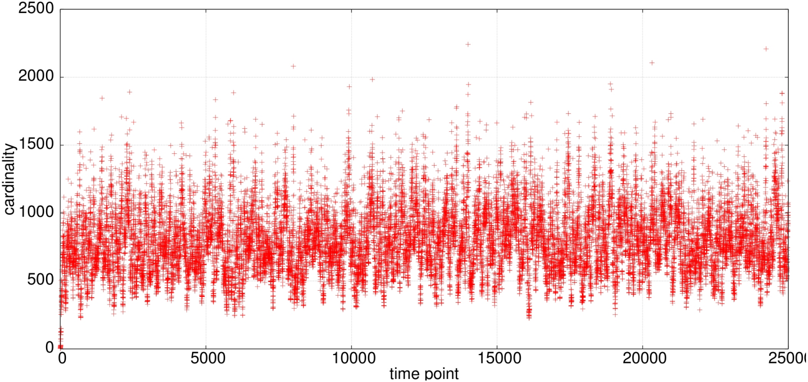
  Only a relevant part of history is required (and must be saved) at any time, with an associated, smaller relevant active domain.

Experimental evaluation

- Prototype implementations in Java (evaluated here) and OCAML
- Evaluated using polices from different domains on synthetically generated event streams
- Measured monitor's space consumption and event processing time
- Where meaningful, we conducted a steady-state analysis (estimated average performance in the long run)

Profiling the monitor

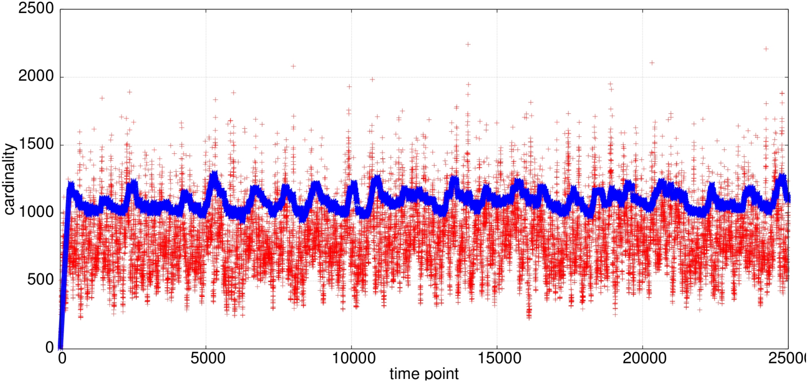
Monitor's space consumption (sum of cardinalities of stored relations at each time point)



$$\Box\, \forall t.\, \forall c.\, \forall a.\, \mathit{trans}(c, t, a) \land \\ \quad \big(\blacklozenge_{[0,31)}\, \exists t'.\, \exists a'.\, \mathit{trans}(c, t', a') \land \Diamond_{[0,6)}\, \mathit{report}(t')\big) \rightarrow \\ \quad \Diamond_{[0,3)}\, \mathit{report}(t)$$

Profiling the monitor

Monitor's space consumption (sum of cardinalities of stored relations at each time point)



Performance depends on data items occurring in processed event stream

The size of the relevant active domains stabilizes after a warm-up phase

Space consumption typically fluctuates around size of the relevant active domains

Experimental evaluation results

		event frequency					
formula	aspect	110	220	330	440	550	sample space
⋮			⋮		⋮		
Transact. policy	*ipt*	2.2	3.5	4.7	6.0	7.6	$\Omega_{1000\times 25000\times 2\times 200}$
	sc	140 ± 2.8	405 ± 9.0	801 ± 19.1	$1{,}334\pm 32.2$	$1{,}994\pm 47.8$	
	omax	723	1,270	2,242	3,302	4,360	
	radom	404	762	1,098	1,422	1,726	
⋮			⋮		⋮		

ipt — estimated mean incremental processing time (in milliseconds)
sc — estimated mean space consumption (# of elements stored in relations, 95% within interval)
omax — observed maximal space consumption
radom — size of relevant active domain

- Moderate space consumption and running times
- Growth rates linear in the event frequency (approximate number of events in formula's time window)
- Past operators are handled more efficiently than future operators
- State predicates increase space consumption

Road map

1. An example
2. Metric First-order Temporal Logic
3. Formalization examples
4. Monitoring
5. Performance
6. **Conclusion**

Conclusion

expressivity
vs.
complexity

- MFOTL good for formalizing and monitoring a wide variety of policies.
  - Arbitrary nesting of operators and quantifiers, although restrictions on negation in finite relation case
  - No such restrictions necessary when using automatic structures
  - Incremental constructions with "bounded history" encoding
  - Studies indicate practical feasibility.
- No silver bullet
  - Not every policy can be formalized in MFOTL
  - Efficiency depends on policy formalization

    E.g., past-time formulations better than equivalent future-time ones

Current and future work

- Case study: Nokia data collection campaign.
  - Complex requirements on how mobile-phone data is shared and used
  - Complex architecture: mobile phones, various servers, etc.
  - Must scale ultimately to $> 10^6$ users. Data-structures critical.
- Implementation using automatic structures
- Enforcement rather than audit
  - Central monitoring easier than distributed control
  - Enforcing constraints on the future (obligations) is nontrivial
    - Logically, e.g., disjunctive conditions
    - Initiating actions more difficult than supressing them

Bibliography

- **Monitoring foundations**
  - D.B., Felix Klaedtke, Samuel Müller: Policy Monitoring in First-order Temporal Logic, CAV 2010.
  - D.B., Felix Klaedtke, Samuel Müller, Birgit Pfitzmann: Runtime Monitoring of Metric First-order Temporal Properties. FSTTCS 2008.
- **Applications and enforcement**
  - D.B., Felix Klaedtke, Samuel Müller: Monitoring security policies with metric first-order temporal logic. SACMAT 2010.
  - Alex Pretschner, Manuel Hilty, D.B., Christian Schaefer, Thomas Walter: Mechanisms for usage control. ASIACCS 2008.
  - Alex Pretschner, Manuel Hilty, D.B., Distributed usage control. Commun. ACM 49(9), 2006.
  - Manuel Hilty, D.B., Alex Pretschner: On Obligations. ESORICS 2005.