Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Composite Decentralized Access Control

Petar Tsankov, Srdjan Marinovic,
Mohammad Torabi Dashti, David Basin

Institute of Information Security
ETH Zurich

Example: SweGrid

Goal

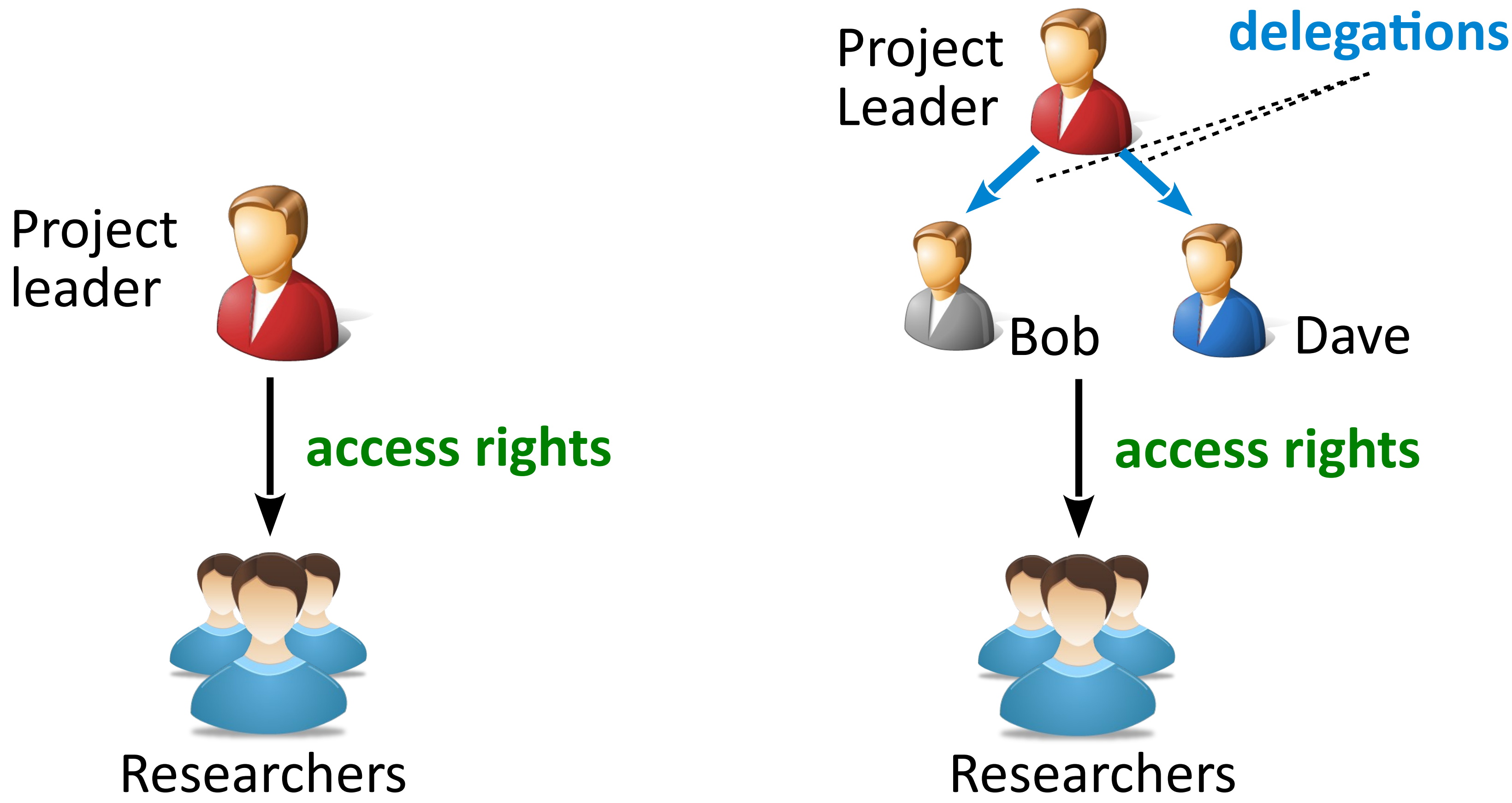
Provides computational and storage resources to researchers

Access Control Requirements

- A project leader **delegates** his authority over resources to principals
- A project leader **composes** the principals' policies (e.g., using permit-override)

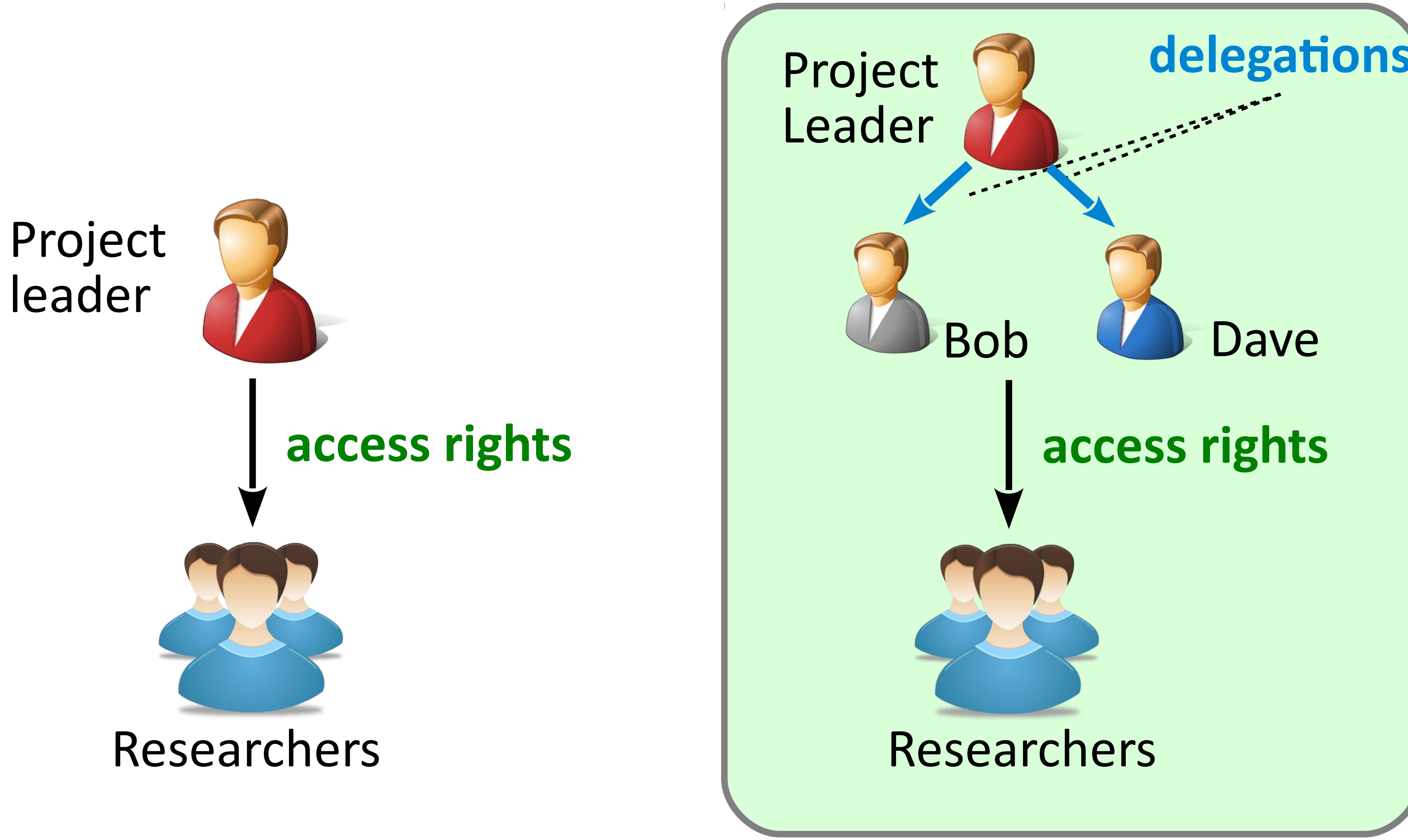
Delegation

Multiple principals can issue access rights



Delegation

Multiple principals can issue access rights

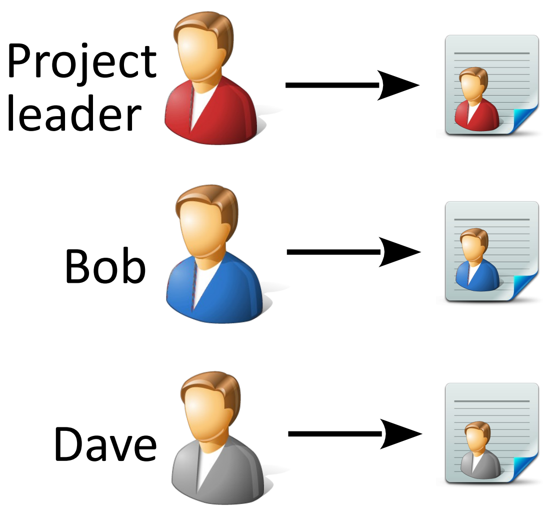


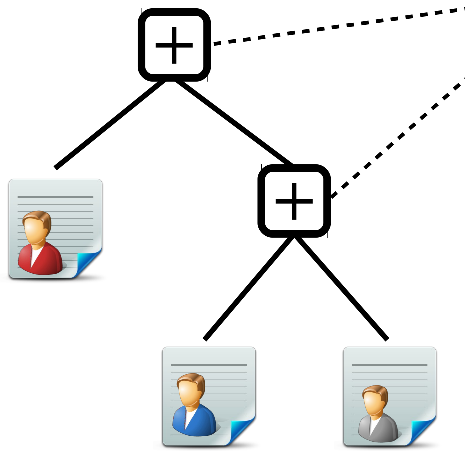
Decentralized Access Control

Composition

Policy decisions in large-scale systems

– Grant, Deny, **Not-applicable**, **Conflict**





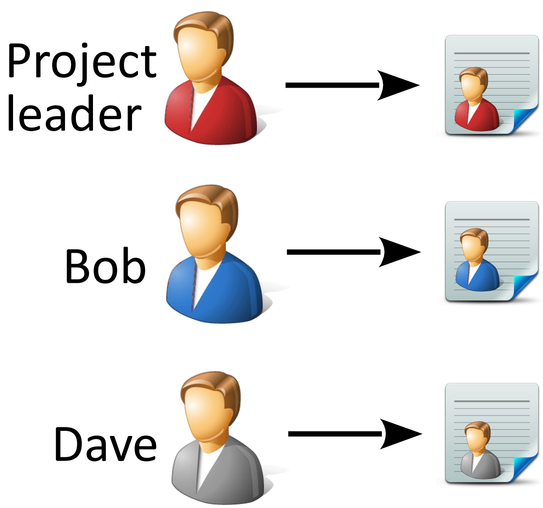
Composition operators, e.g.:

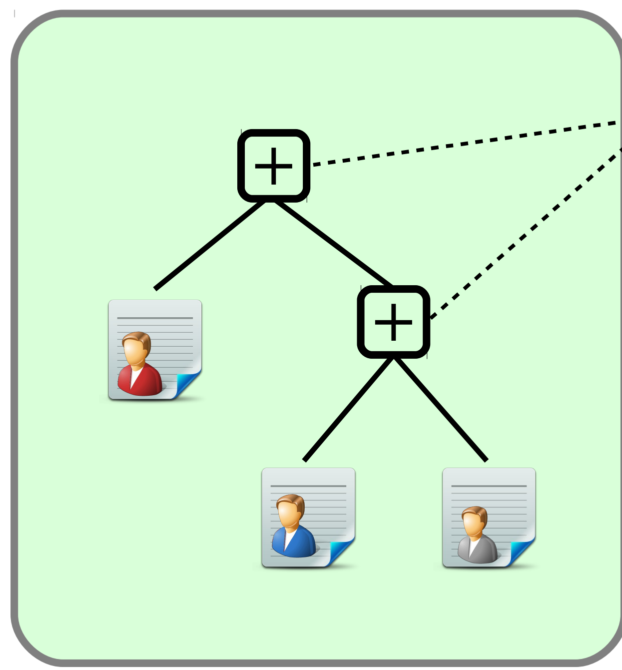
- Permit-override
- Deny-override
- Conflict-override

Composition

Policy decisions in large-scale systems

– Grant, Deny, **Not-applicable**, **Conflict**



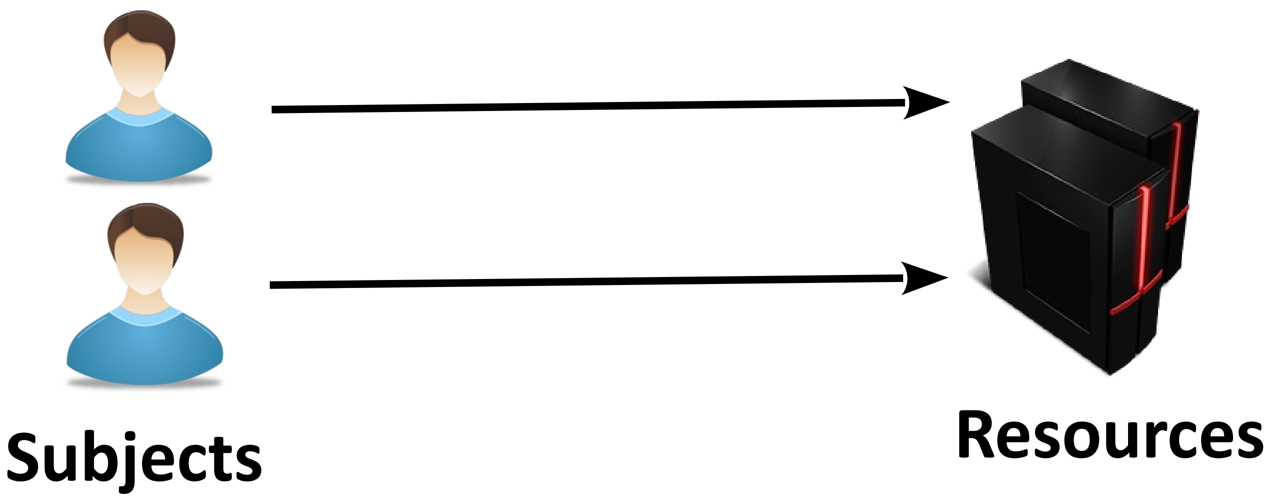


Composition operators, e.g.:

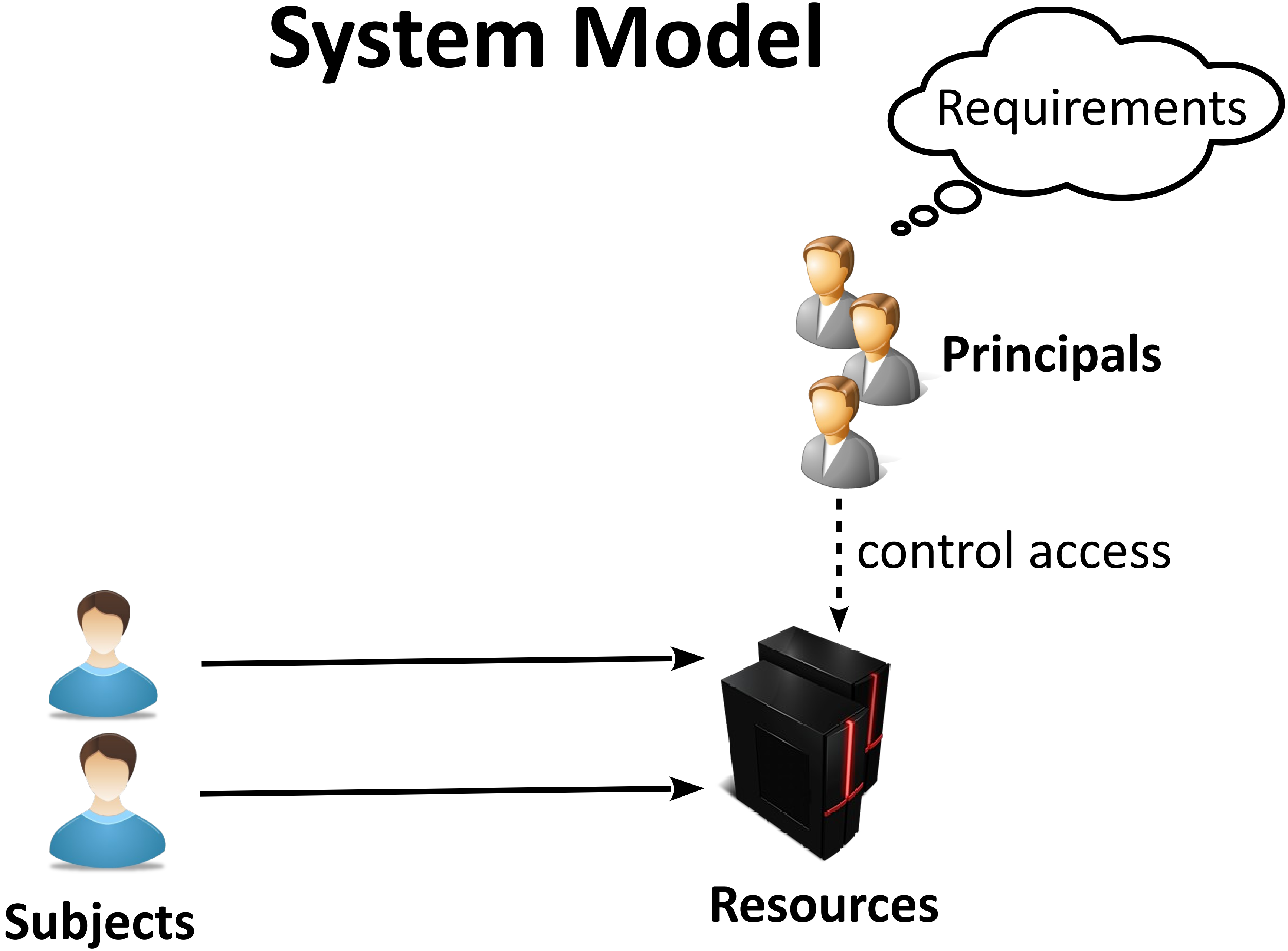
- Permit-override
- Deny-override
- Conflict-override

Composite Access Control

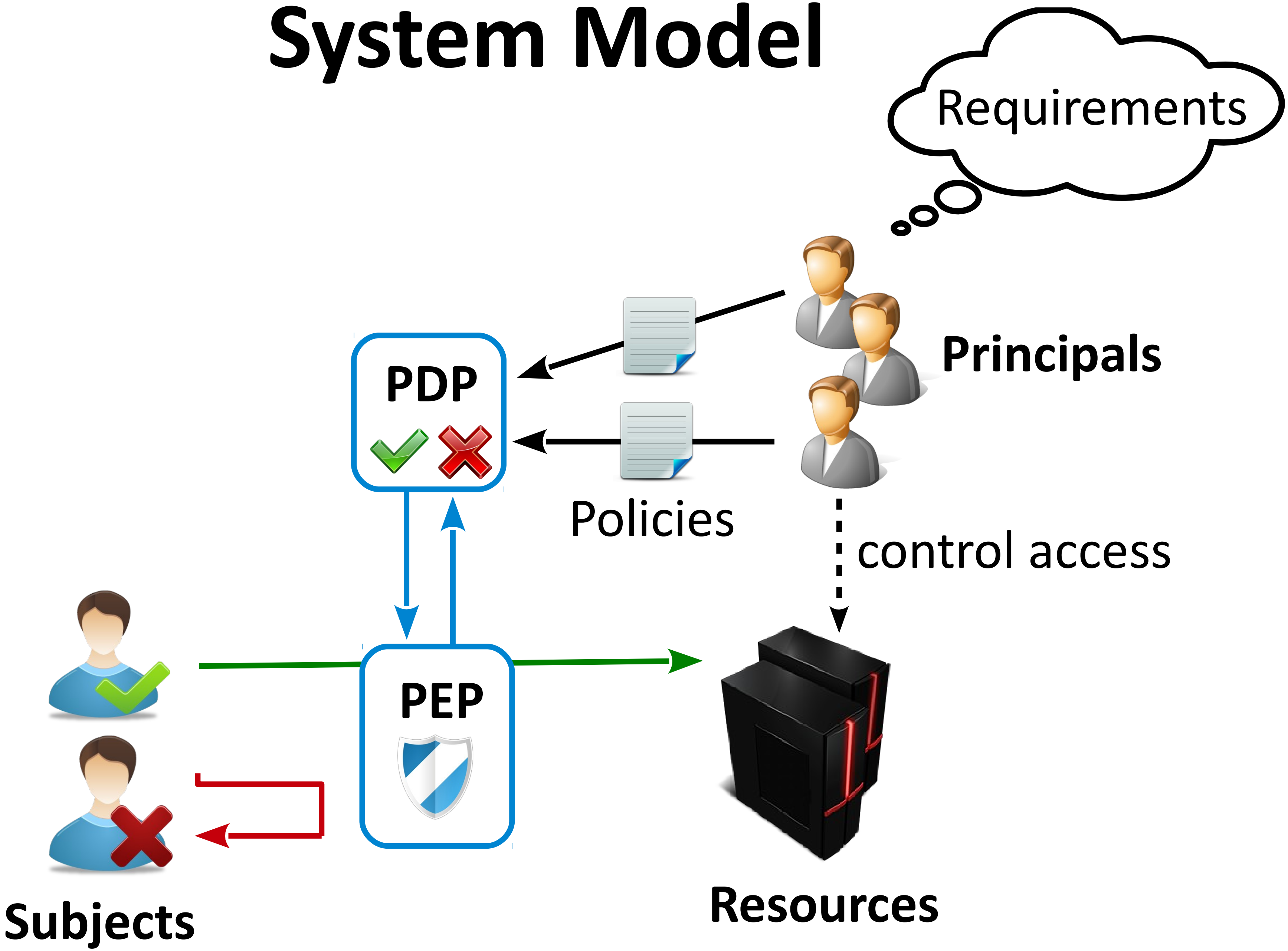
System Model



System Model



System Model



Related Work

Systems and standards	
Formal foundations	

Related Work

	Delegation
Systems and standards	SecPAL for Grid KeyNote PDP (RFC 2704)
Formal foundations	RT ('01) DKAL ('08) ...

Related Work

	Delegation	Composition
Systems and standards	SecPAL for Grid KeyNote PDP (RFC 2704)	XACML v2.0
Formal foundations	RT ('01) DKAL ('08) ...	PBel ('08) D-Algebra ('09) PTaCL ('12) ...

Related Work

	Delegation	Composition	Delegation + Composition
Systems and standards	SecPAL for Grid KeyNote PDP (RFC 2704)	XACML v2.0	SweGrid XACML v3.0 ('13) WSO2 ID Server
Formal foundations	RT ('01) DKAL ('08) …	PBel ('08) D-Algebra ('09) PTaCL ('12) …	

Related Work

	Delegation	Composition	Delegation + Composition
Systems and standards	SecPAL for Grid KeyNote PDP (RFC 2704)	XACML v2.0	SweGrid XACML v3.0 ('13) WSO2 ID Server
Formal foundations	RT ('01) DKAL ('08) ...	PBel ('08) D-Algebra ('09) PTaCL ('12) ...	**BelLog**

How to Build Access Control Systems

Specify Policy

- Formal semantics
- Support for delegation
- Support for composition

Verify Policy

- Analysis language
- Decision algorithms

Construct PDP

- Efficient evaluation algorithm

How to Build Access Control Systems

Specify Policy

- Formal semantics
- Support for delegation
- Support for composition

Verify Policy

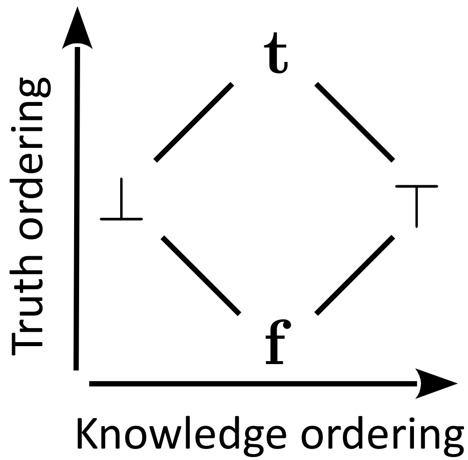
- Analysis language
- Decision algorithms

Construct PDP

- Efficient evaluation algorithm

Belnap Logic + Datalog = BelLog

Belnap Logic

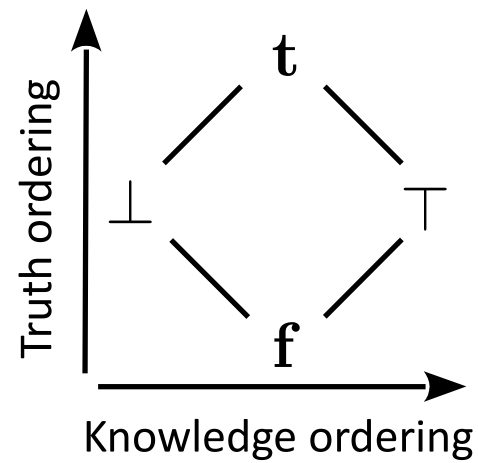


(stratified) Datalog

(Program)	$P ::= \vec{r}$
(rule)	$r ::= a \leftarrow \vec{l}$
(literal)	$l ::= a \mid \neg a$
(atom)	a

Belnap Logic + Datalog = BelLog

Belnap Logic

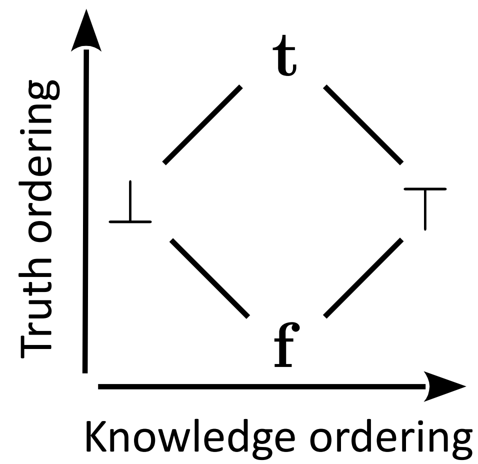


(stratified) Datalog

(Program)	$P ::= \vec{r}$
(rule)	$r ::= a \leftarrow \vec{l}$
(literal)	$l ::= a \mid \neg a$
(atom)	a

Belnap Logic + Datalog = BelLog

Belnap Logic

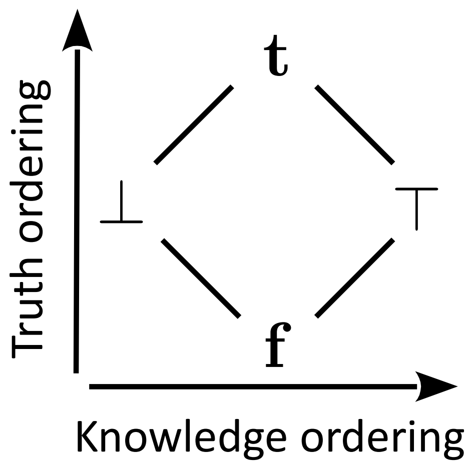


(stratified) Datalog

(Program)	$P ::= \vec{r}$
(rule)	$r ::= a \leftarrow \vec{l}$
(literal)	$l ::= a \mid \neg a$
(atom)	a

Belnap Logic + Datalog = BelLog

Belnap Logic



(stratified) Datalog

(Program) $P ::= \vec{r}$

(rule) $r ::= a \leftarrow \vec{l}$

(literal) $l ::= a \mid \neg a$

(atom) a

BelLog

(Program) $P ::= \vec{r}$

(rule) $r ::= a \leftarrow \vec{l}$

(literal) $l ::= a \mid \neg a \mid \boxed{\sim a \mid \top}$

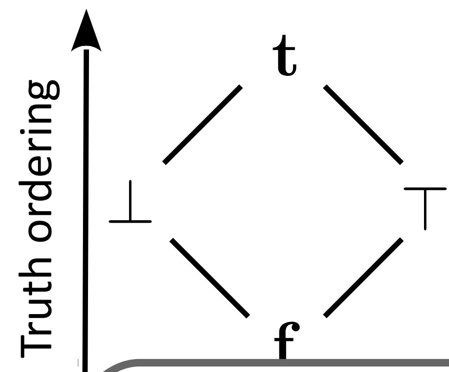
(atom) a

$\neg$ Negation on truth

$\sim$ Negation on knowledge

Belnap Logic + Datalog = BelLog

Belnap Logic



(stratified) Datalog

(Program)	$P ::= \vec{r}$
(rule)	$r ::= a \leftarrow \vec{l}$
(literal)	$l ::= a \mid \neg a$
(atom)	a

Semantics

Extend stratified Datalog to four-valued fixed-point semantics

(Program)	$P ::= \vec{r}$
(rule)	$r ::= a \leftarrow \vec{l}$
(literal)	$l ::= a \mid \neg a \mid \sim a \mid \top$
(atom)	a

$\neg$ Negation on truth

$\sim$ Negation on knowledge

BelLog Examples

BelLog Examples

Transitive delegation $pol(Req)@X \leftarrow pol(Req)@Y, delegate(Y)@X$

BelLog Examples

Transitive delegation	$pol(Req)@X \leftarrow pol(Req)@Y, delegate(Y)@X$
Policy targets	$polA(Req) \leftarrow target(Req) \blacktriangleright polB(Req)$

BelLog Examples

Transitive delegation	$pol(Req)@X \leftarrow pol(Req)@Y, delegate(Y)@X$
Policy targets	$polA(Req) \leftarrow target(Req) \blacktriangleright polB(Req)$
Agreement	$polA(Req) \leftarrow polB(Req) \oplus polC(Req)$

BelLog Examples

Transitive delegation	$pol(Req)@X \leftarrow pol(Req)@Y, delegate(Y)@X$
Policy targets	$polA(Req) \leftarrow target(Req) \blacktriangleright polB(Req)$
Agreement	$polA(Req) \leftarrow polB(Req) \oplus polC(Req)$
Conflict-override	$polA(Req) \leftarrow polB(Req)[\top \mapsto polC(Req)]$

BelLog Examples

Transitive delegatic $eq)@Y, delegate(Y)@X$

Policy targets $Req) \blacktriangleright polB(Req)$

Agreement $eq) \oplus polC(Req)$

Conflict-override $eq)[\top \mapsto polC(Req)]$

Other idioms?

How to Build Access Control Systems

Specify Policy

- ➔ Formal semantics
- ➔ Support for delegation
- ➔ Support for composition

Verify Policy

- ➔ Analysis language
- ➔ Decision algorithms

Construct PDP

- ➔ Efficient evaluation algorithm

Policy Analysis

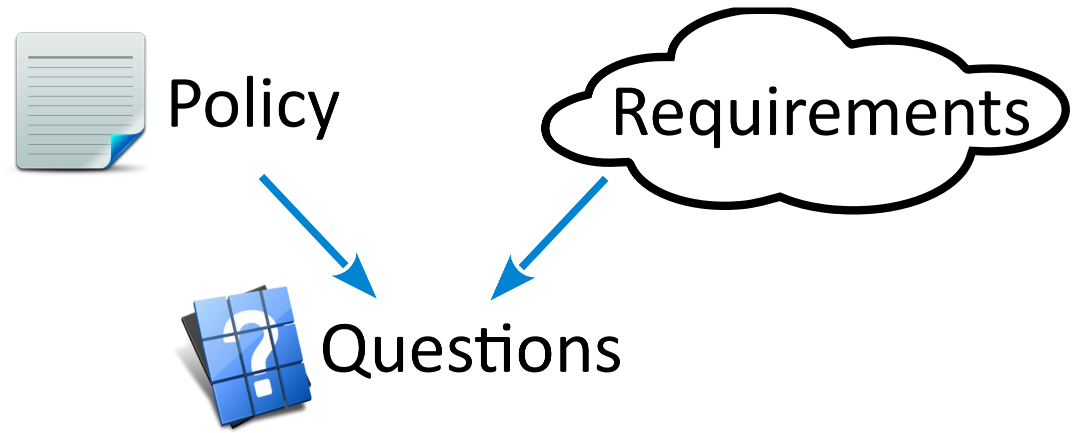
Does the policy meet its requirements?

Policy

Requirements

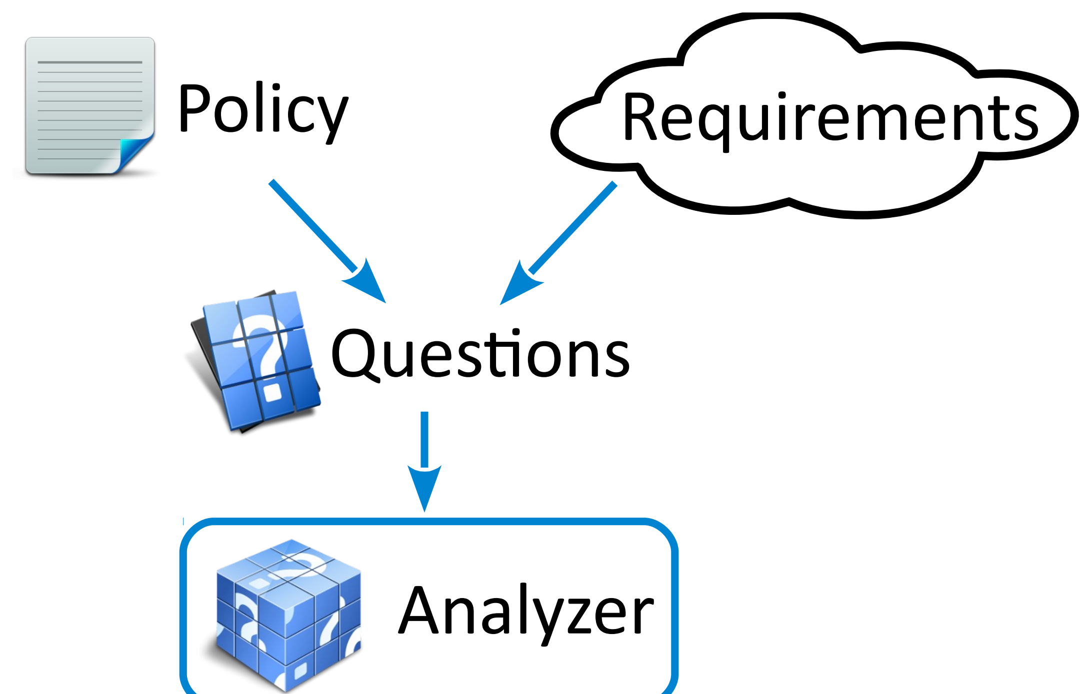
Policy Analysis

Does the policy meet its requirements?



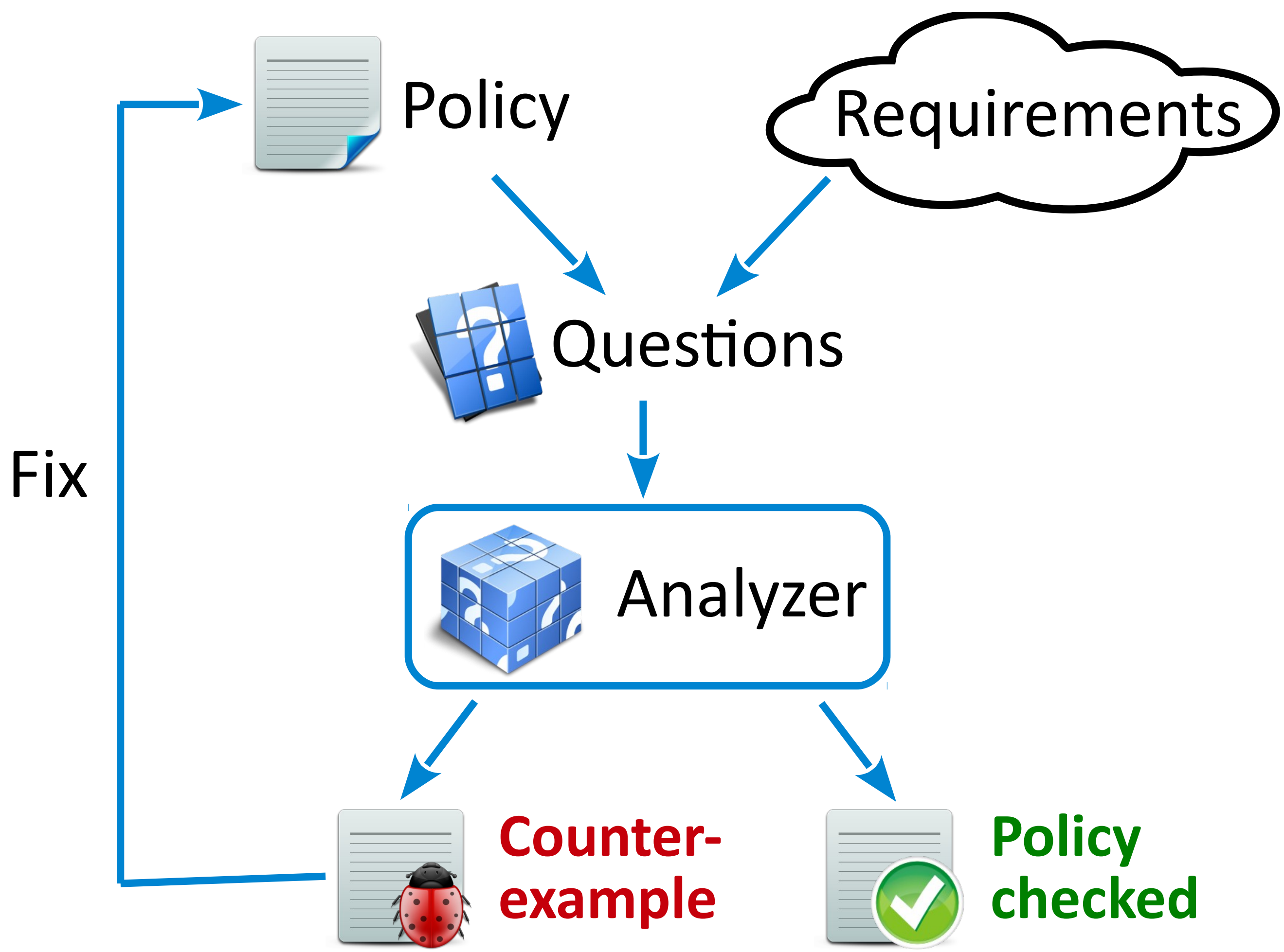
Policy Analysis

Does the policy meet its requirements?



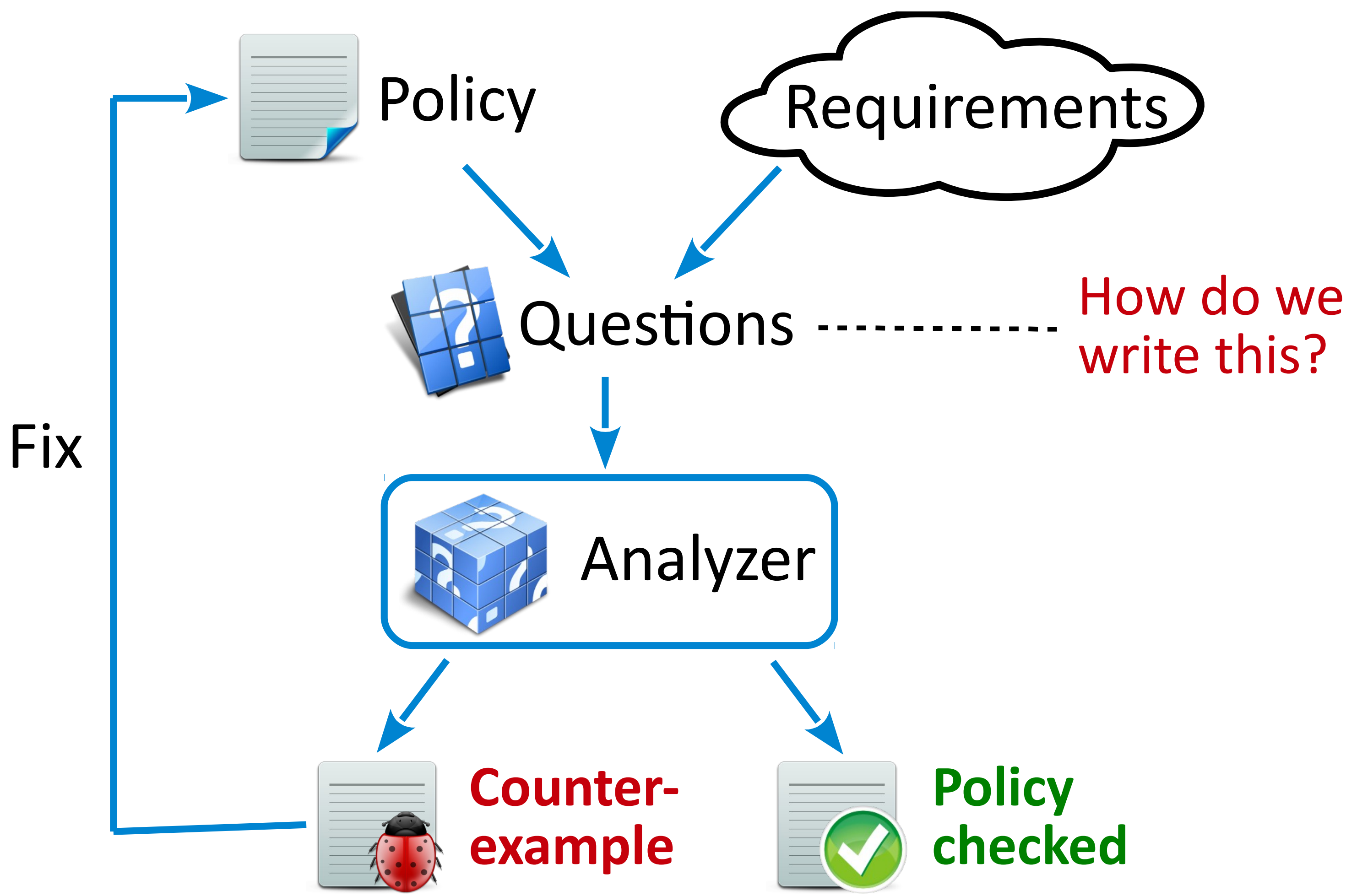
Policy Analysis

Does the policy meet its requirements?



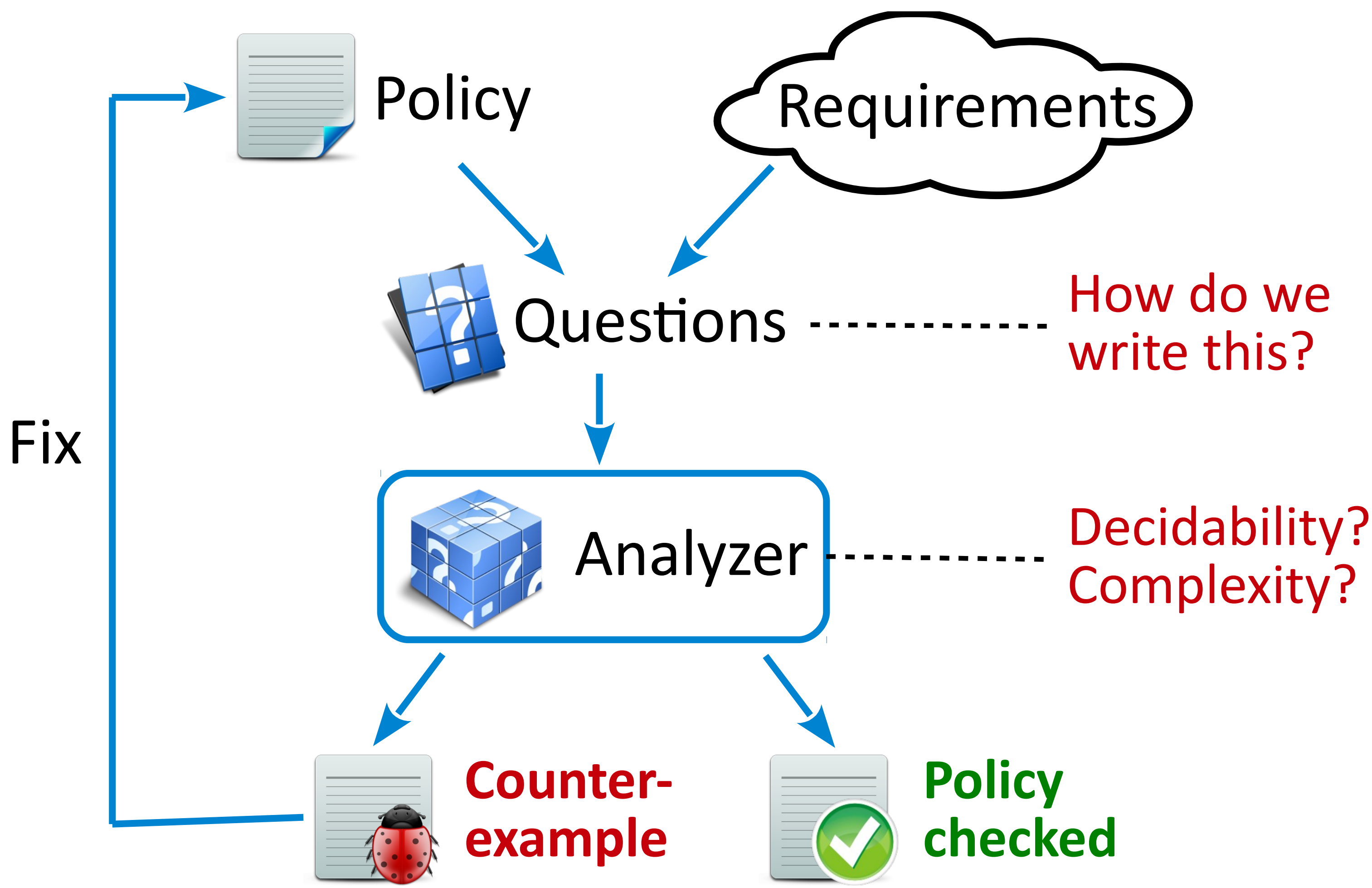
Policy Analysis

Does the policy meet its requirements?



Policy Analysis

Does the policy meet its requirements?



Analysis Questions

Syntax

(question) $(c \Rightarrow P_1 \preceq P_2)$

(condition) $c ::= a(X) = \mathbf{t} \mid \exists X.\ c \mid \neg c \mid \cdots$

– Is policy **P2** more permissive than **P1** for all inputs that satisfy the condition **c**?

Analysis Questions

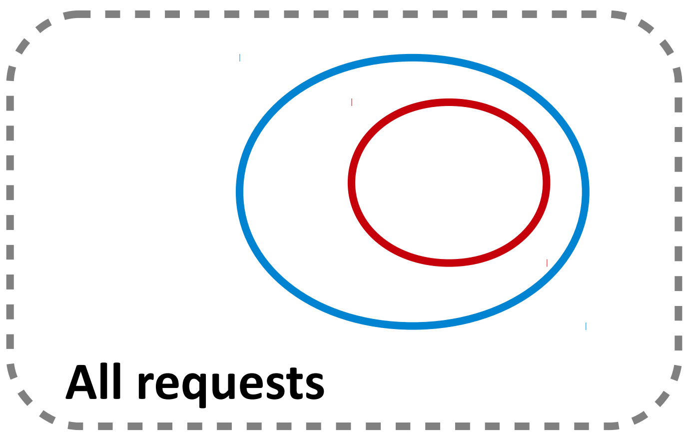
Syntax

(question) $(c \Rightarrow P_1 \preceq P_2)$

(condition) $c ::= a(X) = \mathbf{t} \mid \exists X.\ c \mid \neg c \mid \cdots$

– Is policy **P2** more permissive than **P1** for all inputs that satisfy the condition **c**?

For a given input:



Requests granted by **P1**

Requests granted by **P2**

Analysis Questions

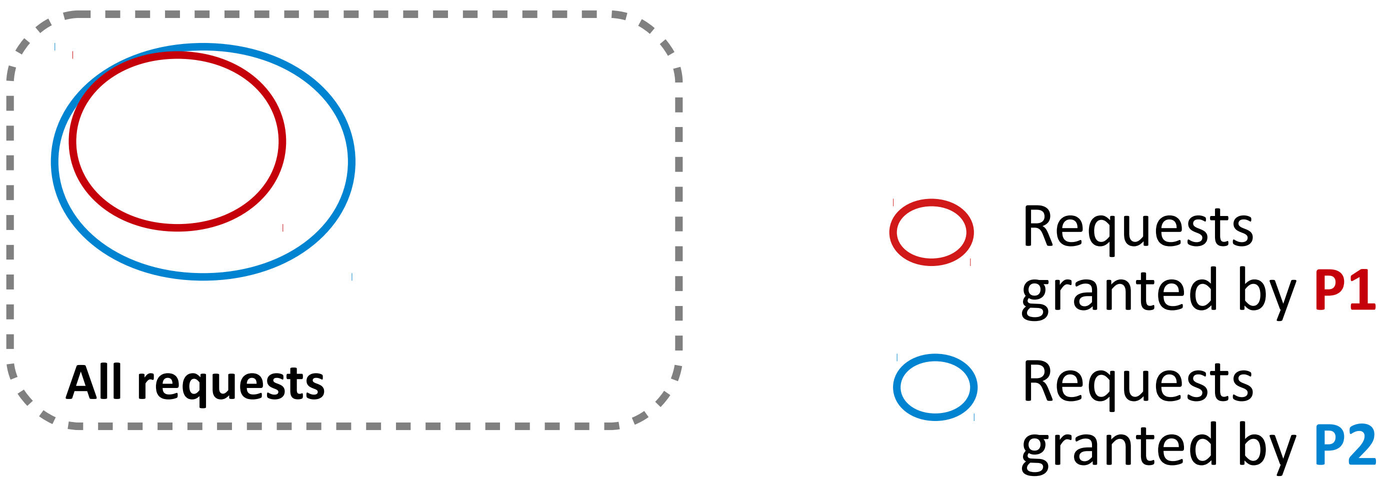
Syntax

(question) $(c \Rightarrow P_1 \preceq P_2)$

(condition) $c ::= a(X) = \mathbf{t} \mid \exists X.\ c \mid \neg c \mid \cdots$

– Is policy **P2** more permissive than **P1** for all inputs that satisfy the condition **c**?

For a given input:



Analysis Questions

Syntax

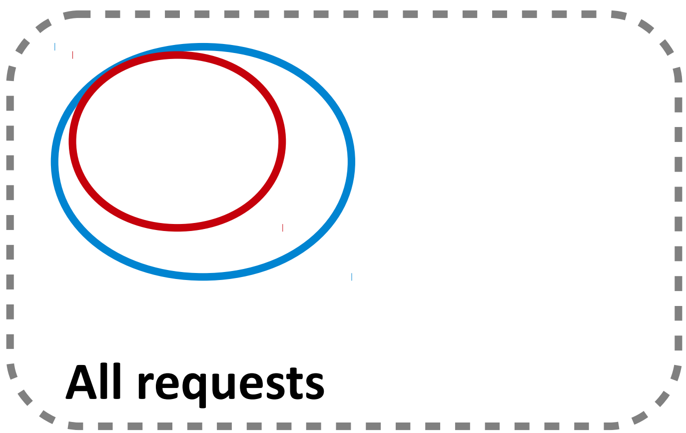
(question) $(c \Rightarrow P_1 \preceq P_2)$

(condition) $c ::= a(X) = \mathbf{t} \mid \exists X.\ c \mid \neg c \mid \cdots$

– Is policy **P2** more permissive than **P1** for all inputs that satisfy the condition **c**?

For a given input:

Check for all inputs that satisfy the condition



Example: Analysis Question

Policy

P

Requirement

If the requester is a project leader, then grant access.

Example: Analysis Question

Policy

P

Requirement

If the requester is a project leader, then grant access.

Analysis Question

$$leader(Sub) \Rightarrow P \preceq \{pol(Sub) \leftarrow \mathbf{t}\}$$

Analysis

Analysis

Theorem 1
Policy containment is **undecidable**

Analysis

Theorem 1
Policy containment is **undecidable**

Theorem 2
Policy containment for **unary-input policies*** is in **CO-NEXP-COMPLETE**

*Unary-input policies

– Example: $pol(Sub, Obj) \leftarrow leader(Sub), pub(Obj)$

Analysis

Theorem 1
Policy containment is **undecidable**

Theorem 2
Policy containment for **unary-input policies*** is in **CO-NEXP-COMPLETE**

Theorem 3
Policy containment for **a finite universe** is in **CO-NP-COMPLETE**

*Unary-input policies

- Example: $pol(Sub, Obj) \leftarrow leader(Sub), pub(Obj)$

How to Build Access Control Systems

Specify Policy

- Formal semantics
- Support for delegation
- Support for composition

Verify Policy

- Analysis language
- Decision algorithms

Construct PDP

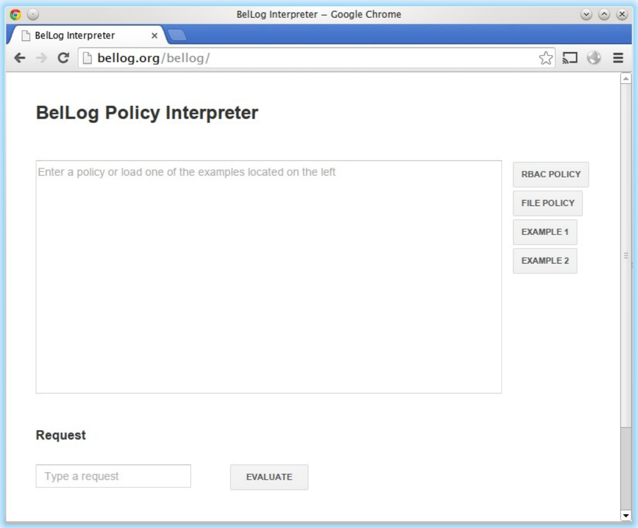
- Efficient evaluation algorithm

Constructing PDPs

Theorem 4
Policy entailment is in **PTIME**

Policy Interpreter

http://bellog.org



GitHub

https://github.com/ptsankov/bellog/

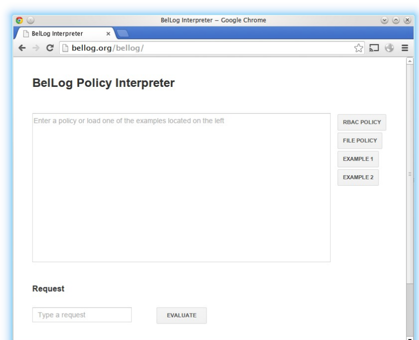
Limitations

- Analysis of administrative changes
- Analysis complexity and tool support
- Usability

BelLog Contributions

A foundation for composite decentralized access control

Policy analysis framework



BelLog PDP
(www.bellog.org)