Margrave:
Query-Based Policy Analysis

Daniel J. Dougherty

Worcester Polytechnic Institute

joint work with:

Kathi Fisler (WPI)

Tim Nelson (WPI)

Shriram Krishnamurthi (Brown)

HCSS   Annapolis   May 11, 2012

the role of policy

Not about protecting against illegitimate or unauthorized uses of a system;

Rather, about making sure that legitimate uses achieve security goals.

Hence, complementary to studies of vulnerabilities and intrusion-detection.

Attacker model is: authorized use of the system!

a range of analyses

1. "Is it possible for someone in Department A to read (certain) files of Department B?"
   a boolean query

2. "What files are administrative assistants forbidden to delete?"
   a typical database query

3. Suppose P_1 is a policy, and P_2 is an update. "Are P_1 and P_2 equivalent?"
   policy comparison...logically complex!

4. "Which rule in my firewall is responsible for that packet being dropped?"

   reflection: rule-blaming

the policy/program split

Reflects reality of development: different authors; different concerns

Different specification formalisms: policy tends to be declarative

Allows for differences in analysis (this talk)

two analysis approaches

Entailment

given protocol or policy P and a verification property α, ask

$P \models \alpha$? does P entail α?

that is, is α true in all [protocol] runs of P?

Scenario-Finding

given protocol or policy P and certain assumptions, or ask

what scenarios are consistent with these?

that is, what are the *models* of $P +$ these observations?

Margrave $\Uparrow$

two analysis approaches

Entailment

given protocol or policy P and a verification property α, ask

$P \models \alpha$? does P entail α?

that is, is α true in all [protocol] runs of P?

Scenario-Finding

given protocol or policy P and certain assumptions, or ask

what scenarios are consistent with these?

that is, what are the *models* of $P +$ these observations?

Margrave $\Uparrow$

Margrave principles

Don't analyze policies in isolation:
policies exist as companions to systems

Scenario-finding:
prefer examples to proofs

Based on first-order logic:
value expressiveness over speed

Supports rigorous property-free analysis:
you shouldn't have to be a logician to use formal methods

an extended example: change-impact analysis

access-control policy differencing

a conference manager policy

Original policy P_1

...

- During the review phase, reviewer r may submit a review for paper p if r is not conflicted with p
- During the meeting phase, reviewer r can read the scores for paper p if r has submitted a review for p and r is not conflicted with p

...

Updated policy P_2

...

- During the review phase, reviewer r may submit a review for paper p if r is assigned to review p
- During the meeting phase, reviewer r can read the scores for paper p if r has submitted a review for p

...

access-control policy differencing

a conference manager policy

Original policy P_1

…

- During the review phase, reviewer r may submit a review for paper p if r is not conflicted with p
- During the meeting phase, reviewer r can read the scores for paper p if r has submitted a review for p and r is not conflicted with p

…

Updated policy P_2

…

- During the review phase, reviewer r may submit a review for paper p if r is assigned to review p
- During the meeting phase, reviewer r can read the scores for paper p if r has submitted a review for p

…

Change impact analysis

Can compute difference in pure policy semantics:

For user r to submit a review for paper p:

1. permitted in P_1 but not permitted in P_2:

   $$\mathsf{reviewPhase} \text{ and not } \mathsf{conflicted}(r,p) \text{ and not } \mathsf{assigned}(r,p)$$

2. permitted in P_2 but not permitted in P_1:

   $$\mathsf{reviewPhase} \text{ and } \mathsf{conflicted}(r,p) \text{ and } \mathsf{assigned}(r,p)$$

For user r to read scores for paper p:

1. permitted in P_1 but not permitted in P_2:

   never

2. permitted in P_2 but not permitted in P_1:

   $$\mathsf{meetingPhase} \text{ and } \mathsf{submitted}(r,p) \text{ and } \mathsf{conflicted}(r,p)$$

Change impact analysis

Can compute difference in pure policy semantics:

For user r to submit a review for paper p:

1. permitted in P_1 but not permitted in P_2:

   $\mathsf{reviewPhase}$ and not $\mathsf{conflicted}(r, p)$ and not $\mathsf{assigned}(r, p)$

2. permitted in P_2 but not permitted in P_1:

   $\mathsf{reviewPhase}$ and $\mathsf{conflicted}(r, p)$ and $\mathsf{assigned}(r, p)$

For user r to read scores for paper p:

1. permitted in P_1 but not permitted in P_2:

   never

2. permitted in P_2 but not permitted in P_1:

   $\mathsf{meetingPhase}$ and $\mathsf{submitted}(r, p)$ and $\mathsf{conflicted}(r, p)$

notes about the analysis

1. Analysis was about policy and program in cooperation:
   system state plays a key role :
   ReviewPhase $\succ$ MeetingPhase $\succ$
2. Interplay between rules for *assigning reviewers* and which user-processes can *read* and *submit reviews*. Latter is standard access-control; the former is a human process!
3. Permissions vary over time, but policy doesn't. Policy is a function

   $$P : (\ \text{ProgramState} \ \times \ \text{Requests}\ ) \rightarrow \ \text{Decisions}$$

   One policy, various program states
4. No deduction in user interface: the tool generated scenarios of interest, which a user can comprehend.

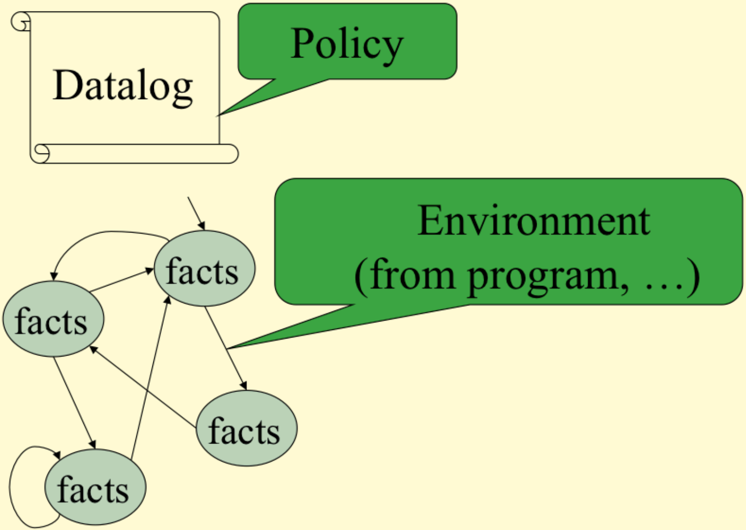
notes about the analysis cont'd

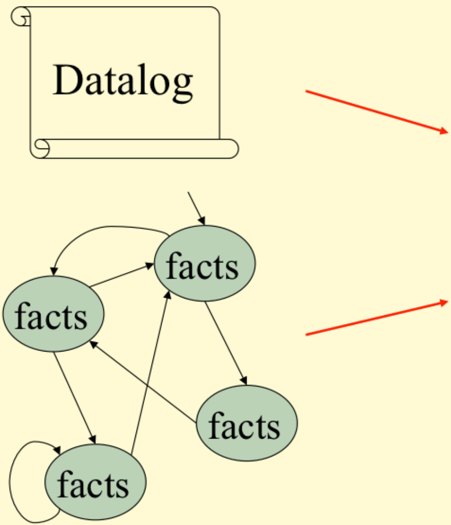
5. A semantically rich question was asked: "how do policies compare?" Not a simple "is a bad state reachable?" question.

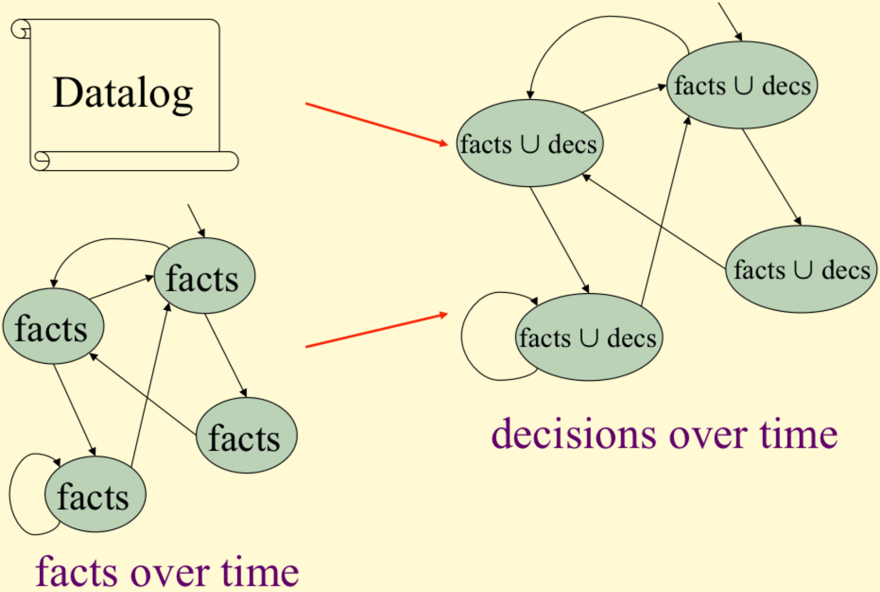
6. For such (datalog-based) policies, change-impact differences are computable..

7. Analysis "workflow" is subtle and interesting: pure policy analysis generated information suitable for pure program analysis.

8. ** Reduced a complex, non- "safety" problem to a reachability-analysis one. Uses mature tools from program-analysis community.


Datalog
Policy
Environment
(from program, …)
facts
facts
facts
facts
facts


Datalog
facts
facts
facts
facts


Datalog
facts
facts
facts
facts
facts ∪ decs
facts ∪ decs
facts ∪ decs
facts ∪ decs
facts over time
decisions over time

Margrave implementation

Passage to propositional logic

- Current version uses SAT solving (Kodkod)

Experimental version using first-order geometric logic

Crucial user aspects:

1. Try to spare user from giving domain bounds:
   compute sufficient domain sizes when possible.
2. Provide rich metaphor for exploring scenarios to gain insight into policies

further aspects

not this talk

- Traditional property-based verification is available:
  α is valid if and only if there are no scenarios for $\neg\alpha$.
- First-order foundations are essential for many analyses:
  go beyond firewalls, RBAC, XACML ...
  [FOSER 2010]
- Typical mode of use for Margrave: "what-if?" exploration user explores the space of scenarios for a given query
  [LISA 2010]
- Support for policy composition
  [Giannakopulous MS thesis 2012]
- For many queries, analysis is complete: sufficient bounds on scenario-sizes can be automatically computed
  [ABZ 2012]

which scenarios?

too many models!

Suppose P has some realizing scenarios. Which ones to report?

Informal intuitions

No unnecessary entities!

No gratuitous facts!

homomorphisms

Let $\mathbb{M}$ and $\mathbb{N}$ be models. A *homomorphism* from $\mathbb{M}$ to $\mathbb{N}$ is a function

$$h : |\mathbb{M}| \to |\mathbb{N}|$$

such that for all predicates R,

whenever	$R(a_1, \ldots, a_n)$	holds in $\mathbb{M}$
we have:	$R(h(a_1), \ldots, h(a_n))$	holds in $\mathbb{N}$

Intuition: "h transforms elements, preserving facts"

Geometric formulas:

built from atomic formulas using
finitary $\wedge$, infinitary $\bigvee$, and $\exists$

A geometric sequent looks like

$$q_1 \rightarrow q_2$$

where q_1, q_2 are geometric formulas

key features

If a geometric formula $q[a_1, \ldots, a_n]$ holds in a model $\mathbb{M}$ then

- there is a finite piece of the model witnessing this, and
- extensions to the model (new facts or new elements) will not disturb the truth of $q[\mathbf{a}]$

Geometric formulas are precisely those preserved by homomorphisms.

why geometric logic?

expressive: naturally captures protocol executions and security goals

admits forward-chaining inference mechanism: the Chase

very convenient for model-building: the Chase builds models *minimal* in the homomorphism partial order

a finite model theorem

Effectively Propositional Logic

the Bernays+Schoenfinkel+Ramsey class

Sentences of the form

$$\exists x_1 \ldots x_n \, \forall y_1 \ldots y_k \, . \, \varphi \qquad \qquad \varphi \text{ quantifier-free}$$

Theorem [Bernays+Schoenfinkel 1928, Ramsey 1930] If a sentence in the above class is satisfiable then it has a model of size bounded by n.

Corollary This class is decidable

Our goal Generalize this class

example

$\forall x\, \exists y\, \forall z\, .\, \varphi$ an undecidable prefix class for satisfiability

$\forall x^A\, \exists\, y^B\, \forall z^A\, .\, \varphi$ sorted version is better-behaved :

Assume $A \leq B$
Suppose have 2 constants at each sort
Then if there are any models at all then there is a model $\mathbb{M}$ with $|\mathbb{M}(A)| \leq 2$ and $|\mathbb{M}(B)| \leq 4$

But if $B \leq A$ instead: no such bounds

example

$\forall x\, \exists y\, \forall z\, .\, \varphi$ an undecidable prefix class for satisfiability

$\forall x^A\, \exists\, y^B\, \forall z^A\, .\, \varphi$ sorted version is better-behaved :

Assume $A \leq B$

Suppose have 2 constants at each sort

Then if there are any models at all then there is a model $\mathbb{M}$ with $|\mathbb{M}(A)| \leq 2$ and $|\mathbb{M}(B)| \leq 4$

But if $B \leq A$ instead: no such bounds

example

$\forall x \, \exists y \, \forall z \, . \, \varphi$ an undecidable prefix class for satisfiability

$\forall x^A \, \exists \, y^B \, \forall z^A \, . \, \varphi$ sorted version is better-behaved :

Assume $A \leq B$

Suppose have 2 constants at each sort

Then if there are any models at all then there is a model $\mathbb{M}$ with $|\mathbb{M}(A)| \leq 2$ and $|\mathbb{M}(B)| \leq 4$

But if $B \leq A$ instead: no such bounds

order-sorted signatures

Language

- as usual, but admit relation symbols
- semantics allows empty sorts

These are essentially tree automata.

Notation: write $\mathbb{T}^{\mathcal{L}}$ for the term model over signature $\mathcal{L}$.

main theorems

Given σ we can Skolemize to get a *universal* $\sigma_\forall$.

Theorem. Let σ be a sentence whose Skolemization $\sigma_\forall$ has signature $\mathcal{L}$. Then σ is satisfiable if and only if σ has a model $\mathbb{M}$ such that for each sort A, $|\mathbb{M}(A)| \leq |\mathbb{T}^{\mathcal{L}}(A)|$.

That is, it suffices to count the size of the language of the tree automaton.

Theorem. We can decide whether $\mathbb{T}^{\mathcal{L}}(A)$ is finite, uniformly for each sort A, in time linear in $\mathcal{L}$.

Theorem. We can compute $|\mathbb{T}^{\mathcal{L}}(A)|$, uniformly for each sort A, in time cubic in $\mathcal{L}$.

Given σ we can Skolemize to get a *universal* $\sigma_\forall$.

Theorem. Let σ be a sentence whose Skolemization $\sigma_\forall$ has signature $\mathcal{L}$. Then σ is satisfiable if and only if σ has a model $\mathbb{M}$ such that for each sort A, $|\mathbb{M}(A)| \leq |\mathbb{T}^{\mathcal{L}}(A)|$.

That is, it suffices to count the size of the language of the tree automaton.

Theorem. We can decide whether $\mathbb{T}^{\mathcal{L}}(A)$ is finite, uniformly for each sort A, in time linear in $\mathcal{L}$.

Theorem. We can compute $|\mathbb{T}^{\mathcal{L}}(A)|$, uniformly for each sort A, in time cubic in $\mathcal{L}$.

technicalities

Classical (unsorted) treatment . . . challenges for order-sorting:

1. By Skolemization, build a universal sentence $\sigma_\forall$ equi-satisfiable with σ;

   When empty sorts are allowed, the Skolem form of σ is not equisatisfiable with σ

2. Any model for $\sigma_\forall$ has a *Skolem hull:* close the interpretation of the constants by the interpretation of the functions.

   When sorts are not disjoint the Skolem hull of $\mathbb{M}$ can be infinite even when term model is finite.

3. The truth of universal sentences is preserved under submodel.

   When sort names can be used as predicates—as in many tools—preservation of universal sentences under submodel fails

So when $\mathbb{T}^{\mathcal{L}}$ is finite (i.e., the B+S+R class) conclude the finite model theorem.

technicalities

Classical (unsorted) treatment . . . challenges for order-sorting:

1. By Skolemization, build a universal sentence $\sigma_\forall$ equi-satisfiable with σ;

   When empty sorts are allowed, the Skolem form of σ is not equisatisfiable with σ

2. Any model for $\sigma_\forall$ has a *Skolem hull:* close the interpretation of the constants by the interpretation of the functions.

   When sorts are not disjoint the Skolem hull of $\mathbb{M}$ can be infinite even when term model is finite.

3. The truth of universal sentences is preserved under submodel.

   When sort names can be used as predicates—as in many tools—preservation of universal sentences under submodel fails

So when $\mathbb{T}^{\mathcal{L}}$ is finite (i.e., the B+S+R class) conclude the finite model theorem.

technicalities

Classical (unsorted) treatment . . . challenges for order-sorting:

1. By Skolemization, build a universal sentence $\sigma_\forall$ equi-satisfiable with σ;

   When empty sorts are allowed, the Skolem form of σ is not equisatisfiable with σ

2. Any model for $\sigma_\forall$ has a *Skolem hull:* close the interpretation of the constants by the interpretation of the functions.

   When sorts are not disjoint the Skolem hull of $\mathbb{M}$ can be infinite even when term model is finite.

3. The truth of universal sentences is preserved under submodel.

   When sort names can be used as predicates—as in many tools—preservation of universal sentences under submodel fails

So when $\mathbb{T}^{\mathcal{L}}$ is finite (i.e., the B+S+R class) conclude the finite model theorem.

technicalities

Classical (unsorted) treatment . . . challenges for order-sorting:

1. By Skolemization, build a universal sentence $\sigma_\forall$ equi-satisfiable with σ;

   When empty sorts are allowed, the Skolem form of σ is not equisatisfiable with σ

2. Any model for $\sigma_\forall$ has a *Skolem hull:* close the interpretation of the constants by the interpretation of the functions.

   When sorts are not disjoint the Skolem hull of $\mathbb{M}$ can be infinite even when term model is finite.

3. The truth of universal sentences is preserved under submodel.

   When sort names can be used as predicates—as in many tools—preservation of universal sentences under submodel fails

So when $\mathbb{T}^{\mathcal{L}}$ is finite (i.e., the B+S+R class) conclude the finite model theorem.

technicalities

Classical (unsorted) treatment ... challenges for order-sorting:

1. By Skolemization, build a universal sentence $\sigma_\forall$ equi-satisfiable with σ;

   When empty sorts are allowed, the Skolem form of σ is not equisatisfiable with σ

2. Any model for $\sigma_\forall$ has a *Skolem hull:* close the interpretation of the constants by the interpretation of the functions.

   When sorts are not disjoint the Skolem hull of $\mathbb{M}$ can be infinite even when term model is finite.

3. The truth of universal sentences is preserved under submodel.

   When sort names can be used as predicates—as in many tools—preservation of universal sentences under submodel fails

So when $\mathbb{T}^{\mathcal{L}}$ is finite (i.e., the B+S+R class) conclude the finite model theorem.

OS-EPL

Definition *Order-Sorted Effectively Propositional Logic (OS-EPL)* is the class of sentences σ such that each $|\mathbb{T}^{\mathcal{L}}(A)|$ is finite (where $\mathcal{L}$ is the language of the Skolemization of σ).

Empirical fact: a wide class of policy queries lies in OS-EPL

Theorem: Membership in OS-EPL is decidable in linear time

Theorem: Model-size bounds can be computed in cubic time

Corollary: OS-EPL is decidable.

Application: Bounds-checking is incorporated into Margrave

Margrave principles

Don’t analyze policies in isolation:
policies exist as companions to systems

Scenario-finding:
prefer examples to proofs

Based on first-order logic:
value expressiveness over speed

Supports rigorous property-free analysis:
you shouldn’t have to be a logician to use formal methods