PPLV SEMINAR · UNIVERSITY COLLEGE LONDON

Robust Properties of Unfinished Models

Alexander V. Gheorghiu
University of Southampton · University College London

PART 1

Modelling Systems

Structure and behaviour.

STRUCTURE AND BEHAVIOUR

Two views of a system

SYSTEM AS STRUCTURE

Its **states**, and their relations.

a program at a version · a heap · a Kripke model

SYSTEM AS BEHAVIOUR

Its **dynamics**: the moves it permits.

a transition system · a protocol · a policy · a logic program

STRUCTURE AND BEHAVIOUR

Two verification questions

STRUCTURE

$$\mathcal{M} \models \varphi$$

does the model make φ true?

BEHAVIOUR

$$\Vdash_{\mathcal{P}} \varphi$$

do the rules commit to φ?

STRUCTURE AND BEHAVIOUR

How the two views are related

The two views do not compete: a structure is a special, finished case of behaviour.

First-order structures correspond to **complete**, consistent sets of atomic rules — sets in which every atomic question is settled.

Lindenbaum and Tarski (1935); Makinson (2014), roughly.

*A structure is what behaviour becomes when **every** question has been decided.*

STRUCTURE AND BEHAVIOUR

Unfinished specifications

*An unfinished specification determines **no** model.*

*It does determine **commitments**: what must hold however it is finished.*

STRUCTURE AND BEHAVIOUR

Requirements

PRINCIPLED

Meaning from the rules themselves.

COMPOSITIONAL

Verdicts survive extension.

IMPLEMENTABLE

One can build a tool.

Proof-theoretic semantics gives the first; this talk shows the other two.

PART 2

Case Study

Access-control policies.

CASE STUDY

Writing a policy

- Bob is on the security team.
- He writes the policy deciding who may see what — in XACML, the OASIS standard.
- Like most engineers now, he writes a chunk of it with AI assistance.

```
<Policy RuleCombiningAlgId=
  "permit-overrides">
  <Rule Effect="Permit">
    …
  </Rule>
</Policy>
```

(partly generated)

CASE STUDY

Policies as logic programs



Ramli, Nielson & Nielson (2014): many-sorted FOL. Ahn et al. (2010): answer-set programs.

▶ From here on, a policy *is* its program.

CASE STUDY

Properties of real policies

PARTIAL

Written rule by rule; most permissions never explicitly decided.

COMPOSED

Engineering, legal and security each author fragments.

EXTENDED

New roles, resources, regulations. Never finished.

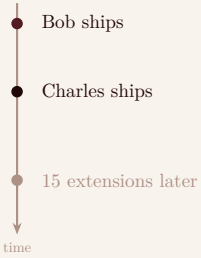
Asking about the fully decided policy is asking about a state it will ***never*** *be in.*

CASE STUDY

The verification requirement

Alice has two questions:

- Does Bob's policy do what it should?
- Does it *still*, once Charles ships next week?



She wants it verified — and she wants the verification to ***survive.***

CASE STUDY

Existing tools

DOES WELL

Questions over a **fully specified** policy, at industrial scale.

SMT (Z3, Zelkova), BDDs (Margrave), Alloy, ASP.

DOES NOT ADDRESS

Partiality — they treat anything unstated as false.

Composition — results for parts do not combine.

Extension — every change means verifying again.

Exactly the three things Alice needs.

CASE STUDY

Robust properties

THE WRONG QUESTION

"Does φ hold of Bob's policy?"

THE RIGHT QUESTION

"Is φ a **robust** property of Bob's policy?"

Robust: guaranteed however the policy is completed or extended.

PART 3

Support

The semantics of robust properties.

SUPPORT

Notation

GENERAL

$\mathcal{P}$, $\mathcal{Q}$

Range over **any** policy. Definitions and theorems are stated with these.

Sandqvist calls them *bases*.

PARTICULAR

Conf

One **fixed** policy: the running example.

Results about it are computed.

Each variant of it gets its own name.

$\Vdash_{\mathcal{P}} \varphi$ *is a definition;* $\Vdash_{\mathsf{Conf}} \varphi$ *is a fact to check.*

SUPPORT

Example policy Conf (1/2)

FACTS

pcMember(ada). pcMember(ben). pcMember(cleo).
author(ada, p007).

CONFLICT OF INTEREST

pcMember$(m) \supset$ author$(m, p) \supset$ conflicted(m, p)

A CONFLICTED REVIEWER IS A VIOLATION

conflicted$(m, p) \supset$ canReview$(m, p) \supset$ violation(m, p)

SUPPORT

Example policy Conf (2/2)

SEPARATION OF DUTY

$\mathsf{chair}(m) \supset \mathsf{author}(m, p) \supset \mathsf{violation}(m, p)$

READINESS

$\mathsf{chair}(m) \supset \mathsf{pcMember}(m) \supset \mathsf{chairReady}$

INTEGRITY — $\mathbf{0}$ MEANS "CORRUPT", EXPLAINED SHORTLY

$\mathsf{violation}(m, p) \supset \mathbf{0}$

SUPPORT

Policies so far

Conf the running example: facts; reviewing, chair and integrity rules

Every result in this talk names one of these policies; the newest is highlighted.

SUPPORT

What Conf derives now

$$\vdash_{\mathsf{Conf}} \mathsf{conflicted}(\mathsf{ada}, \mathsf{p007}) \quad \checkmark$$

$$\vdash_{\mathsf{Conf}} \mathsf{violation}(\mathsf{ada}, \mathsf{p007}) \quad \times$$

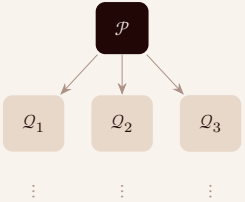
$$\vdash_{\mathsf{Conf}} \mathsf{chair}(\mathsf{ben}) \quad \times$$

No reviewer assigned, no chair appointed. Both decisions are ***pending****.*

SUPPORT

Extensions

- $\mathcal{Q} \supseteq \mathcal{P}$: $\mathcal{Q}$ extends $\mathcal{P}$.
- Rules are added, never retracted.
- The extensions are every way $\mathcal{P}$ may yet be completed.



SUPPORT

Analogy with contextual equivalence

CONTEXTUAL EQUIVALENCE

$M \cong N$ iff for every context C:
$C[M] \Downarrow$ if and only if $C[N] \Downarrow$

$\Downarrow$: terminates

SUPPORT

$\Vdash_{\mathcal{P}} \varphi$ quantifies over every **extension** $\mathcal{Q} \supseteq \mathcal{P}$.

Meaning is behaviour in every context — here, every future.

SUPPORT

Atoms as observations

- The only thing we observe: whether an atom is derivable, $\vdash_Q A$.
- For example $\mathsf{author}(\mathsf{ada}, \mathsf{p007})$ or $\mathsf{canReview}(\mathsf{ben}, \mathsf{p007})$.
- A compound property is never observed directly.

A property is characterised by ***which atoms it makes derivable.***

SUPPORT

Atoms

$$\Vdash_{\mathcal{P}} A \text{ if and only if } \vdash_{\mathcal{P}} A$$

The ground floor: what the policy derives now.

SUPPORT

Conditional support

$$\Gamma \Vdash_{\mathcal{P}} \varphi \text{ if and only if, for every } \mathcal{Q} \supseteq \mathcal{P}:$$
$$\text{if } \Vdash_{\mathcal{Q}} \psi \text{ for every } \psi \in \Gamma, \text{ then } \Vdash_{\mathcal{Q}} \varphi.$$

Every extension that establishes Γ *also establishes* φ*.*

SUPPORT

Implication

$$\Vdash_{\mathcal{P}} \varphi \Rightarrow \psi \quad \text{if and only if} \quad \varphi \Vdash_{\mathcal{P}} \psi$$

- $\Rightarrow$ is a connective *inside* formulas. Statements *about* formulas use "if … then".
- Not material implication: it does not hold merely because φ is not yet derivable.
- The hypothesis may be a *future fact*, such as $\mathsf{author}(\mathsf{ben}, \mathsf{p007})$.

SUPPORT

Disjunction: what we need

- A policy may say "one of Ada, Ben or Cleo will chair" without saying who. (Part 5 builds it.)
- We want to state that commitment — and *use* it.
- Using it means reasoning by cases: if each choice of chair leads to A, then A.

*A disjunction must hold **before** the choice is made.*

SUPPORT

Disjunction: the obvious clause

$$\Vdash_{\mathcal{P}} \varphi \vee \psi \text{ if and only if } \Vdash_{\mathcal{P}} \varphi \text{ or } \Vdash_{\mathcal{P}} \psi$$

- The chair policy supports none of $\mathsf{chair}(\mathsf{ada})$, $\mathsf{chair}(\mathsf{ben})$, $\mathsf{chair}(\mathsf{cleo})$.
- So it would not support their disjunction: "one of them will chair" could not be stated.

*It demands that the decision be **already** made.*

SUPPORT

Disjunction: the clause

$\Vdash_{\mathcal{P}} \varphi \vee \psi$ if and only if, for every $\mathcal{Q} \supseteq \mathcal{P}$ and every atom A:

if $\varphi \Vdash_{\mathcal{Q}} A$ and $\psi \Vdash_{\mathcal{Q}} A$, then $\vdash_{\mathcal{Q}} A$.

The policy honours every argument by cases — now and in every extension.

SUPPORT

Disjunction: why each part is there

- **By cases.** "If φ yields A and ψ yields A, then A" is how a disjunction is used. Neither disjunct need hold.
- **Every extension $\mathcal{Q}$.** A case argument may rely on rules added later. Checking only $\mathcal{P}$ would let a new rule create an argument the policy does not honour.
- **Atoms A.** Atoms are what we observe. Agreeing on every observation is all we can ask of a commitment.

SUPPORT

Conjunction: what we need

- Requirements come in groups: no conflicted reviewer *and* no author as chair.
- We want a single property that commits the policy to both.

*A conjunction holds when the policy is committed to **each** part.*

SUPPORT

Conjunction: the obvious clause

$$\Vdash_{\mathcal{P}} \varphi \wedge \psi \text{ if and only if } \Vdash_{\mathcal{P}} \varphi \text{ and } \Vdash_{\mathcal{P}} \psi$$

- This one **does** give what we want.
- Nothing is left undecided between the two parts, so nothing is lost.
- It is robust: if both hold of $\mathcal{P}$, both hold of every extension.

SUPPORT

Conjunction: the clause

$\Vdash_{\mathcal{P}} \varphi \wedge \psi$ if and only if, for every $\mathcal{Q} \supseteq \mathcal{P}$ and every atom A:

$$\text{if } \varphi, \psi \Vdash_{\mathcal{Q}} A, \text{ then } \vdash_{\mathcal{Q}} A.$$

- By robustness (Part 4), this agrees with the obvious clause.
- We use it for **uniformity**: like disjunction, it is defined by what it lets us derive.
- The proof-search encoding (Part 6) relies on that shared shape.

SUPPORT

Corruption: what we need

- Prohibitions: “an author must never review her own paper”.
- We define $\neg\varphi$ as $\varphi \Rightarrow \mathbf{0}$: establishing φ leads to a bad state.
- So we need a bad state $\mathbf{0}$ — one that a policy can actually reach.

Prohibition must mean “this would break the policy”.

SUPPORT

Corruption: the obvious clause

0 is never supported, by any policy

- Any atom A can be added as a fact: $\mathcal{P} \cup \{A\}$ supports A.
- So some extension establishes A without reaching **0**.
- Hence $\neg A$ would fail for *every* atom, in every policy.

A bad state nothing can reach makes every prohibition ***false****.*

SUPPORT

Corruption: the clause

$$\Vdash_{\mathcal{P}} \mathbf{0} \text{ if and only if } \Vdash_{\mathcal{P}} A \text{ for every atom } A$$

- A policy that derives every atom permits everything. It no longer separates permitted from forbidden.
- A policy *can* reach this state — through rules it contains.
- An integrity rule such as $\mathsf{violation}(m, p) \supset \mathbf{0}$ says: a violation collapses the policy.

SUPPORT

Corruption: why every atom

- **Reachable.** Unlike "never supported", a policy can get here — so prohibitions can hold.
- **No designated atom.** A special error atom would also work, but every policy would have to agree on its name.

"Everything is derivable" is the bad state every policy recognises.

SUPPORT

Negation

$$\neg\varphi \;:=\; \varphi \Rightarrow \mathbf{0}$$

- "A is not derivable" is **not** robust: one added rule can destroy it.
- $\neg\varphi$ says: any extension that establishes φ is corrupt.

SUPPORT

Prohibitions

$$\Vdash_{\mathcal{P}} \neg A \quad \text{if and only if} \quad \mathcal{P} \cup \{A\} \text{ is corrupt}$$

- So a robust prohibition holds only if the rules *say* so.
- e.g. an integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$.

SUPPORT

Variant: $\overline{\mathsf{Conf}}$

$$\overline{\mathsf{Conf}} \;:=\; \mathsf{Conf} \text{ without the integrity clause}$$

- Same facts, same rules — except $\mathsf{violation}(m, p) \supset \mathbf{0}$ is gone.
- $\overline{\mathsf{Conf}} \subseteq \mathsf{Conf}$: a *different* policy, not a future of Conf.

SUPPORT

Policies so far

Conf	the running example: facts; reviewing, chair and integrity rules
Conf^{-}	Conf without the integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$

Every result in this talk names one of these policies; the newest is highlighted.

SUPPORT

Negation in Conf and Conf^{-}

$\mathsf{Conf} \Vdash \neg\,\mathsf{canReview}(\mathsf{ada}, \mathsf{p007})$ ✓ integrity clause present

$\mathsf{Conf}^{-} \Vdash \neg\,\mathsf{canReview}(\mathsf{ada}, \mathsf{p007})$ × integrity clause removed

Absence of evidence is not a property.

SUPPORT

Summary of clauses

$$
\begin{aligned}
&\Vdash_{\mathcal{P}} A && \text{iff } \vdash_{\mathcal{P}} A \\
&\Vdash_{\mathcal{P}} \varphi \Rightarrow \psi && \text{iff } \varphi \Vdash_{\mathcal{P}} \psi \\
&\Vdash_{\mathcal{P}} \varphi \wedge \psi && \text{iff for all } \mathcal{Q} \supseteq \mathcal{P} \text{ and atoms } A\colon \text{ if } \varphi, \psi \Vdash_{\mathcal{Q}} A \text{ then } \vdash_{\mathcal{Q}} A \\
&\Vdash_{\mathcal{P}} \varphi \vee \psi && \text{iff for all } \mathcal{Q} \supseteq \mathcal{P} \text{ and atoms } A\colon \text{ if } \varphi \Vdash_{\mathcal{Q}} A \text{ and } \psi \Vdash_{\mathcal{Q}} A \text{ then } \vdash_{\mathcal{Q}} A \\
&\Vdash_{\mathcal{P}} \mathbf{0} && \text{iff } \Vdash_{\mathcal{P}} A \text{ for every atom } A
\end{aligned}
$$

At the empty policy, $\Vdash_{\emptyset} \varphi$ holds exactly when φ is intuitionistically valid (Sandqvist 2015). The paper writes $\otimes, \oplus$ for $\wedge, \vee$; they are intuitionistic here, not linear.

PART 4

Robustness

Monotonicity under extension.

ROBUSTNESS

Robustness theorem

$$\text{If } \Vdash_{\mathcal{P}} \varphi \text{ and } \mathcal{Q} \supseteq \mathcal{P}, \text{ then } \Vdash_{\mathcal{Q}} \varphi.$$

Proof: induction on φ. Atoms by monotonicity of $\vdash$; every other clause quantifies over extensions, and extensions of $\mathcal{Q}$ are extensions of $\mathcal{P}$.

ROBUSTNESS

Consequences

SURVIVAL

Verify once; it survives Charles.

COMPOSITION

Verify fragments separately; combine.

PARTIALITY

Verify before the policy is finished.

*Verification effort tracks the **change**, not the size of the policy.*

ROBUSTNESS

Composition

If $\Vdash_{\mathcal{P}_1} \varphi$ and $\Vdash_{\mathcal{P}_2} \psi$, then $\Vdash_{\mathcal{P}_1 \cup \mathcal{P}_2} \varphi \wedge \psi$.

Two teams, two checks, one guarantee.

ROBUSTNESS

Limitation

*Robustness is relative to **additive** change.*

Revocation — deleting a rule — is outside this model. That is a modelling choice, and it should be stated up front.

PART 5

Pending Decisions

Reasoning before a decision is made.

PENDING DECISIONS

Scenario

One of Ada, Ben or Cleo will chair. Nobody knows who.

*The constraints must hold **now**.*

PENDING DECISIONS

Chair rules in Conf

SEPARATION OF DUTY

$$\mathsf{chair}(m) \supset \mathsf{author}(m, p) \supset \mathsf{violation}(m, p)$$

READINESS

$$\mathsf{chair}(m) \supset \mathsf{pcMember}(m) \supset \mathsf{chairReady}$$

Conf says nothing about *who* chairs. Shorthand: $c_a = \mathsf{chair}(\mathsf{ada})$, $c_b = \mathsf{chair}(\mathsf{ben})$, $c_c = \mathsf{chair}(\mathsf{cleo})$, $c_d = \mathsf{chair}(\mathsf{dev})$.

PENDING DECISIONS

Option (a): decision as a hypothesis

$$\Vdash_{\mathsf{Conf}} \ (c_a \vee c_b \vee c_c) \Rightarrow \mathsf{chairReady}$$

This holds exactly when each candidate alone yields chairReady*:*
$c_x \Vdash_{\mathsf{Conf}} \mathsf{chairReady}$.

The policy is plain Conf. The decision appears only in the formula.

PENDING DECISIONS

Option (a): adding a candidate

- The three old checks still hold, by robustness.
- One new check: $c_d \Vdash_{\mathsf{Conf}} \mathsf{chairReady}$.
- It fails, because Dev is not on the PC. The failing check names the candidate.

Adding a candidate adds one check.

PENDING DECISIONS

Option (b): decision as a clause

$$\mathsf{Pend}(\text{Ada, Ben, Cleo}) \;:=\; \forall X.\,(c_a \supset X) \supset (c_b \supset X) \supset (c_c \supset X) \supset X$$

IN WORDS

For any atom X: if appointing Ada would yield X, and so would Ben, and so would Cleo, then X holds.

It says "one of them will chair" — without saying ***which****.*

PENDING DECISIONS

Variant: $\mathsf{Conf}^{\mathsf{pend}}$

$$\mathsf{Conf}^{\mathsf{pend}} := \mathsf{Conf} \cup \{\mathsf{Pend}(\text{Ada}, \text{Ben}, \text{Cleo})\}$$

- $\mathsf{Conf}^{\mathsf{pend}} \supseteq \mathsf{Conf}$: an *extension*. The decision is now part of the policy.
- Contrast option (a), where the policy stayed Conf and the decision was a hypothesis.

PENDING DECISIONS

Policies so far

Conf	the running example: facts; reviewing, chair and integrity rules
Conf^{-}	Conf without the integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$
$\mathsf{Conf}^{\mathsf{pend}}$	Conf plus the clause $\mathsf{Pend}(\text{Ada, Ben, Cleo})$

$$\mathsf{Pend}(x_1, \ldots, x_n) \;:=\; \forall X.\, (\mathsf{chair}(x_1) \supset X) \supset \cdots \supset (\mathsf{chair}(x_n) \supset X) \supset X$$

"one of $x_1, \ldots, x_n$ will chair, without saying which"

Every result in this talk names one of these policies; the newest is highlighted.

PENDING DECISIONS

Results for $\mathsf{Conf}^{\mathsf{pend}}$ (1/2)

$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}}$	$\mathsf{chairReady}$	✓	whoever is appointed
$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}}$	$\mathsf{chair}(\mathsf{ben})$	×	the policy does not decide who

A guarantee, while the decision is still open.

PENDING DECISIONS

Results for $\mathsf{Conf}^{\mathsf{pend}}$ (2/2)

$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}} \neg\,\mathsf{chair}(\mathsf{ada})$	✓	she authored p007
$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}} \mathsf{chair}(\mathsf{ben}) \vee \mathsf{chair}(\mathsf{cleo})$	✓	so Ada is ruled out
$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}} \mathbf{0}$	×	and the policy is consistent

Derived from the rules alone. No completed policy was ever built.

PENDING DECISIONS

Variant: $\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}}$

$$\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}} := \mathsf{Conf} \cup \{\mathsf{Pend}(\mathrm{Ada}, \mathrm{Ben}, \mathrm{Cleo}, \mathrm{Dev})\}$$

$\Vdash_{\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}}} \mathsf{chairReady}$ $\times$ Dev is not on the PC

*The shortlist clause is **replaced**, not added to: this is not an extension of* $\mathsf{Conf}^{\mathsf{pend}}$.

PENDING DECISIONS

Policies so far

Conf	the running example: facts; reviewing, chair and integrity rules
Conf^{-}	Conf without the integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$
$\mathsf{Conf}^{\mathsf{pend}}$	Conf plus the clause Pend(Ada, Ben, Cleo)
$\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}}$	Conf plus the clause Pend(Ada, Ben, Cleo, Dev)

$$\mathsf{Pend}(x_1, \ldots, x_n) \;:=\; \forall X.\, (\mathsf{chair}(x_1) \supset X) \supset \cdots \supset (\mathsf{chair}(x_n) \supset X) \supset X$$

"one of $x_1, \ldots, x_n$ will chair, without saying which"

Every result in this talk names one of these policies; the newest is highlighted.

PART 6

Support is Search

Computing support by proof-search.

SUPPORT IS SEARCH

The problem

- $\forall\, \mathcal{Q} \supseteq \mathcal{P}$: every extension of the policy.
- $\forall A$: every atom, including ones nobody has written.

Taken literally, there is no obvious way to check it.

SUPPORT IS SEARCH

The approach



The quantifiers become ***fresh names*** *in proof-search — not a search through infinitely many cases.*

SUPPORT IS SEARCH

The clause language

$$H \;::=\; A \;\mid\; H \supset H \;\mid\; \forall X.\, H$$

- X ranges over *atoms* only.
- Implications may nest, so a goal can carry its own hypotheses.

A program is a finite set of clauses.

SUPPORT IS SEARCH

Not Horn clauses

- The paper calls the language “Horn”. Because implications nest, it is hereditary Harrop.
- With Horn clauses alone (no nesting), support gives **classical** logic (Sandqvist 2009).
- The encoding that follows needs the nesting.

SUPPORT IS SEARCH

Proof-search rules

Augment $\quad \mathcal{P} \vdash_{\mathsf{O}} D \supset G \;$ if $\; \mathcal{P}, D \vdash_{\mathsf{O}} G$

Generic $\quad \mathcal{P} \vdash_{\mathsf{O}} \forall X.G \;$ if $\; \mathcal{P} \vdash_{\mathsf{O}} G[\alpha/X], \;\; \alpha$ fresh

Backchain $\quad \mathcal{P} \vdash_{\mathsf{O}} A \;$ if $\; \forall \vec{X}.\, G_1 \supset \cdots \supset G_n \supset H \in \mathcal{P},$

$$H\sigma = A, \quad \text{and } \mathcal{P} \vdash_{\mathsf{O}} G_i\sigma \text{ for each } i$$

Goals drive the search; the program is consulted only at atoms (Miller, Nadathur, Pfenning, Scedrov 1991).

SUPPORT IS SEARCH

A correction to the rules

Some clauses conclude a *variable*: $\forall X.\ \cdots \supset X$. To use one, instantiate X with the **atom being proved**.

- The papers' Instance rule says to use a *fresh* atom instead.
- Their proofs, correctly, use the atom being proved.

SUPPORT IS SEARCH

The encoding

$$
\begin{aligned}
\ulcorner A \urcorner &= A \\
\ulcorner \mathbf{0} \urcorner &= \forall X.\, X \\
\ulcorner \varphi \Rightarrow \psi \urcorner &= \ulcorner \varphi \urcorner \supset \ulcorner \psi \urcorner \\
\ulcorner \varphi \wedge \psi \urcorner &= \forall X.\, (\ulcorner \varphi \urcorner \supset \ulcorner \psi \urcorner \supset X) \supset X \\
\ulcorner \varphi \vee \psi \urcorner &= \forall X.\, (\ulcorner \varphi \urcorner \supset X) \supset (\ulcorner \psi \urcorner \supset X) \supset X
\end{aligned}
$$

Each semantic clause, read as a program.

SUPPORT IS SEARCH

Reading the encoding

$$\forall X.\,(\varphi \supset X) \supset X$$

- In words: for every atom X, if φ would yield X, then X.
- A property is known by the atoms it yields — the observational reading again.

Continuation-passing style; the shape of Church encodings and $\top\top$-closure, with atoms as the observations.

SUPPORT IS SEARCH

Main theorem

$$\Vdash_{\mathcal{P}} \varphi \quad \text{if and only if} \quad \mathcal{P} \vdash_{\mathsf{O}} \ulcorner \varphi \urcorner$$

By simultaneous induction on φ. Gheorghiu, *Support is Search* (arXiv:2603.13018); instantiated to policies in arXiv:2603.13181.

SUPPORT IS SEARCH

Proof outline

$\forall\, \mathcal{Q} \supseteq \mathcal{P}$

becomes Augment.

$\mathcal{P} \cup \{\ulcorner \varphi \urcorner\}$ is the most general extension supporting φ.

$\forall A$, TO PROVE

becomes Generic.

A fresh atom stands for any atom.

$\forall A$, TO USE

becomes Backchain.

A disjunctive hypothesis is used by instantiating X with the atom being proved.

SUPPORT IS SEARCH

Policies so far

Conf	the running example: facts; reviewing, chair and integrity rules
Conf^{-}	Conf without the integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$
$\mathsf{Conf}^{\mathsf{pend}}$	Conf plus the clause Pend(Ada, Ben, Cleo)
$\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}}$	Conf plus the clause Pend(Ada, Ben, Cleo, Dev)

$$\mathsf{Pend}(x_1, \ldots, x_n) \;:=\; \forall X.\, (\mathsf{chair}(x_1) \supset X) \supset \cdots \supset (\mathsf{chair}(x_n) \supset X) \supset X$$

"one of $x_1, \ldots, x_n$ will chair, without saying which"

Every result in this talk names one of these policies; all four, for reference.

SUPPORT IS SEARCH

Example derivation (1/2)

Goal: $\mathsf{Conf}^{\mathsf{pend}} \vdash_{\mathsf{O}} \ulcorner \mathsf{chair}(\mathsf{ben}) \vee \mathsf{chair}(\mathsf{cleo}) \urcorner$

1. Generic: pick a fresh atom α.
2. Augment: assume $c_b \supset \alpha$ and $c_c \supset \alpha$. Goal: α.
3. Backchain on Pend(Ada, Ben, Cleo), with $X := \alpha$:

$$c_a \supset \alpha \qquad c_b \supset \alpha \qquad c_c \supset \alpha$$

The last two are hypotheses from step 2. One subgoal remains.

SUPPORT IS SEARCH

Example derivation (2/2)

4. $c_a \supset \alpha$: assume $\mathsf{chair}(\mathsf{ada})$.
5. With $\mathsf{author}(\mathsf{ada}, \mathsf{p007})$, separation of duty gives $\mathsf{violation}$.
6. The integrity clause gives $\forall X.X$; backchain with $X := \alpha$. ✓

*Ben and Cleo would each force α; Ada would force **everything**.*

PART 7

Implementation

Status, and a prototype.

IMPLEMENTATION

What the theorem provides

- **Finitary.** One finite program, one finite goal.
- **Sound and complete search.** A uniform proof exists iff $\Vdash_{\mathcal{P}} \varphi$.
- **Witnesses.** Every "yes" comes with a derivation.

IMPLEMENTATION

What the theorem does not provide

- **Termination.** Proof-search here can loop. “Decidable” needs an argument.
- **Off-the-shelf execution.** $\vdash_0$ is a *variant* of λProlog: it allows clauses that conclude a variable.
- λProlog forbids those clauses, so running the encoding needs a meta-interpreter.

IMPLEMENTATION

Prototype

- The three rules above, in ∼200 lines of Python.
- Fresh atoms chosen canonically: always the first unused α_k, so a repeated subproblem is recognised.
- Tabling: each atomic subproblem is solved once, as a least fixed point. A “no” means *no proof exists*.
- Growth guard: if the program grows too large, answer *unknown* rather than loop.

IMPLEMENTATION

Test: intuitionistic logic

$p \vee \neg p$	$\times$	$\neg\neg(p \vee \neg p)$	✓
$\neg\neg p \Rightarrow p$	$\times$	$(p \vee q) \wedge \neg p \Rightarrow q$	✓
$((p \Rightarrow q) \Rightarrow p) \Rightarrow p$	$\times$	$\neg\neg(((p \Rightarrow q) \Rightarrow p) \Rightarrow p)$	✓

Agrees with intuitionistic logic on every test.

IMPLEMENTATION

Policies so far

Conf	the running example: facts; reviewing, chair and integrity rules
Conf^{-}	Conf without the integrity clause $\mathsf{violation}(m, p) \supset \mathbf{0}$
$\mathsf{Conf}^{\mathsf{pend}}$	Conf plus the clause Pend(Ada, Ben, Cleo)
$\mathsf{Conf}^{\mathsf{pend}}_{\mathsf{dev}}$	Conf plus the clause Pend(Ada, Ben, Cleo, Dev)

$$\mathsf{Pend}(x_1, \ldots, x_n) \;:=\; \forall X.\, (\mathsf{chair}(x_1) \supset X) \supset \cdots \supset (\mathsf{chair}(x_n) \supset X) \supset X$$

"one of $x_1, \ldots, x_n$ will chair, without saying which"

Every result in this talk names one of these policies; all four, for reference.

IMPLEMENTATION

Results: reviewing

Conf	$\mathsf{conflicted}(\mathsf{ada}, \mathsf{p007})$	✓
Conf	$\mathsf{violation}(\mathsf{ada}, \mathsf{p007})$	×
Conf	$\neg\, \mathsf{canReview}(\mathsf{ada}, \mathsf{p007})$	✓
$\overline{\mathsf{Conf}}$	$\neg\, \mathsf{canReview}(\mathsf{ada}, \mathsf{p007})$	×
Conf	$\mathsf{author}(\mathsf{ben}, \mathsf{p007}) \Rightarrow \neg\, \mathsf{canReview}(\mathsf{ben}, \mathsf{p007})$	✓

IMPLEMENTATION

Results: pending decisions

Conf	$(c_a \vee c_b \vee c_c) \Rightarrow$ chairReady	✓
$\text{Conf}^{\text{pend}}$	$c_b \vee c_c$	✓
$\text{Conf}^{\text{pend}}$	chair(ben)	×
$\text{Conf}^{\text{pend}}_{\text{dev}}$	chairReady	×
Conf	$c_d \Rightarrow$ chairReady (the one new case)	×

Every verdict as predicted by hand; each under ***2 s****.*

IMPLEMENTATION

Termination

- First attempt: for each program, compute everything it derives — over every atom, plus a fresh one.
- On $\neg\neg(p \vee \neg p)$ it **diverges**.
- Each fresh atom is a new test; each test grows the program; each program needs a new fresh atom.

Goal-directed tabling terminates there — evidence, not proof.

IMPLEMENTATION

Ground policies

- A policy is ground if its rules contain no first-order variables.
- To ground a rule, replace it by all its instances over the names in the policy.

$\mathsf{pcMember}(m) \supset \mathsf{author}(m, p) \supset \mathsf{conflicted}(m, p)$ becomes

$\mathsf{pcMember}(\mathsf{ada}) \supset \mathsf{author}(\mathsf{ada}, \mathsf{p007}) \supset \mathsf{conflicted}(\mathsf{ada}, \mathsf{p007})$
$\mathsf{pcMember}(\mathsf{ben}) \supset \mathsf{author}(\mathsf{ben}, \mathsf{p007}) \supset \mathsf{conflicted}(\mathsf{ben}, \mathsf{p007})$ … and so on.

Each ground atom can be read as a propositional letter. The second-order $\forall X$ over atoms remains. The prototype grounds before it searches.

IMPLEMENTATION

Ground policies: a caveat

- Grounding fixes the names to those the policy already mentions.
- An extension may add a new person, Eve. The original rule applies to her; the grounded rules do not.
- So the two agree only on properties about names already present.

Enough for every example here — not the general first-order case.

IMPLEMENTATION

Decidability

QUESTION

Is $\Vdash_{\mathcal{P}} \varphi$ **decidable** for finite $\mathcal{P}$?

- Lower bound for free: at $\mathcal{P} = \emptyset$ it is IPL, so PSPACE-hard (Statman 1979).
- Conjecture: PSPACE-complete for ground policies — that is, no harder than propositional IPL.
- Possible route: show a proof needs no more fresh atoms than there are $\forall$s in the program and goal.

IMPLEMENTATION

Place in a verification workflow



Complementary to closed-world tools, not a competitor.

PART 8

Future Work

Resources, revocation, structure.

FUTURE WORK

Resources

- Single-use tokens; review quotas.
- Index support by resources: $\Gamma \Vdash_{\mathcal{P}}{}^{R} \varphi$.
- Separated resources compose.

Gheorghiu, Gu & Pym, *Inferentialist Resource Semantics* (MFPS XL, 2024). Here $\otimes$ and $*$ earn their names.

$$\frac{\Vdash_{\mathcal{P}}{}^{R} \varphi \qquad \Vdash_{\mathcal{P}}{}^{S} \psi}{\Vdash_{\mathcal{P}}{}^{R,S} \varphi * \psi}$$

, = separated resources

FUTURE WORK

Revocation and categorical structure

REVOCATION

Change that removes rules.

Needs a different ordering of policies. What replaces robustness?

STRUCTURE

Presheaves over the poset of policies.

Pym, Ritter, Robinson: $\vee$ is a join in a sublocale, not a coproduct.

FUTURE WORK

Broader programme

*A methodology for verification based on the **behavioural** view of systems.*

Access control is the first case study.

SUMMARY

Summary

- Judge a specification by its **behaviour under extension**.
- A pending decision is a disjunction you can reason *from*.
- Prohibitions must be modelled; silence is not robust.
- The quantifier over all extensions becomes proof-search: support is search.
- Execution works. Decidability is open.

REFERENCES

References

- *Verification of Robust Properties for Access Control Policies.* arXiv:2603.13181.
- *Support is Search.* arXiv:2603.13018.
- Sandqvist, *Base-extension semantics for IPL.* Logic J. IGPL **23** (2015).
- Miller, Nadathur, Pfenning, Scedrov, *Uniform proofs …* APAL **51** (1991).
- Gheorghiu, Gu & Pym, *Inferentialist Resource Semantics.* MFPS XL (2024).

Alexander V. Gheorghiu · University of Southampton & UCL

Thank you.