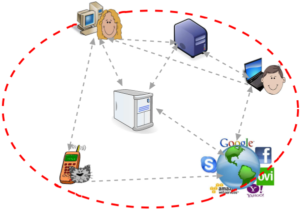
Monitoring Security Policies

Felix Klaedtke
NEC Labs Europe

Story so far . . .

- Which policies are **enforceable**?
  - Characterization for an abstract setting
  - Enforcement via execution monitoring



policies

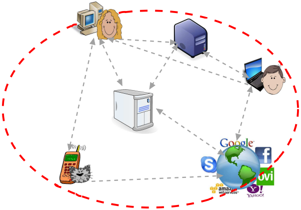
system

allowed action?

enforcement mechanism

Story so far ...

- Which policies are **enforceable**?
  - Characterization for an abstract setting
  - Enforcement via execution monitoring



policies

system

allowed action?

enforcement mechanism

- In the following:
  How to check **policy compliance** of system behavior?

$$\text{behavior} \overset{?}{\models} \text{policy}$$

Why relevant?

- Policies are omnipresent but not all are enforceable

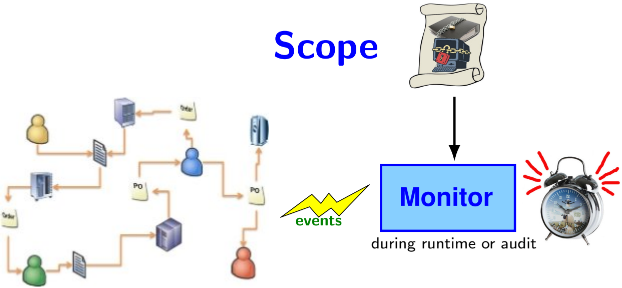
- Even when enforceable, the enforcement mechanism might be missconfigured or corrupted

- Strengthen security controls, audits, system debugging, . . .
  See NIST SP 800-92: "Guide to Computer Security Log Management"

Why different?

- **Policy enforcement and monitoring are related but ...**
- *Monitoring is simpler!*
  A monitor only needs to observe the system and report the violations
  * Events must only be observable
  * When monitoring online, violations can be reported possibly with a delay
  * Monitoring a trace offline is also possible
- *Monitoring is more generally applicable!*
  * For $P \subseteq \Sigma^\infty$, if P is enforceable then P is "monitorable"
  * Pnueli & Zaks (2006):
    "A verdict for an infinite sequence is always possible by an observation."
  * Examples: ω-safety properties and also some ω-liveness properties (e.g., *eventually p*)
  * Nonexamples: some ω-liveness properties (e.g., *always eventually p*)
  * Alternative characterizations/views exist (e.g., [Falcone et al. '12])

Scope



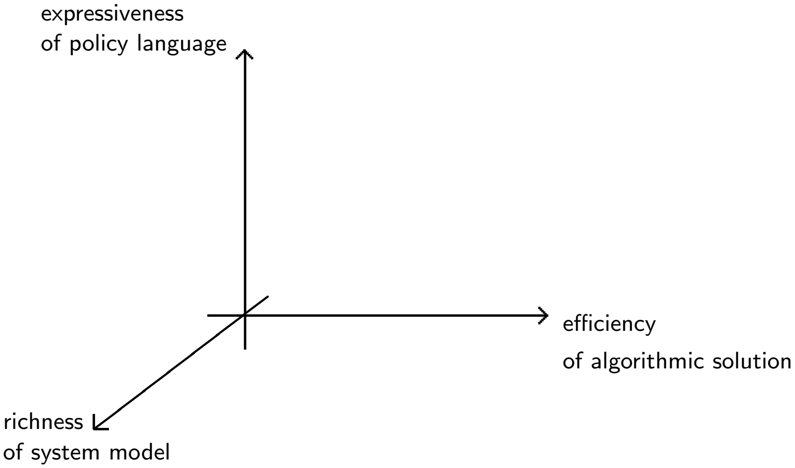
- **Setting:** policies stipulate data usage and agent behavior in IT systems or business processes

  HIPAA, SOX, separation of duty, etc.

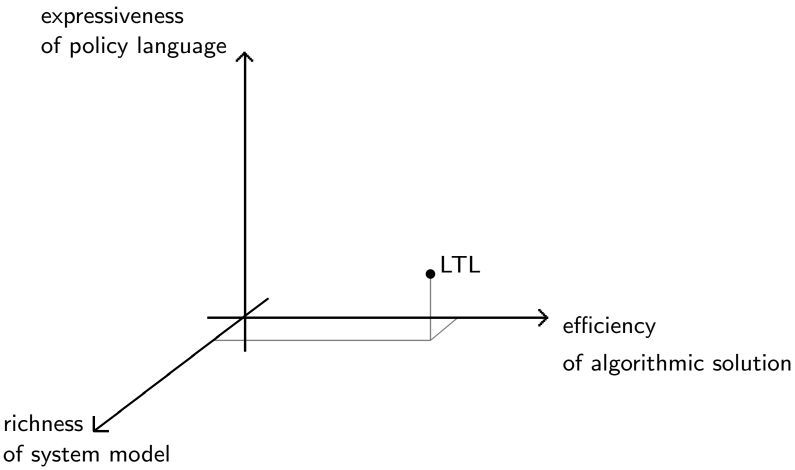
- **Objective:** detect policy violations

- **Focus:** **policy specification** and **monitoring**

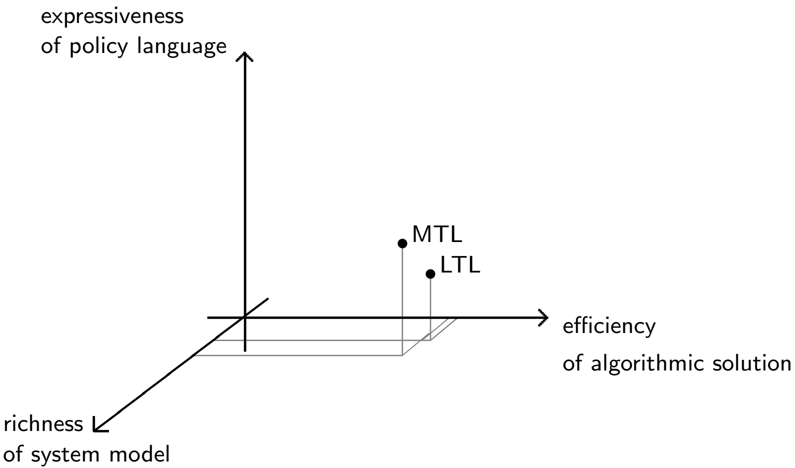
Why challenging?



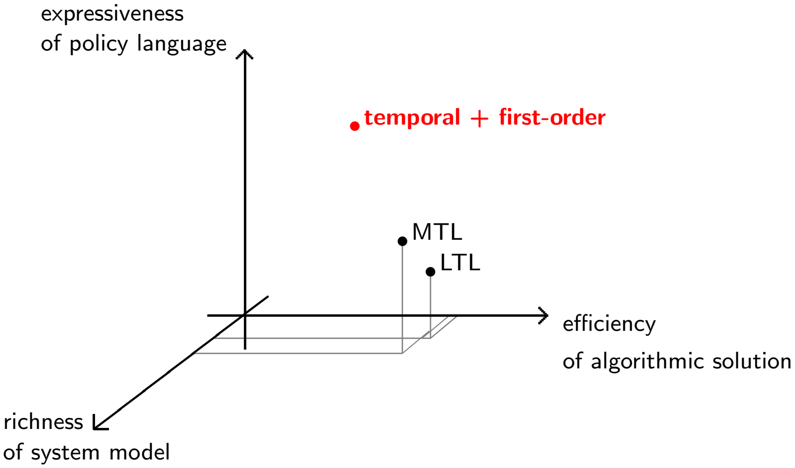
Why challenging?



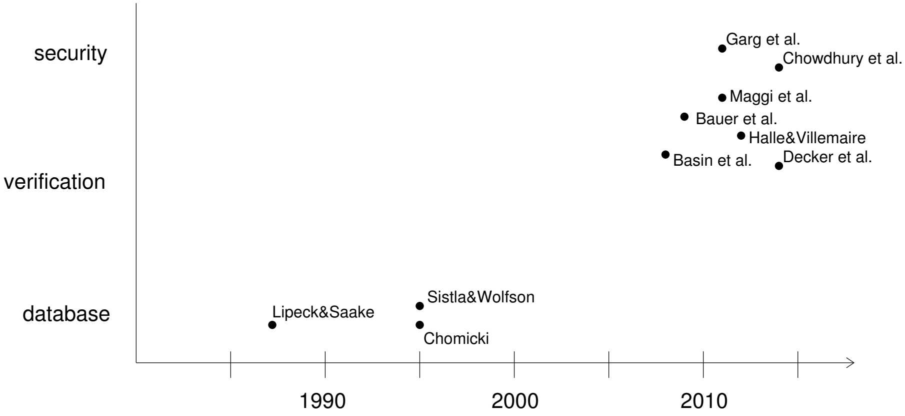
Why challenging?



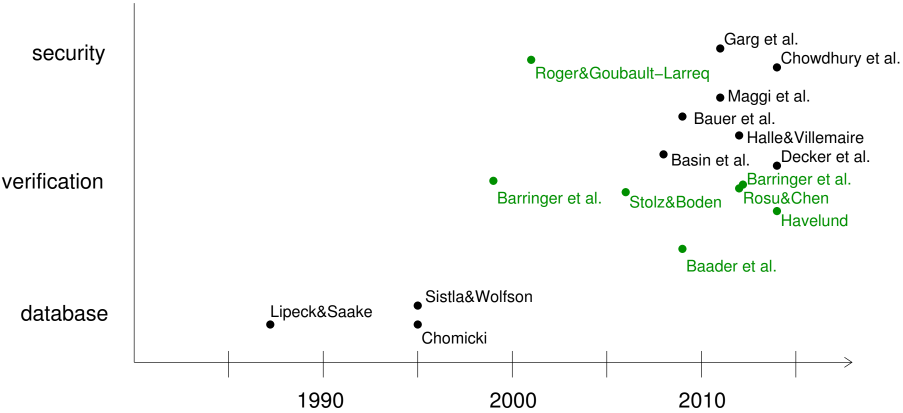
Why challenging?



Monitoring first-order temporal properties

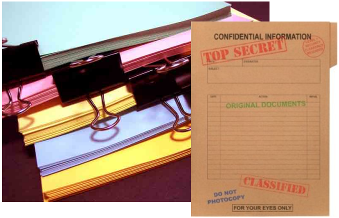


Monitoring first-order temporal properties



Policy Specification

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

```
2013-03-03    publish_report (Charlie, #234)
2013-03-04    archive_report (Alice, #104)
    ⋮               ⋮
2013-03-09    approve_report (Alice, #248)
2013-03-13    publish_report (Bob, #248)
    ⋮               ⋮
```

- 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.*

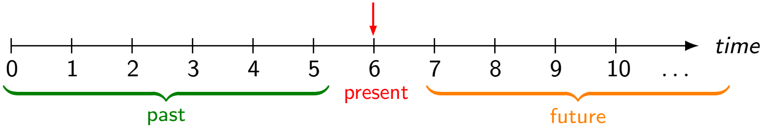
Temporal aspects

- qualitative: **before** and **always**
- quantitative: **at most 10 days**

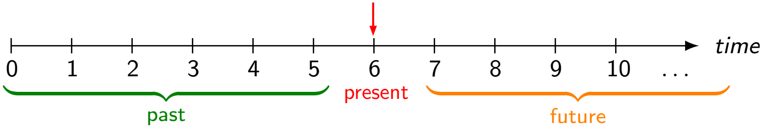
State predicates

- **being someone's manager**
- have a duration

Linear-time temporal logic

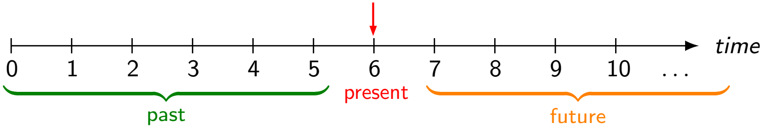


Linear-time temporal logic



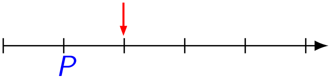
- At each time point $i \in \mathbb{N}$, a proposition P is either true or false

Linear-time temporal logic

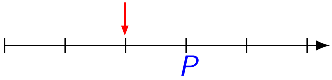


- At each time point $i \in \mathbb{N}$, a proposition P is either true or false
- **Previous** and **Next**

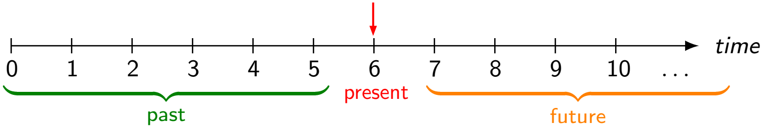
$\bullet P$



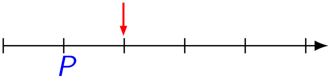
$\circ P$

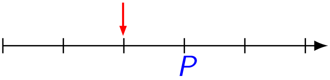


Linear-time temporal logic

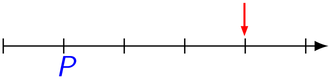


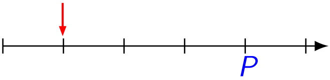
- At each time point $i \in \mathbb{N}$, a proposition P is either true or false
- **Previous** and **Next**

$\bullet P$ 

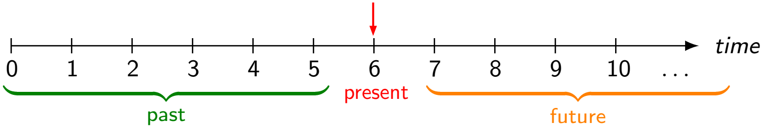
$\bigcirc P$ 

- **Once** and **Eventually** (including **present**)

$\blacklozenge P$ 

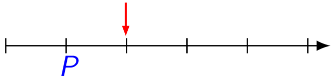
$\Diamond P$ 

Linear-time temporal logic

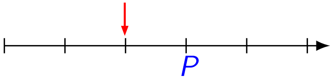


- At each time point $i \in \mathbb{N}$, a proposition P is either true or false
- **Previous** and **Next**

$\bullet P$

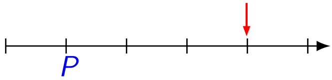


$\bigcirc P$

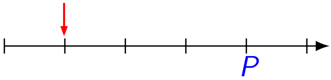


- **Once** and **Eventually** (including **present**)

$\blacklozenge P$

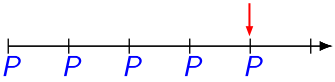


$\Diamond P$

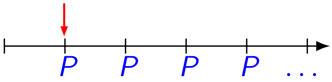


- **Historically** and **Generally** (including **present**)

$\blacksquare P$

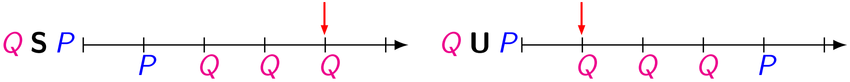


$\Box P$



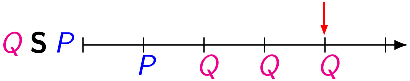
Since and Until

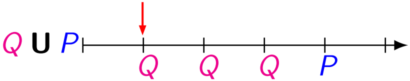
- Temporal operators: **Since** and **Until**



Since and Until

- Temporal operators: **Since** and **Until**





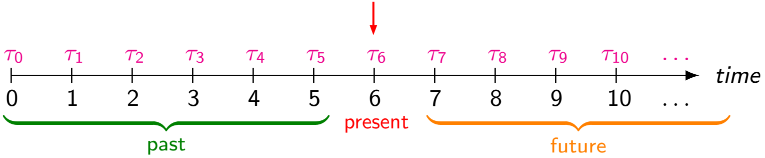
- Examples:

$$\Box\, access \rightarrow \blacklozenge\, login \qquad\qquad \neg\big((\neg login)\ \mathbf{U}\ (access \wedge \neg login)\big)$$

$$\Box\, access \rightarrow \big((\neg logout)\ \mathbf{S}\ login\big)$$

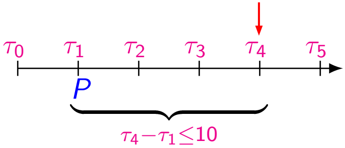
“a user is not allowed to acccess a file before he has not logged in”

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 timining constraints to temporal operators

$\blacklozenge_{\leq 10}\, P$



Propositional MTL

- **Syntax:** P an atomic proposition from AP and I an interval over $\mathbb{N}$

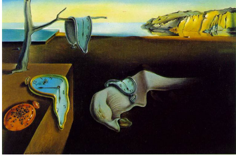
$$\phi ::= P \;\big|\; \neg\phi \;\big|\; \phi \vee \phi \;\big|\; \bullet_I\, \phi \;\big|\; \bigcirc_I\, \phi \;\big|\; \phi\, \mathbf{S}_I\, \phi \;\big|\; \phi\, \mathbf{U}_I\, \phi$$

- **Semantics:** $\bar{D} = (D_0, D_1, \dots)$ with $D_0, \dots \subseteq AP$, $\bar{\tau} = (\tau_0, \tau_1, \dots)$, and $i \in \mathbb{N}$

$$\begin{array}{lll}
(\bar{D}, \bar{\tau}, i) \models P & \text{iff} & P \in D_i \\
(\bar{D}, \bar{\tau}, i) \models \neg\phi & \text{iff} & (\bar{D}, \bar{\tau}, i) \not\models \phi \\
(\bar{D}, \bar{\tau}, i) \models \phi \vee \psi & \text{iff} & (\bar{D}, \bar{\tau}, i) \models \phi \text{ or } (\bar{D}, \bar{\tau}, i) \models \psi \\
 & & \\
(\bar{D}, \bar{\tau}, i) \models \bullet_I\, \phi & \text{iff} & i > 0,\ \tau_i - \tau_{i-1} \in I,\ \text{and } (\bar{D}, \bar{\tau}, i-1) \models \phi \\
(\bar{D}, \bar{\tau}, i) \models \bigcirc_I\, \phi & \text{iff} & \tau_{i+1} - \tau_i \in I \text{ and } (\bar{D}, \bar{\tau}, i+1) \models \phi \\
 & & \\
(\bar{D}, \bar{\tau}, i) \models \phi\, \mathbf{S}_I\, \psi & \text{iff} & \text{there is some } j \leq i \text{ with } \tau_i - \tau_j \in I,\ (\bar{D}, \bar{\tau}, j) \models \psi, \\
 & & \text{and } (\bar{D}, \bar{\tau}, k) \models \phi,\ \text{for all } k \text{ with } j < k \leq i \\
(\bar{D}, \bar{\tau}, i) \models \phi\, \mathbf{U}_I\, \psi & \text{iff} & \text{there is some } j \geq i \text{ with } \tau_j - \tau_i \in I,\ (\bar{D}, \bar{\tau}, j) \models \psi, \\
 & & \text{and } (\bar{D}, \bar{\tau}, k) \models \phi,\ \text{for all } k \text{ with } i \leq k < i
\end{array}$$

- **Syntactic Sugar:** $\blacklozenge_I\, \phi := \mathit{true}\, \mathbf{S}_I\, \phi$, $\blacksquare_I\, \phi := \neg \blacklozenge_I \neg\phi$, ...

Remarks on time model



- Zoo of temporal logics: CTL, LTL, PSL, ITL, MTL, TPTL, ...
  - Dedicated temporal operators; temporal reasoning restricted to a few cases
  - 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 (see [Basin et al. '11])
  - 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 '90]

$$\Box \forall e. \forall r.\, \mathit{publish\_report}(e, r) \rightarrow \blacklozenge_{\leq 10} \exists m.\, \underline{\mathit{manager}}(m, e) \wedge \mathit{approve\_report}(m, r)$$

- **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)

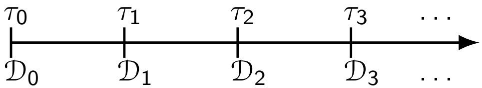
  C is a finite set of constant symbols and R is a finite set of predicates, each with an associated arity

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

$$\begin{array}{rcl} \phi & ::= & t_1 \approx t_2 \;\big|\; t_1 \prec t_2 \;\big|\; r(t_1, \ldots, t_n) \;\big|\; \exists x.\, \phi \;\big| \\ & & \neg\phi \;\big|\; \phi \vee \phi \;\big|\; \CIRCLE_I\, \phi \;\big|\; \Circle_I\, \phi \;\big|\; \phi \,\mathbf{S}_I\, \phi \;\big|\; \phi \,\mathbf{U}_I\, \phi \end{array}$$

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

Semantics



- A *temporal structure* (over S) is a pair $(\bar{\mathcal{D}}, \bar{\tau})$.
  - Sequence $\bar{\tau} = (\tau_0, \tau_1, \dots)$ of timestamps, $\tau_i \in \mathbb{N}$
    monotonically increasing and progressing
  - Sequence of structures $\bar{\mathcal{D}} = (\mathcal{D}_0, \mathcal{D}_1, \dots)$
    constant domains and rigid interpretation of constant symbols
- $(\bar{\mathcal{D}}, \bar{\tau}, v, i) \models \phi$ 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 part

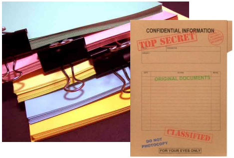
Differences to other FO monitoring approaches

- Temporal **past** and **future** operators
  As we will see, the operator **S** will be particularly handy
- **Fixed (infinite) domain** $|\bar{\mathcal{D}}|$
  But multiple (finite) events at each time point

$$(\mathsf{Alice}, 234) \in \mathit{approve\_report}^{\mathcal{D}_i} \quad \text{and}$$
$$(\mathsf{Bob}, 248), (\mathsf{Charlie}, 249) \in \mathit{publish\_report}^{\mathcal{D}_i}$$

- **Quantification**
  $\left(\bar{\mathcal{D}}, \bar{\tau}, v, i\right) \models \exists x.\, \phi \quad \text{iff} \qquad \left(\bar{\mathcal{D}}, \bar{\tau}, v[x \mapsto d], i\right) \models \phi,$ for some $d \in |\bar{\mathcal{D}}|$
  Alternatives:
  - freeze quantification ("half-order" [Henzinger '94])
  - guarded quantification [Garg et al. '11, Chowdhury et al. '14]
  - range restricted to data items occurring at current time point [Hallé&Villemaire '12, Bauer et al. '09]
- For monitoring, we will impose syntactic restrictions

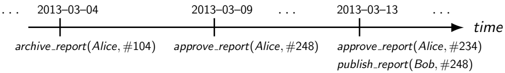
Policy revisited and simplified



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.*~~

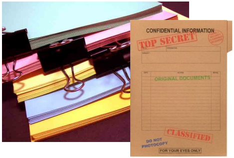
- Publishing and approving events are logged with time-stamps

```
    ⋮              ⋮
2013-03-04    archive_report (Alice, #104)
2013-03-04    approve_report (Alice, #248)
    ⋮              ⋮
2013-03-13    approve_report (Alice, #234)
              publish_report (Bob, #248)
    ⋮              ⋮
```



- Simplified policy in MFOTL:

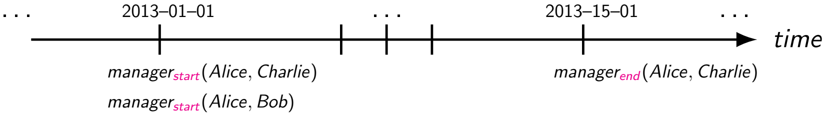
$$\Box\, \forall e.\, \forall r.\, \mathit{publish\_report}(e, r) \rightarrow \blacklozenge_{\leq 10}\, \exists m.\, \mathit{approve\_report}(m, r)$$



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
  - ∗ 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:

$$\Box \forall e. \forall r.\ publish\_report(e, r) \rightarrow$$
$$\blacklozenge_{\leq 10} \exists m.\ \underline{manager}(m, e) \wedge approve\_report(m, r)$$

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

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

(Assumption: X irreflexive and symmetric)

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

$$\Box\, \forall r.\, \forall r'.\, \underline{X}(r, r') \rightarrow \neg \exists s.\, \underline{roles}(s, r) \wedge \big(\neg S_{end}(s)\ \mathbf{S}\ \underline{roles}(s, r')\big)$$

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

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.

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

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):

$$\begin{aligned}\Box\, \forall s.\, \forall o.\, \forall d.\, \forall d'.\ & access(s,o) \wedge \underline{dataset}(o,d) \wedge \\ & \big(\exists o'.\, (\blacklozenge_{<4}\, access(s,o')) \wedge \underline{dataset}(o',d')\big) \ \rightarrow \\ & \neg \underline{conflict}(d,d')\end{aligned}$$

  Assume that:
  - At each time point, *conflict* is irreflexive and symmetric
  - At each time point, *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."*

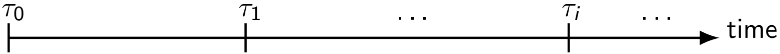
Monitoring

Monitoring Objective

- For a policy given as an **MFOTL formula** ϕ

$$\Box \forall c.\, \forall t.\, \forall a.\, \mathit{trans}(c,t,a) \wedge \mathit{th} < a \rightarrow \Diamond_{<6}\, \mathit{report}(t)$$

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



trans	cID	tID	amount
	Bob	#34	$100'000
	Eve	#37	$1'000

report	tID

trans	cID	tID	amount
	Eve	#45	$999'999

report	tID
	#34

trans	cID	tID	amount
	Bob	#78	$24
	Mallory	#99	$333'333

report	tID

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

Monitoring Objective

- For a policy given as an **MFOTL formula** ϕ

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

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

τ_0 — τ_1 — $\dots$ — τ_i — $\dots$ → time

At τ_0:

trans	cID	tID	amount
	Bob	#34	$100'000
	Eve	#37	$1'000

report	tID

At τ_1:

trans	cID	tID	amount
	Eve	#45	$999'999

report	tID
	#34

At τ_i:

trans	cID	tID	amount
	Bob	#78	$24
	Mallory	#99	$333'333

report	tID

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

Monitoring Objective

- For a policy given as an **MFOTL formula** ϕ

$$\Box \forall c.\, \forall t.\, \forall a.\, \mathit{trans}(c,t,a) \wedge \left(\blacklozenge_{<31} \exists t'.\, \exists a'.\, t \not\approx t' \wedge \mathit{trans}(c,t',a') \wedge \lozenge_{<6}\, \mathit{report}(t') \right) \\ \rightarrow \\ \lozenge_{<3}\, \mathit{report}(t)$$

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

τ_0 $\quad$ τ_1 $\quad$ ... $\quad$ τ_i $\quad$ ... $\quad$ time

τ_0:

trans	cID	tID	amount
	Bob	**#34**	**$100'000**
	Eve	#37	$1'000

report	tID

τ_1:

trans	cID	tID	amount
	Eve	#45	$999'999

report	tID
	#34

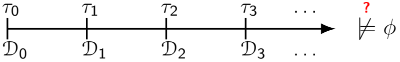
τ_i:

trans	cID	tID	amount
	Bob	**#78**	**$24**
	Mallory	#99	$333'333

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** ϕ must be of the form $\Box\,\phi'$
  - Temporal future operators in ϕ' only refer finitely into the future
    So ϕ describes an ***ω*-safety property**
  - Further restrictions on ϕ' to guarantee finiteness of intermediate results

$$r(x) \wedge \blacksquare_{<7} \neg q(x) \qquad \rightsquigarrow \qquad r(x) \wedge \neg \blacklozenge_{<7}\, q(x)$$

Related to domain independence of database queries
(see, e.g., [Fagin 1982])

Preprocessing: Negation and Rewriting

- Input formula ϕ

$$\square \forall t.\, \forall c.\, \forall a.\, \mathit{trans}(t, c, a) \wedge \big(\blacklozenge_{<31} \exists t'.\, \exists a'.\, t \not\approx t' \wedge \mathit{trans}(t', c, a') \wedge \lozenge_{<6}\, \mathit{report}(t') \big) \\ \rightarrow \\ \lozenge_{<3}\, \mathit{report}(t)$$

Preprocessing: Negation and Rewriting

- Input formula ϕ

$$\Box \forall t.\, \forall c.\, \forall a.\, \mathit{trans}(t,c,a) \land \left(\blacklozenge_{<31} \exists t'.\, \exists a'.\, t \not\approx t' \land \mathit{trans}(t',c,a') \land \Diamond_{<6}\, \mathit{report}(t') \right) \\ \rightarrow \\ \Diamond_{<3}\, \mathit{report}(t)$$

- Negate, rewrite, and drop outermost $\Diamond$ and $\exists$ quantifier(s), yielding ψ

$$\cancel{\Diamond \exists t.}\, \exists c.\, \exists a.\, \mathit{trans}(t,c,a) \land \left(\blacklozenge_{<31} \exists t'.\, \exists a'.\, t \not\approx t' \land \mathit{trans}(t',c,a') \land \Diamond_{<6}\, \mathit{report}(t') \right) \\ \land \\ \neg \Diamond_{<3}\, \mathit{report}(t)$$

Preprocessing: Negation and Rewriting

- Input formula ϕ

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

- Negate, rewrite, and drop outermost $\Diamond$ and $\exists$ quantifier(s), yielding ψ

$$\cancel{\Diamond \exists t.}\, \exists c.\, \exists a.\, \mathit{trans}(t,c,a) \wedge \big(\blacklozenge_{<31} \exists t'.\, \exists a'.\, t \not\approx t' \wedge \mathit{trans}(t',c,a') \wedge \Diamond_{<6}\, \mathit{report}(t') \big) \\ \wedge \\ \neg \Diamond_{<3}\, \mathit{report}(t)$$

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

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

These are the transactions that should have been reported at time point i

Preprocessing: Reduction to First-Order Queries

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

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

Preprocessing: Reduction to First-Order Queries

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

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

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

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

Preprocessing: Reduction to First-Order Queries

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

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

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

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

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

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

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

$$\left\{ \bar{a} \;\middle|\; (\hat{\mathcal{D}}_i, v[\bar{x}/\bar{a}]) \models \hat{\psi} \right\}$$

Preprocessing: Reduction to First-Order Queries

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

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

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

$$\exists c.\, \exists a.\, \mathit{trans}(t,c,a) \wedge p_{\alpha_2}(c,t) \wedge \neg p_{\alpha_3}(t)$$

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

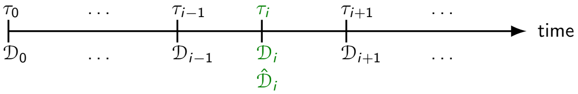
$$p_\alpha^{\hat{\mathcal{D}}_i} = \{\bar{a} \mid (\bar{\mathcal{D}}, \bar{\tau}, v[\bar{x}/\bar{a}], i) \models \hat{\alpha}\}$$

  - For each $i \in \mathbb{N}$, 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 construct $p_\alpha^{\hat{\mathcal{D}}_i}$ for each $i \in \mathbb{N}$

Constructing the Auxiliary Relations



- Construct auxiliary relations $p_\alpha^{\hat{\mathcal{D}}_i}$ inductively over α's formula structure and using also relations from both **previous** and **subsequent** structures
- Case where α has form $\bullet_I\, \beta$: $$p_\alpha^{\hat{\mathcal{D}}_i} = \begin{cases} \hat{\beta}^{\hat{\mathcal{D}}_{i-1}} & \text{if } i > 0 \text{ and } \tau_i - \tau_{i-1} \in I \\ \emptyset & \text{otherwise} \end{cases}$$
- Case where α has form $\bigcirc_I\, \beta$: $$p_\alpha^{\hat{\mathcal{D}}_i} = \begin{cases} \hat{\beta}^{\hat{\mathcal{D}}_{i+1}} & \text{if } \tau_{i+1} - \tau_i \in I \\ \emptyset & \text{otherwise} \end{cases}$$
  - ∗ Construction depends on relations in $\hat{\mathcal{D}}_{i+1}$ for which the predicates occur in $\hat{\beta}$
  - ∗ Monitor constructs $p_\alpha^{\hat{\mathcal{D}}_i}$ with a delay of at least one time step

Construction for $\mathbf{S}_{[0,\infty)}$

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

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

- 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 previous time point

- Constructions for metric $\mathbf{S}_I$ and $\mathbf{U}_I$ slightly more involved

Monitoring Algorithm

$i \leftarrow 0$ *% 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 $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$*

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

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 relation for α in $\hat{\mathcal{D}}_j$.

    Discard auxiliary relation 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 $q > 0$.

    $q \leftarrow q + 1$

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

  $i \leftarrow i + 1$ *% process next element in input sequence $(\mathcal{D}_{i+1}, \tau_{i+1})$*

end loop

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

Monitoring Algorithm

$i \leftarrow 0$ *% 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* $(\mathcal{D}_0, \tau_0), (\mathcal{D}_1, \tau_1), \ldots$
$Q \leftarrow \big\{ \big(\alpha, 0, \mathit{waitfor}(\alpha)\big) \,\big|\, \alpha \text{ temporal subformula of } \psi \big\}$
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 relation for α in $\hat{\mathcal{D}}_j$.
    Discard auxiliary relation 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 $q > 0$.
    $q \leftarrow q + 1$
  $Q \leftarrow \big\{ \big(\alpha, i+1, \mathit{waitfor}(\alpha)\big) \,\big|\, \alpha \text{ temporal subformula of } \psi \big\} \cup$
    $\big\{ \big(\alpha, j, \bigcup_{\alpha' \in \mathit{update}(S, \tau_{i+1} - \tau_i)} \mathit{waitfor}(\alpha')\big) \,\big|\, (\alpha, j, S) \in Q \text{ and } S \neq \emptyset \big\}$
  $i \leftarrow i + 1$ *% process next element in input sequence* $(\mathcal{D}_{i+1}, \tau_{i+1})$
end loop

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

Monitoring Algorithm

$i \leftarrow 0$ *% 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 relation for α in $\hat{\mathcal{D}}_j$.
        Discard auxiliary relation 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 relations $p_\alpha^{\hat{\mathcal{D}}_q}$ are built for all temporal subformulas α of ψ **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* $(\mathcal{D}_{i+1}, \tau_{i+1})$
end loop

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 $p(x) \wedge \bullet \neg q(x)$
  - In $(i+1)$st iteration, monitor constructs auxiliary relation $p^{\hat{\mathcal{D}}_i}_{\bullet \neg q(x)}$
- **Solution:** rewrite to a formula so that auxiliary relations are finite
  - $p(x) \wedge \bullet \neg q(x)$ is rewritten to $p(x) \wedge \bullet \big(\neg q(x) \wedge \bigcirc\, p(x)\big)$
  - 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 $p(x) \wedge \bullet \neg q(x)$
  - In $(i+1)$st iteration, monitor constructs auxiliary relation $p^{\hat{\mathcal{D}}_i}_{\bullet \neg q(x)}$
- **Solution:** rewrite to a formula so that auxiliary relations are finite
  - $p(x) \wedge \bullet \neg q(x)$ is rewritten to $p(x) \wedge \bullet \big(\neg q(x) \wedge \bigcirc p(x)\big)$
  - 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

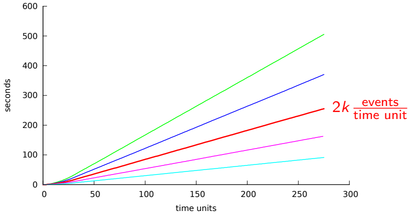
MONPOLY

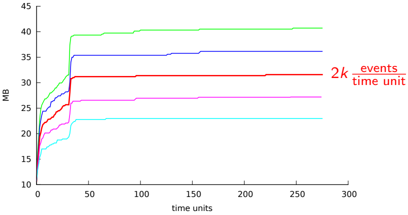
- Implementation of our monitoring algorithm for MFOTL
  - Usage: `monpoly -sig` *signature* `-formula` *policy* `-log` *logfile*
  - Output: policy violations
- Open source, GNU public license
  - Available at `http://sourceforge.net/projects/monpoly`
  - Written in OCaml
- Also handles policies with aggregations:

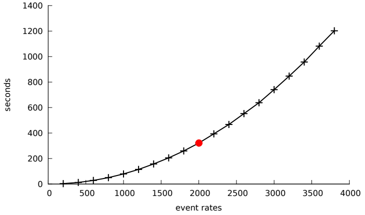
$$\Box\, \forall u.\, \forall s.\, \big[\mathrm{SUM}_a\ a, t.\ \blacklozenge_{<31}\ \mathit{withdraw}(u, t, a)\big](s; u) \rightarrow s \preceq 5000$$

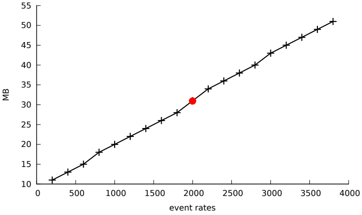
Performance Evaluation

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



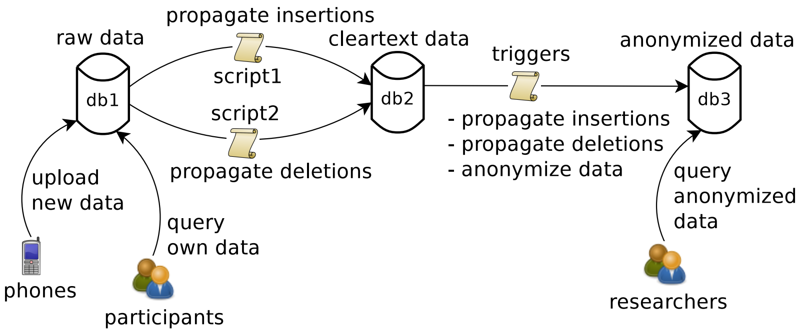






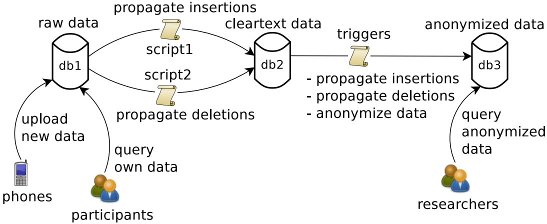
- PostgreSQL does not scale to larger log files

Case study: NOKIA's data-collection campaign



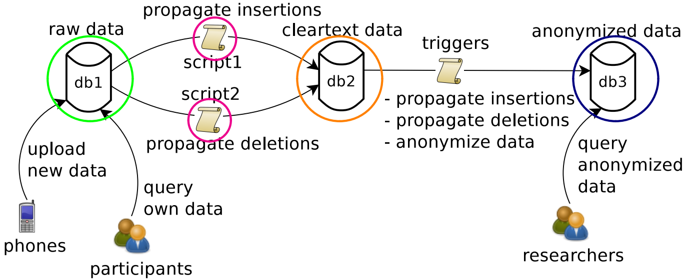
- Phone data collected and propagated to databases: location, call and SMS info, accelerometer, . . .
- Participants can view and delete their data
- Clear-text data used for personalized apps, e.g., location-history maps
- Anonymized data is used for research

Policies (sample)



1. Access-control rules restrict who accesses and modifies data in databases
   (A) Only user *script2* may delete data from *db2*
   (B) Databases *db1* and *db2* are accessed by *script1* account only while *script1* is running
2. Data changes are propagated between databases
   (C) Data deleted from *db2* is deleted from *db3* within 60 seconds
   (D) Data inserted into *db1* is, within 30 hours, either inserted into *db2* or deleted from *db1*

Logs



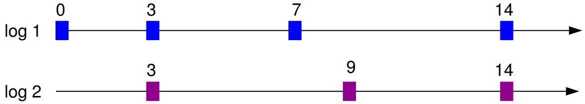
- Log entries are produced at multiple places
- Need to combine logs
- No total order on log entries
- Compliance might depend on order

Log sample

@unix time			
event	*db user*	*db*	*data id*
@1272902328			
insert	(eu.030,	db1,	146368038)
insert	(eu.031,	db2,	122368122)
@1272902355			
delete	(script2,	db2,	108031209)
select	(res.012,	db3,	146368038)
@1273158243			
script_end	(script1)		

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 ϕ
* prefixes $\bar{D}^1$ and $\bar{D}^2$ of length $n \geq 1$
  with $\bar{D}^i = (D_1^i, \tau_1)\,(D_1^i, \tau_1)\ \ldots\ (D_n^i, \tau_n)$, for $i \in \{1, 2\}$

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

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

Collapsed Logs

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

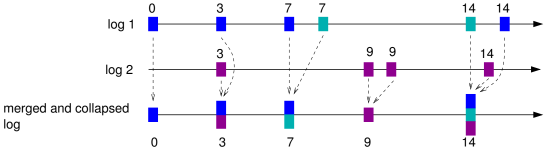
$$\Box\,\forall u.\,\forall d.\,\mathit{delete}(u, db2, d) \rightarrow \blacklozenge_{<1s}\,\Diamond_{<60s}\,\exists u'.\,\mathit{delete}(u', db3, d)$$

Collapsed Logs

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

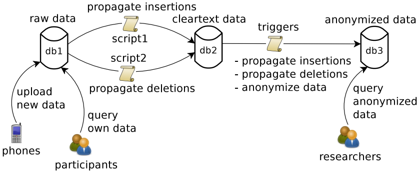
$$\Box\, \forall u.\, \forall d.\, delete(u, db2, d) \rightarrow \blacklozenge_{<1s} \Diamond_{<60s}\, \exists u'.\, delete(u', db3, d)$$

- 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)

policy	time / space
easiest	17 minutes / 14 MB
hardest	1 hour / 3.3 GB (mostly within 600 MB)

  - 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

Conclusion

Conclusion

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

- Logical methods are well suited for reasoning about policies

  MFOTL: expressive, yet monitoring practically feasible

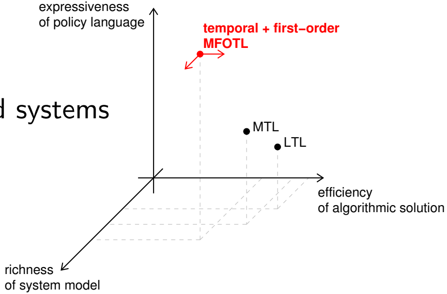
- Tool support publicly available
  MONPOLY at `http://sourceforge.net/projects/monpoly`
  including sanitized log data from NOKIA case study

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

Challenges

- **Scaling-up**
  How to monitor terabytes/petabytes of logged data?

- **Distributed monitoring (and enforcement)**
  How to (online) monitor distributed systems in a distributed way?
  What policies are enforceable in a concurrent setting?



- **Incomplete knowledge**
  How account for actions that are not logged (e.g., logging failures)?
  What if observations are contradictory or imprecise?

References

- David Basin, Matúš Harvan, Felix Klaedtke, and Eugen Zălinescu.
  Monitoring usage-control policies in distributed systems.
  IEEE Transactions on Software Engineering, 2013.
- David Basin, Matúš Harvan, Felix Klaedtke, and Eugen Zălinescu.
  MONPOLY: Monitoring usage-control policies.
  Proceedings of the 2nd International Conference on Runtime Verification (RV'11).
- David Basin, Felix Klaedtke, Eugen Zăalinescu.
  Algorithms for monitoring real-time properties
  Proceedings of the 2nd International Conference on Runtime Verification (RV'11).
- David Basin, Felix Klaedtke, and Samuel Müller.
  Policy monitoring in first-order temporal logic.
  Proceedings of the 22nd International Conference on Computer Aided Verification (CAV'10).
- David Basin, Felix Klaedtke, and Samuel Müller.
  Monitoring security policies with metric first-order temporal logic.
  Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT'10).
- David Basin, Felix Klaedtke, Samuel Müller, and Birgit Pfitzmann.
  Runtime monitoring of metric first-order temporal properties.
  Proceedings of the 28th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'08).