Imperial College London

Access Control via Belnap Logic: Intuitive, Expressive, and Analyzable Policy Composition

Michael Huth (joint work with Glenn Bruns, Bell Labs)

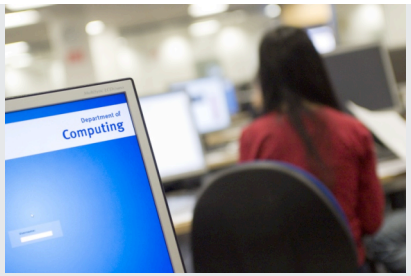
Imperial College
London

Outline of Access Control talk

- Motivation
- Belnap Logic
- Core Policy Language
- Expressiveness of Core Language
- Policy Analysis
- Conclusions

Imperial College London

Motivation



Access control in IT systems increasingly relies on ability to compose policies

Composition framework should

- be intuitive, formal, expressive, and analyzable
- be independent of application domains but extendable to such domains
- support change management, separation of concerns, and reuse

We develop here such a framework based on Belnap Logic

Belnap Logic used in the past for reasoning in Artificial Intelligence

Imperial College London

$$p \vee (\neg(p \vee -p) \wedge q)$$

Belnap Logic

In the 1970ies, Belnap suggested the use of a four-valued logic

- Ordinary truth values for truth and falsity
- A third truth value that expresses lack of knowledge
- And a fourth truth value that expresses inconsistent knowledge

He developed a semantics and a sound and complete Hilbert style proof system for this logic

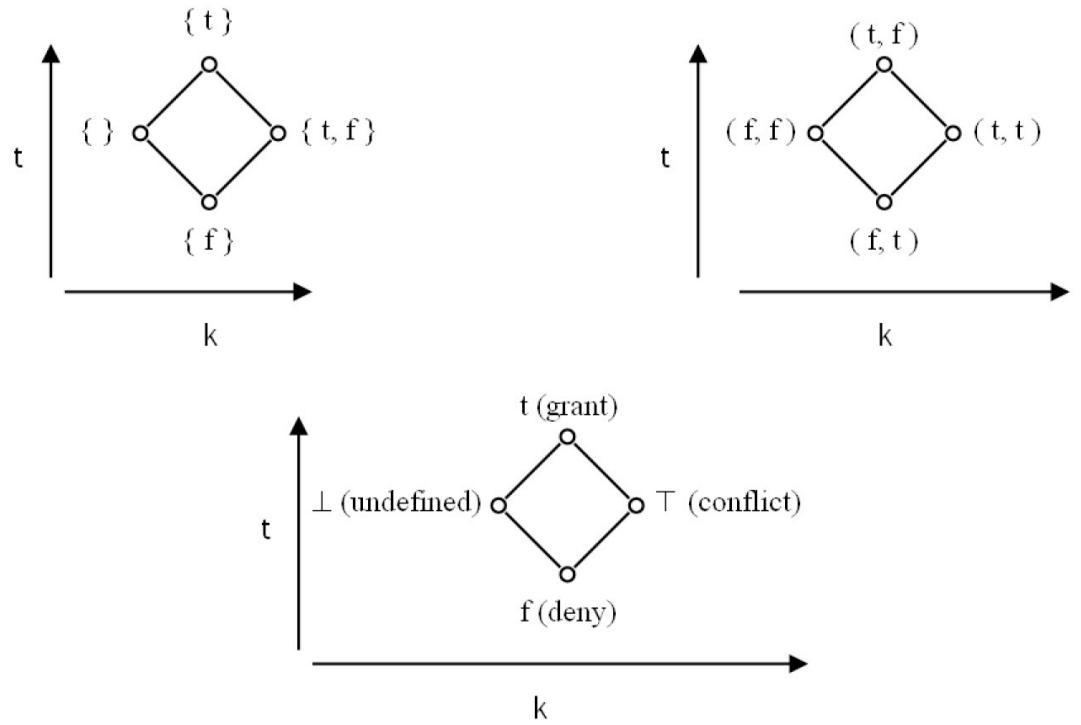
Belnap logic extends naturally to first- and higher-order logics

Key idea for us: Belnap's evidence-based notion of truth, e.g.

- conjunction of ''don't know'' and ''false'' is ''false''
- but conjunction of ''don't know'' and ''true'' is ''don't know''

Imperial College
London

Belnap Logic: four values as composition of two



{..} collects results of policies p and q, e.g. {t,f} as conflict

(.,.) asks ``does single policy p (grant?,deny?)", e.g. (t,t) means p grants and denies

Third graph captures both interpretations abstractly

Imperial College London

Belnap space $(4, \leq_t, \leq_k, \neg)$

Belnap bilattice over $\mathbf{4} = \{\mathbf{t}, \mathbf{f}, \top, \bot\}$

Axioms:

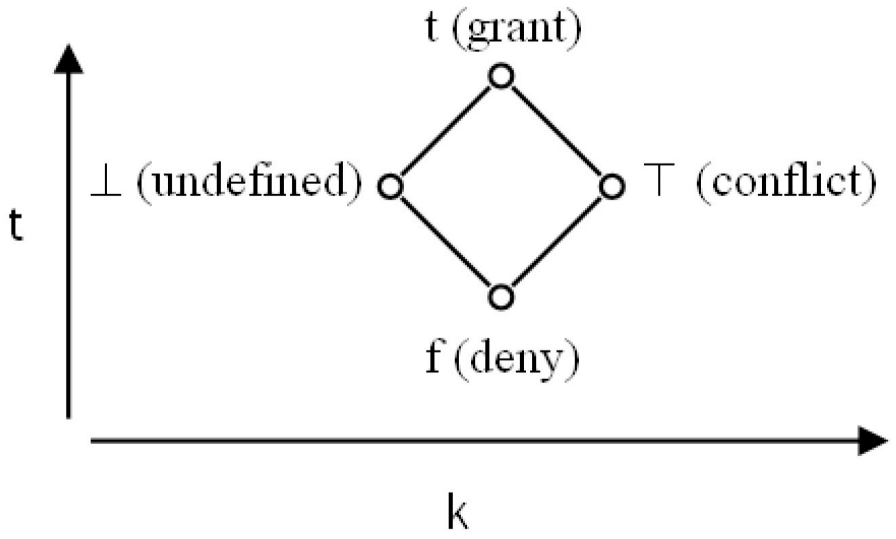
$$x \leq_t y \Rightarrow \neg y \leq_t \neg x,$$

$$x \leq_k y \Rightarrow \neg x \leq_k \neg y, \ and$$

$$\neg\neg x = x.$$

Truth negation $\neg$ swaps denials and grants, and leaves other two values fixed.

x-axis: knowledge ordering
y-axis: truth ordering



Imperial College London

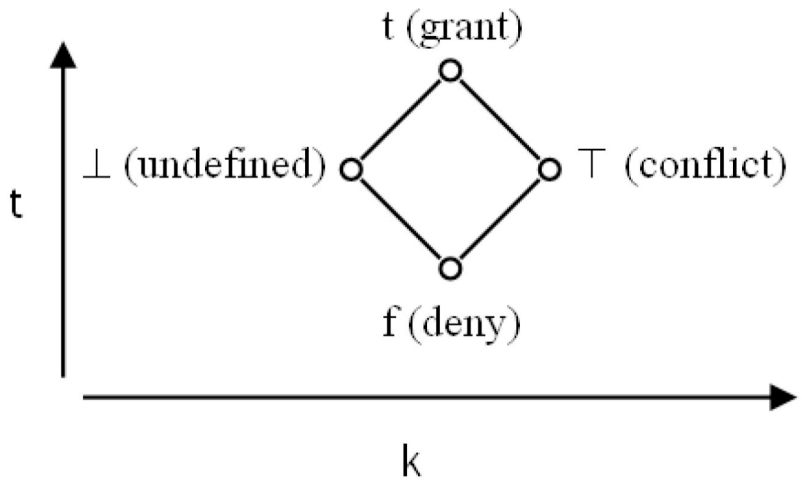
$\{\top, \neg, \wedge, \supset\}$

Belnap space functionally complete

Implication: $a \supset b = b$ if $a \in \{\mathbf{t}, \top\}$

$a \supset b = \mathbf{t}$ otherwise

Conjunction: $\wedge$ meet (aka infimum) in truth ordering



Imperial College
London

Policy Language

Core language PBel, pronounced "pebble"

rp ::= *Request Predicate*

a	Atomic
true	Truth
false	Falsity

p, p' ::= *Policy*

b if rp	Basic policy
$\top$	Conflict
$\neg p$	Logical negation
$p \wedge p'$	Logical meet
$p \supset p'$	Implication

$\top$ *overloaded: denotes policy and denotes element of Belnap space*

Atomic request predicates a *denote sets of access requests.*

$b \in \{\mathbf{t}, \mathbf{f}\}$

Imperial College
London

Syntactic sugar

PBel is a core language, similar to byte code for higher-level languages.

Convenient policy composition operators map (syntactically and semantically) into this core language, e.g.

$$
\begin{array}{lcl}
\mathbf{t} & = & \mathbf{t} \text{ if true} \\
\mathbf{f} & = & \neg\mathbf{t} \\
p \vee q & = & \neg(\neg p \wedge \neg q) \\
p \oplus q & = & (p \wedge \top) \vee (q \wedge \top) \vee (p \wedge q) \\
p[\mathbf{f} \mapsto q] & = & p \vee (\neg(p \vee -p) \wedge q) \\
p[\bot \mapsto q] & = & p \oplus (-(p \oplus \neg p) \otimes q) \\
p \text{ if } rp & = & p \otimes ((\mathbf{t} \text{ if } rp) \oplus (\mathbf{f} \text{ if } rp)) \\
p{\downarrow} & = & p[\top \mapsto \mathbf{f}][\bot \mapsto \mathbf{f}]
\end{array}
$$

$$
\begin{array}{lcl}
\bot & = & \mathbf{t} \text{ if false} \\
\top & & \text{already in PBel} \\
p \otimes q & = & (p \wedge \bot) \vee (q \wedge \bot) \vee (p \wedge q) \\
-p & = & (\neg p \supset \bot) \oplus (\neg(p \supset \bot)) \\
p[\mathbf{t} \mapsto q] & = & p \wedge (\neg(p \wedge -p) \vee q) \\
p[\top \mapsto q] & = & p \otimes (-(p \otimes \neg p) \oplus q) \\
p : q & = & (p \supset q) \otimes \neg(p \supset \neg q) \\
p{\uparrow} & = & p[\top \mapsto \mathbf{t}][\bot \mapsto \mathbf{t}]
\end{array}
$$

Imperial College
London

Models

Access-control model $\mathcal{M}$

consists of non-empty set of requests $\mathsf{R}_{\mathcal{M}}$

and interpretations of request predicates: $rp^{\mathcal{M}} \subseteq \mathsf{R}_{\mathcal{M}}$

$$\mathsf{true}^{\mathcal{M}} = \mathsf{R}_{\mathcal{M}} \ \ and \ \ \mathsf{false}^{\mathcal{M}} = \{\}$$

Example:

- set of requests as triples of form (subject,object, action, context)
- interpretation of atom *manager* is all triples whose subjects are managers
- interpretation of atom *lowThreat* is all triples whose context represents a low threat level

Imperial College
London

Semantics

$$
\begin{aligned}
[\![b \text{ if } rp]\!]_{\mathcal{M}}(r) &= \begin{cases} b & \text{if } r \in rp^{\mathcal{M}} \\ \bot & \text{otherwise} \end{cases} \\
[\![\top]\!]_{\mathcal{M}}(r) &= \top \\
[\![\neg p]\!]_{\mathcal{M}}(r) &= \neg [\![p]\!]_{\mathcal{M}}(r) \\
[\![p \wedge q]\!]_{\mathcal{M}}(r) &= [\![p]\!]_{\mathcal{M}}(r) \wedge [\![q]\!]_{\mathcal{M}}(r) \\
[\![p \supset q]\!]_{\mathcal{M}}(r) &= [\![p]\!]_{\mathcal{M}}(r) \supset [\![q]\!]_{\mathcal{M}}(r)
\end{aligned}
$$

Meaning of policy p in model M maps requests r to element of Belnap space

Key point: policy composition is pointwise extension of Belnap operators

Imperial College
London

Support for composite request predicates

Propositional logic structure on request predicates compiles into PBel, e.g.

$$(\mathbf{t} \text{ if } (\text{Manager} \wedge \text{OnDuty} \wedge \neg\text{Weekend} \wedge \text{ReadPDF})) > \mathbf{f}$$

is translated into PBel with function T below, for semantics on the left:

$$\begin{aligned}
[\![\mathsf{a}]\!]_{\mathcal{M}} &= \mathsf{a}^{\mathcal{M}} \\
[\![\text{true}]\!]_{\mathcal{M}} &= \mathsf{R}_{\mathcal{M}} \\
[\![\text{false}]\!]_{\mathcal{M}} &= \{\} \\
[\![\neg cp]\!]_{\mathcal{M}} &= \mathsf{R}_{\mathcal{M}} \setminus [\![cp]\!]_{\mathcal{M}} \\
[\![cp \wedge cp']\!]_{\mathcal{M}} &= [\![cp]\!]_{\mathcal{M}} \cap [\![cp']\!]_{\mathcal{M}} \\
[\![cp \vee cp']\!]_{\mathcal{M}} &= [\![cp]\!]_{\mathcal{M}} \cup [\![cp']\!]_{\mathcal{M}}
\end{aligned}$$

$$\begin{aligned}
T(\mathbf{t} \text{ if } \neg cp) &= T(\mathbf{t} \text{ if } cp) \supset \bot \\
T(\mathbf{t} \text{ if } cp \wedge cp') &= T(\mathbf{t} \text{ if } cp) \wedge T(\mathbf{t} \text{ if } cp') \\
T(\mathbf{t} \text{ if } cp \vee cp') &= \neg(T(\mathbf{f} \text{ if } cp) \wedge T(\mathbf{f} \text{ if } cp'))
\end{aligned}$$

$$\begin{aligned}
T(\mathbf{f} \text{ if } \neg cp) &= \neg(T(\mathbf{t} \text{ if } cp) \supset \bot) \\
T(\mathbf{f} \text{ if } cp \wedge cp') &= \neg(T(\mathbf{t} \text{ if } cp) \wedge T(\mathbf{t} \text{ if } cp')) \\
T(\mathbf{f} \text{ if } cp \vee cp') &= T(\mathbf{f} \text{ if } cp) \wedge T(\mathbf{f} \text{ if } cp')
\end{aligned}$$

Imperial College
London

Safe sublanguages

Policy p is conflict-free if it never returns $\top$ for any request in any model.

Sublanguage in which all and only conflict-free policies can be written:

p, q ::=	*Conflict-free policy*
b if rp	Basic policy
$\bot$	Gap
$\neg p$	Logical negation
$p \wedge q$	Logical meet
$\mathbf{r} \supset q$	Implication
$p \vee q$	Logical Join
$p \otimes q$	Knowledge meet

$\mathbf{r}[\top \mapsto p]$	Conflict resolution
$p[v \mapsto q]$	Sequential composition
p if rp	Generalized basic policy
$p : q$	Guard connective
$\mathbf{r}\downarrow$	Pessimistic wrapper
$\mathbf{r}\uparrow$	Optimistic wrapper

Boldface r denotes any policy expression of PBel

Imperial College
London

Safe sublanguage for gap-freedom

Policy p is gap-free if it never returns $\bot$ for any request in any model.

Sublanguage in which all and only gap-free policies can be written:

p, q ::=	*Gap-free policy*
b if true	Gap-free basic policy
$\top$	Conflict
$\neg p$	Logical negation
$p \wedge q$	Logical meet
$\mathbf{r} \supset q$	Implication
$p \vee q$	Logical join
$\mathbf{r} \oplus q$	Left knowledge join
$p \oplus \mathbf{r}$	Right knowledge join
$\mathbf{r}[\bot \mapsto p]$	Gap resolution
$p[v \mapsto q]$	Sequential composition
p if true	Generalized basic policy
$\mathbf{r}\downarrow$	Pessimistic wrapper
$\mathbf{r}\uparrow$	Optimistic wrapper

Boldface r denotes any policy expression of PBel

Imperial College
London

Safe sublanguage for conclusive decisions

Sublanguage in which only (and all) policies can be written that are gap-free and conflict-free:

p, q ::=	*Conclusive policy*
b if true	Gap-free basic policy
$\neg p$	Logical negation
$p \wedge q$	Logical meet
$\mathbf{r} \supset q$	Implication
$p \vee q$	Logical join
$p[v \mapsto q]$	Sequential composition
p if rp	Generalized basic policy
$\mathbf{r}\downarrow$	Pessimistic wrapper
$\mathbf{r}\uparrow$	Optimistic wrapper

Boldface r denotes any policy expression of PBel

Imperial College
London

Retrofitting policies

Abbreviation for priority composition: $p > q \;=\; p[\bot \mapsto q]$

Exceptional override: $(\mathbf{f} \text{ if } rp_{exc}) > p$

Behaves like policy p except at exceptional request set rp_{exc} *at which it denies*

Exclusive rights and exclusive prohibitions: $((\mathbf{t} \text{ if } rp_1) \oplus (\mathbf{f} \text{ if } rp_2)) > p$

Behaves like p except at two (assumed to be disjoint) request sets that encode absolute rights and absolute prohibitions (respectively)

Imperial College
London

Composing request predicates as policies

$$(\mathbf{t} \text{ if } ChW) \wedge (\mathbf{t} \text{ if } RegisteredAnalyst) > \mathbf{f}$$

Might model that an access is granted if

- *the requester is a registered analyst*
- *and if the request is compliant with a Chinese Wall policy*

Otherwise, the request is denied.

Note: ChW is a request predicate that presumably stems from a policy.

PBel supports such demotions of policies to predicates (see analysis part below).

Imperial College
London

Expressiveness of PBel: policy functions

Policy function for model $\mathcal{M}$ is total function $f : \mathsf{R}_{\mathcal{M}} \to \mathbf{4}$

Policy p expresses policy function f if $[\![p]\!]_{\mathcal{M}} = f$

Policy functions of form $[\![p]\!]_{\mathcal{M}} = f$ have the same output for requests that cannot be distinguished by any request predicate in p

It turns out that all functions of that form are expressible as meanings of policies in PBel.

These results customize to the safe sublanguages for gap-free, conflict-free, and conclusive policies (not shown in this talk).

Imperial College
London

Expressing data-independent policy functions

Let R be a set of request predicates, e.g. those occurring in a policy p.

On each model $\mathcal{M}$ *we define equivalence relation* $\equiv^R_{\mathcal{M}}$ *to be* $\{(r, r') \in \mathsf{R}_{\mathcal{M}} \times \mathsf{R}_{\mathcal{M}} \mid \forall \mathsf{a} \in R\colon r \in \mathsf{a}^{\mathcal{M}} \text{ iff } r' \in \mathsf{a}^{\mathcal{M}}\}$.

A policy function $f\colon \mathsf{R}_{\mathcal{M}} \to \mathbf{4}$ *is* data-independent for R *in* $\mathcal{M}$ *iff* $r \equiv^R_{\mathcal{M}} r'$ *implies* $f(r) = f(r')$.

We write $PBel^R$ *for the set of PBel policies that contain only atoms from* R. *In particular,* $PBel^{\mathsf{AP}}$ *equals PBel.*

E.g. $(\mathbf{t} \text{ if } ChW) \wedge (\mathbf{t} \text{ if } RegisteredAnalyst) > \mathbf{f}$ *has four equivalence classes*

Result:

For each p *in* $PBel^R$, *policy function* $[\![p]\!]_{\mathcal{M}}$ *is data-independent for* R *in* $\mathcal{M}$.

Conversely, let f *be a policy function that is data-independent for* R *in* $\mathcal{M}$ *for finite set* $\mathsf{R}_{\mathcal{M}}$. *Then there is some* p *in* $PBel^R$ *with* $f = [\![p]\!]_{\mathcal{M}}$.

Imperial College London

Proof that data-independent function f is expressible

Policy p expressing f is knowledge join of a grant and a denial part:

$$p = p_{\mathbf{t}} \oplus p_{\mathbf{f}}$$

Each part is the n-ary knowledge join of (negations of) policies p_r

$$p_{\mathbf{t}} = \sum_{r \in \mathsf{R}_{\mathcal{M}} \mid \mathbf{t} \leq_k f(r)} p_r \qquad p_{\mathbf{f}} = \sum_{r \in \mathsf{R}_{\mathcal{M}} \mid \mathbf{f} \leq_k f(r)} \neg p_r$$

Each building block p_r *is a characteristic function that grants on the equivalence class* $\equiv^R_{\mathcal{M}}$ *of request r and is undefined otherwise:*

$$p_r = \Big(\bigwedge_{\mathsf{a} \in R \mid r \in \mathsf{a}^{\mathcal{M}}} \mathbf{t} \text{ if a} \Big) \wedge \Big(\bigwedge_{\mathsf{a} \in R \mid r \notin \mathsf{a}^{\mathcal{M}}} (\mathbf{t} \text{ if a}) \supset \bot \Big)$$

Imperial College
London

Expressiveness of PBel: policy composition

Example: majority vote of three policies should make as decision the majority of decisions

$$G(p_1, p_2, p_3) = (p_1 \wedge p_2) \vee (p_1 \wedge p_3) \vee (p_2 \wedge p_3)$$

Our pointwise composition means that any such function G is determined by a function

$$g\colon \mathbf{4}^n \to \mathbf{4}$$

G(p1,…,pn) at request r decides g(v1,…,vn) where vi is decision of pi on r

Imperial College London

All composition functions are expressible

Free algebra $\mathcal{A}$
generated from operators $\{\top, \neg, \wedge, \supset\}$
and variables $X_1, X_2, \ldots$

Result:

$$Let\ n \geq 0\ and\ g \in \mathbf{4}^n \rightarrow \mathbf{4}$$
$$Then\ there\ is\ a\ term\ t_g \in \mathcal{A}\ such\ that$$
$$[\![t_g(p_1, \ldots, p_n)]\!]\mathcal{M}(r) = g([\![p_1]\!]\mathcal{M}(r), \ldots, [\![p_n]\!]\mathcal{M}(r))$$
$$(\forall p_i \in PBel, \mathcal{M}, r \in \mathsf{R}_\mathcal{M})$$

Example: term for knowledge join $\oplus$ is

$$\neg(\neg(X_1 \wedge \top) \wedge \neg(\neg((\top \wedge X_2) \wedge \neg(X_1 \wedge X_2))))$$

Imperial College
London

Policy Analysis

We develop a simple query language.

Many important policy analyses are expressible as queries in this language.

Here, we ask whether queries hold in all models: validity checking.

Validity checking is appropriate, e.g. for gap and conflict analysis.

Queries can be implications whose antecedents encode domain-specific or other assumptions.

This policy analysis reduces to validity checking of propositional logic.

Imperial College
London

Query Language

cp, cp' ::= *Composite Request Predicate*

a	Atomic	$\neg cp$	Negation
true	Truth	$cp \wedge cp'$	Conjunction
false	Falsity	$cp \vee cp'$	Disjunction

ϕ, ϕ' ::= *Query*

$p \leq_t q$	Policy refinement in truth ordering
$p \leq_k q$	Policy refinement in information ordering
$\alpha \Rightarrow \phi$	Assume-guarantee query
$\phi \wedge \phi'$	Conjunction

where assumption α *ranges over composite request predicates*

Imperial College
London

Query Examples

Policy q is more defined and less permissive than p

$$(p \leq_k q) \wedge (p \leq_t q)$$

Policy q is more defined but less permissive than p

$$(p \leq_k q) \wedge (q \leq_t p)$$

Policies p and q are equivalent

$$(p \leq_t q) \wedge (q \leq_t p)$$

Policy p has no gaps $p \leq_t p[\bot \mapsto \mathbf{f}]$

Policy p has no conflicts $p \leq_k p[\top \mapsto \mathbf{f}]$

Imperial College
London

Query Semantics

$$\mathcal{M} \models p \leq_k q \text{ iff for all } r \in \mathsf{R}_{\mathcal{M}} \text{ we have } [\![p]\!]_{\mathcal{M}}(r) \leq_k [\![q]\!]_{\mathcal{M}}(r)$$
$$\mathcal{M} \models p \leq_t q \text{ iff for all } r \in \mathsf{R}_{\mathcal{M}} \text{ we have } [\![p]\!]_{\mathcal{M}}(r) \leq_t [\![q]\!]_{\mathcal{M}}(r)$$
$$\mathcal{M} \models \alpha \Rightarrow \phi \text{ iff } [\![\alpha]\!]_{\mathcal{M}} \neq \mathsf{R}_{\mathcal{M}} \text{ or } \mathcal{M} \models \phi$$
$$\mathcal{M} \models \phi \wedge \psi \text{ iff } \mathcal{M} \models \phi \text{ and } \mathcal{M} \models \psi$$

Atomic queries are interpreted universally: policy p is below policy q for all requests

Implication is modeled universally: if all requests of the model satisfy assumption, then the guarantee query has to be true in the model.

Conjunction has standard interpretation.

Imperial College
London

Query Analysis

For each query ϕ we generate a composite request predicate $C(\phi)$ such that

Query ϕ is valid in all models

Iff

$C(\phi)$ is valid when interpreted as formula in propositional logic

Definition of $C(\phi)$ proceeds in two steps:

1. Capture constraints for grants and denials of policies contained in query
2. Capture the logical structure of the query in terms of these policy constraints

Imperial College
London

Constraints for PBel policies

Each request r in model M determines model of propositional logic:

$$\rho_r^{\mathcal{M}}(\mathsf{a}) = \mathbf{t} \;\; if \; r \in \mathsf{a}^{\mathcal{M}} \qquad \rho_r^{\mathcal{M}}(\mathsf{a}) = \mathbf{f} \;\; if \; r \notin \mathsf{a}^{\mathcal{M}}$$

Semantics of that model matches that of request model M: $\rho_r^{\mathcal{M}}$
satisfies those composite request predicates that contain r in their interpretation

Define, below, propositional logic formula $p \Uparrow b$ *such that*

$$\rho_r^{\mathcal{M}} \models p \Uparrow b \;\; iff \; b \leq_k [\![p]\!]_{\mathcal{M}}(r)$$

$$(b' \text{ if } rp) \Uparrow b = \begin{cases} rp & \text{if } b = b' \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$$\begin{aligned}
\top \Uparrow b &= \mathsf{true} \\
(p \wedge q) \Uparrow \mathbf{f} &= p \Uparrow \mathbf{f} \vee q \Uparrow \mathbf{f} \\
(p \supset q) \Uparrow \mathbf{f} &= p \Uparrow \mathbf{t} \wedge q \Uparrow \mathbf{f} \\
(\neg p) \Uparrow b &= p \Uparrow \neg b \\
(p \wedge q) \Uparrow \mathbf{t} &= p \Uparrow \mathbf{t} \wedge q \Uparrow \mathbf{t} \\
(p \supset q) \Uparrow \mathbf{t} &= \neg(p \Uparrow \mathbf{t}) \vee q \Uparrow \mathbf{t}
\end{aligned}$$

Imperial College
London

Constraints for Queries

Query for knowledge ordering uses that truth and falsity are prime elements of the distributive lattice in the knowledge ordering

Query for assume-guarantee reasoning translates this into a propositional implication (as done in assume-guarantee reasoning for linear-time temporal logic)

$$C(p \leq_k q) = (p \Uparrow \mathbf{f} \rightarrow q \Uparrow \mathbf{f}) \wedge (p \Uparrow \mathbf{t} \rightarrow q \Uparrow \mathbf{t})$$
$$C(\alpha \Rightarrow \phi) = \alpha \rightarrow C(\phi)$$

Query for truth ordering uses characterization of truth ordering in terms of knowledge ordering

Query for conjunction is interpreted compositionally (sound for validity checking)

$$C(p \leq_t q) = (q \Uparrow \mathbf{f} \rightarrow p \Uparrow \mathbf{f}) \wedge (p \Uparrow \mathbf{t} \rightarrow q \Uparrow \mathbf{t})$$
$$C(\phi \wedge \psi) = C(\phi) \wedge C(\psi)$$

Imperial College London

Example Policy Analysis

$$p = (\mathbf{t} \text{ if } rd) \oplus (\mathbf{f} \text{ if } wr) \qquad q = p[\top \mapsto \mathbf{f}]$$

Valid query (assumes that no read action is also a write action):

$$\neg(rd \wedge wr) \;\Rightarrow\; (p \;\leq_t\; q)$$

Propositional constraints for policies:

$$\begin{aligned}
p \Uparrow \mathbf{t} &= rd \vee \mathsf{false} = rd \\
p \Uparrow \mathbf{f} &= \mathsf{false} \vee wr = wr \\
q \Uparrow \mathbf{t} &= p \Uparrow \mathbf{t} \wedge (\neg(p \Uparrow \mathbf{f}) \vee \mathbf{f} \Uparrow \mathbf{t}) = rd \wedge (\neg wr \vee \mathsf{false}) = rd \wedge \neg wr \\
q \Uparrow \mathbf{f} &= p \Uparrow \mathbf{f} \wedge (\neg(p \Uparrow \mathbf{t}) \vee \mathbf{t} \Uparrow \mathbf{t}) = wr \wedge (\neg rd \vee \mathsf{true}) = wr
\end{aligned}$$

Propositional constraint for above query is equivalent to valid formula

$$rd \vee wr \vee ((\neg rd \vee wr) \wedge (\neg rd \vee \neg wr))$$

Imperial College
London

Some Related Work

- Halpern and Weissman 2003: policies specified in first-order logic, access granted if formula logically entails formal permission predicate
- XACML standard: has no formal semantics, its semantic values and policy combination algorithms can be interpreted within PBel for suitable interpretation
- Ni et al. 2009: D-algebras as functionally complete value spaces with algebraic operators, result spaces rather ad-hoc in nature
- Bauer et al. 2005: Polymer, access-control language for untrusted Java applications, policy is query method that returns one of six values for code execution request
- Moffett and Sloman 1994: early work on policy conflict analysis
- Ribeiro et al. 2001: access-control language SPL, three-valued, can be cleanly embedded into PBel

Imperial College
London

Things we did but didn't mention here

- Support for attribute language for request predicates
- Clean encoding of policy language SPL in PBel
- Expressiveness results specialized to safe sub-languages, both for policy functions and for policy composition
- Symbolic query analysis, where policies are parameters
- Methods and interface specifications
- Request mappings, e.g. ``grant read access whenever write access is given''

Imperial College
London

Conclusions

We developed an access-control policy language based on

- abstract request predicates that encapsulated domain-specific aspects of sets of access requests
- the 4-valued Belnap bilattice whose operators were extended pointwise to our language

This gave us a very expressive core language over which common policy combination idioms can be expressed.

The core language (and so any idioms compiling into it) has a simply query analysis that supports many of the desired policy analyses and reduces them to validity checking of propositional logic.

Imperial College
London

Acknowlegments

Thanks to Jason Crampton and Daniel Dantas.