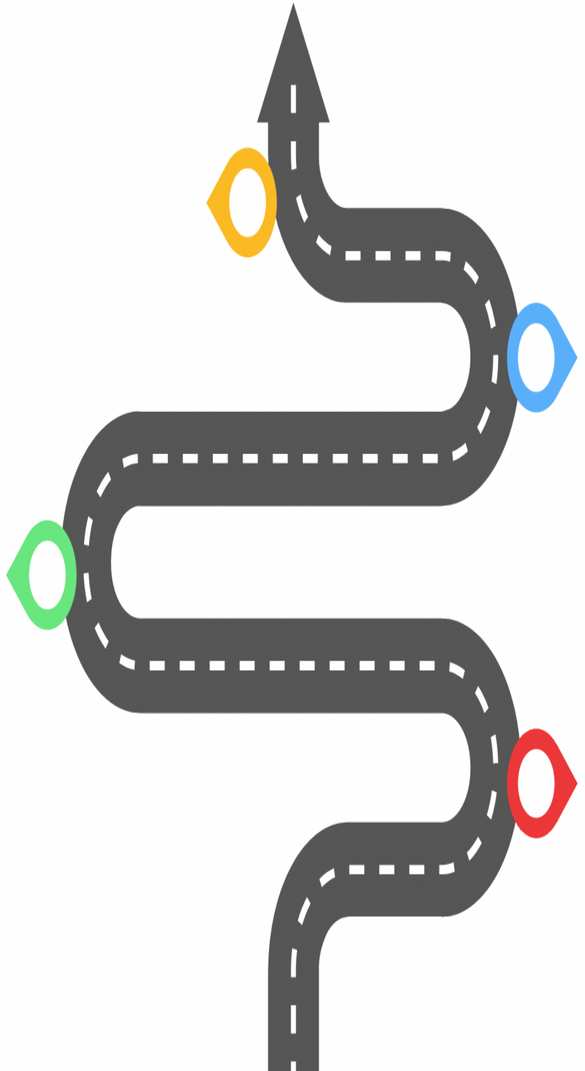
A Tour of Probabilistic Separation Logic

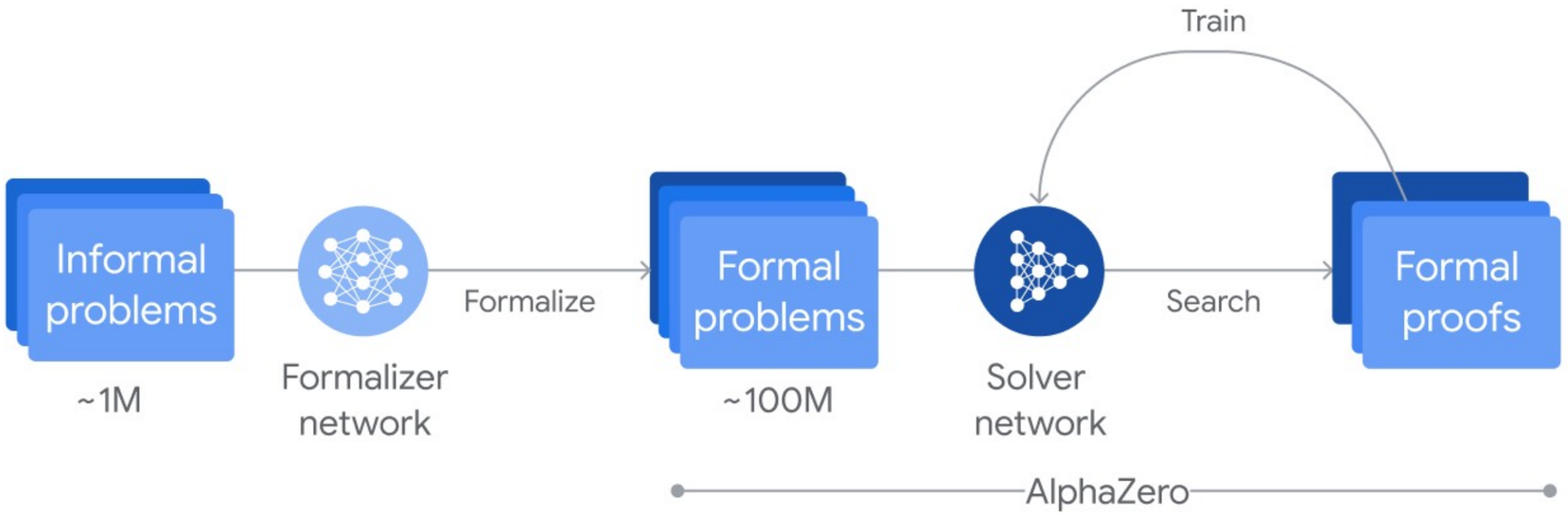


Jialu Bao

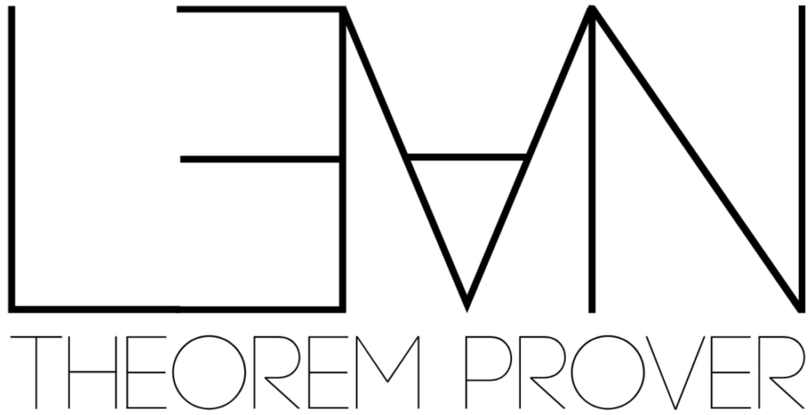
Aug. 2nd 2024


DETOUR

AI achieves silver-medal standard solving International Mathematical Olympiad problems



"Formal languages offer the critical advantage that proofs involving mathematical reasoning can be formally verified for correctness ... In contrast, natural language based approaches can hallucinate plausible but incorrect intermediate reasoning steps and solutions, despite having access to orders of magnitudes more data." (From Deepmind's Blogpost)



User writes both the theorem and proof in a **formal logic**

Lean checks whether the proof proves the theorem

Theorems as types, proofs as programs

Program C has type T $\iff$ Proof C proves theorem T


END
DETOUR


Formal Logic
for Programs

Logic

Syntax

- Formulas

$$P, Q ::= p \in \mathscr{AP} \mid \top \mid \bot \mid P \wedge Q \mid P \Rightarrow Q$$

- Rules

$$\frac{\Gamma \vdash \phi \quad \Gamma \vdash \psi}{\Gamma \vdash \phi \wedge \psi}\,(\wedge\mathbf{I}) \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \phi}\,(\wedge\mathbf{E1}) \qquad \frac{\Gamma \vdash \phi \wedge \psi}{\Gamma \vdash \psi}\,(\wedge\mathbf{E2})$$

Semantics

- Interpretation on a strcuture

A	B	$A \wedge B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

Program Logic

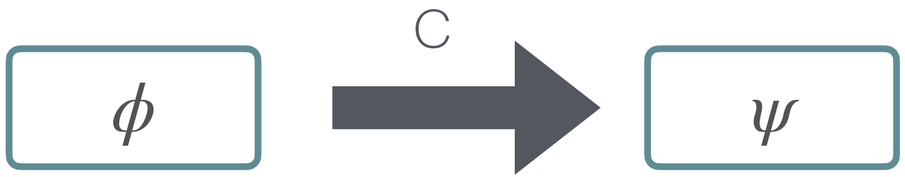
Judgement: $\{\phi\}\mathrm{C}\{\psi\}$ where C is a program and ϕ, ψ are logic formulas

Syntax

- Assertion logic rules
  - Ex. $\phi \wedge \psi \vdash \phi$
- Program rules
  - Ex. $$\frac{\{P\}\ c_1\ \{Q\} \qquad \{Q\}\ c_2\ \{R\}}{\{P\}\ c_1; c_2\ \{R\}}$$

Semantics

- Assertion logic semantics
- Program semantics



Programming Language

Syntax

Imp

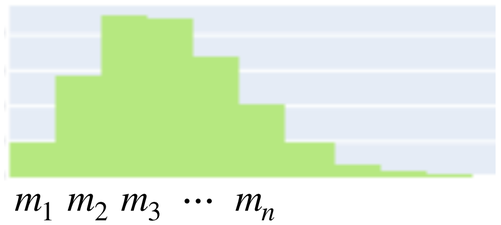
$$
\begin{aligned}
c ::= \ & \textbf{skip} \\
\mid \ & x := a \\
\mid \ & c_1; c_2 \\
\mid \ & \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\
\mid \ & \textbf{while } b \textbf{ do } c \\
\mid \ & x \sim \mu \\
\mid \ & \sout{observe(b)}
\end{aligned}
$$

PWhile

Semantics

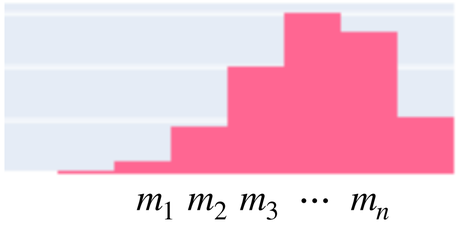
Distribution Transformer

A distribution over program states



C

A distribution over program states



Lower-level logic

- Ex. Coq, Lean
- Very expressive
- A full proof can be tedious

Higher-level Logic

- Domain-specific
- Encapsulation of key notions
- More succint proofs


Probabilistic
Independence

Probabilistic Independence

Definition: random variables X, Y independent iff,

$$\mathbb{P}(X, Y) = \mathbb{P}(X) \cdot \mathbb{P}(Y).$$

Intuition: the value of one variable does not give information about the other.

Example:

$X :\sim$ 

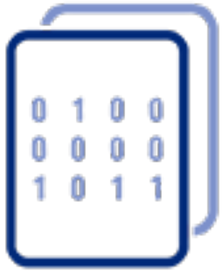
$Y :\sim$ 

Motivating Example

How do we ensure the security of an encryption algorithm?

Motivating Example

How do we ensure the security of an encryption algorithm?

Check 

Encrypted text

does not give information about  **.**

Plain text

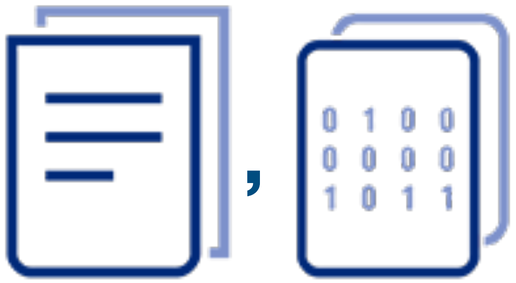
Motivating Example

How do we ensure the security of an encryption algorithm?

Encode  **as probabilistic programs;**

Encryption

Reasoning about independence can be tricky.

Verify  **are probabilistically independent.**

Plain text Encrypted text

One-time Pad Encryption

Fair coin

m :~ 

Fair coin

k :~ 

c = m xor k

Fact.

m, k independent

m, c **independent**

c, k independent

Preliminary Goal

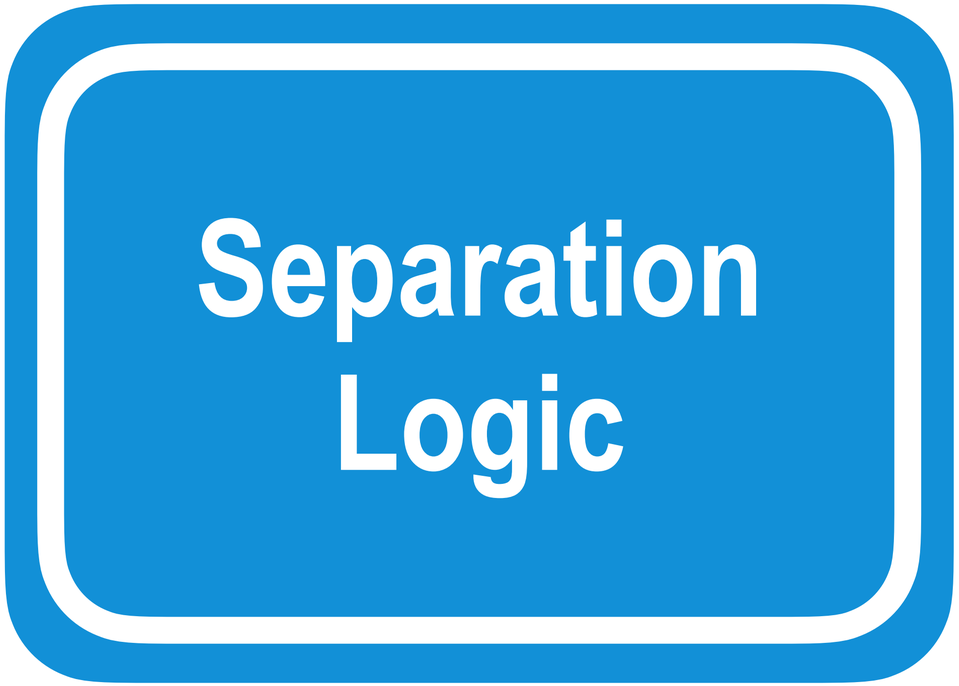
Program Logic with

- Judgment $\{\phi\}\mathsf{C}\{\psi\}$, where C is a PWhile program and ϕ, ψ are logic formulas
- Assertion for **expressing independence**
- Program rules for **reasoning about independence**

Two Events are independent, when they have no connexion one with the other, and that the happening of one neither forwards nor obstructs the happening of the other.

(*The Doctrine of Chances,* 1738, Abraham de Moivre)

Independence is a kind of separation


Separation
Logic

Separation Logic is a Program Logic

It has judgements of form $\{\phi\}\mathsf{C}\{\psi\}$ where C is a program and ϕ, ψ are formulas in **Bunched Logic**

Bunched Logic (O'Hearn and Pym 1999)

Syntax $P, Q ::= p \in \mathscr{AP} \mid \top \mid \bot \mid P \wedge Q \mid P \Rightarrow Q \mid I \mid P * Q$

The star $*$ is another conjunction and I is its unit.

$$\frac{P \vdash R \qquad Q \vdash S}{P * Q \vdash R * S}\ *\text{-Conj} \qquad \frac{}{P * Q \vdash Q * P}\ *\text{-Comm}$$

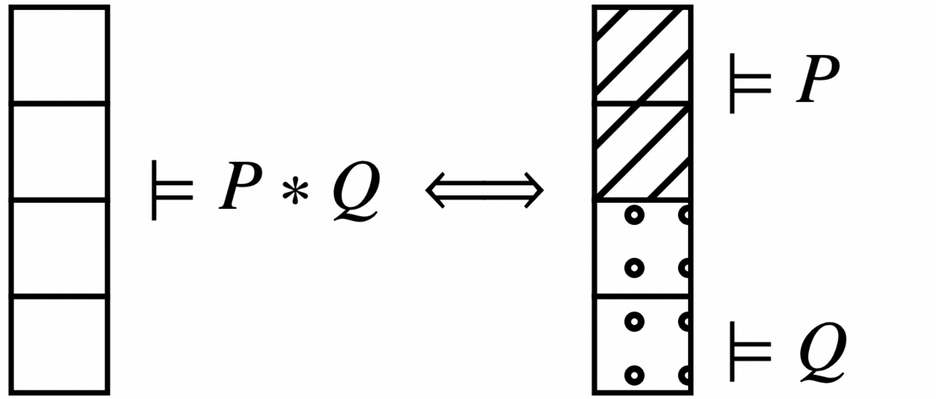
$$\frac{}{(P * Q) * R \dashv\vdash P * (Q * R)}\ *\text{-Assoc} \qquad \frac{}{P \dashv\vdash P * I}\ *\text{-Unit}$$

Bunched Logic (O'Hearn and Pym 1999)

Key difference $A \vdash A \wedge A$ but not $A \vdash A * A$

$A \wedge A \vdash A$ but not $A * A \vdash A$ in general

Semantics



Bunched Logic (O'Hearn and Pym 1999)

Resource Semantics

Bunched logic formulas are interpreted through a Kripke resource monoid, which is a set M with

- a partial binary operation $\circ : M \times M \rightharpoonup M$ that is
  - associative
  - commutative
- an identity element $e \in M$

Bunched Logic (O'Hearn and Pym 1999)

Resource Semantics

We inductively define the satisfaction relations on $m \in M$ and formulas:

- $m \vDash p$ iff $m \in \mathscr{V}(p)$
- ...
- $m \vDash P \wedge Q$ iff $m \vDash P$ and $m \vDash Q$
- $m \vDash P * Q$ iff exist m_1, m_2 with $m_1 \circ m_2$ defined and $m_1 \circ m_2 \sqsubseteq m$ such that $m_1 \vDash P$ and $m_2 \vDash Q$

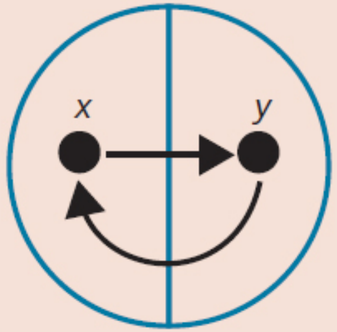
$\models (x \mapsto y) \wedge (y \mapsto x)$

$\not\models (x \mapsto y) * (y \mapsto x)$

value	42	42
location	42	42

$\models (x \mapsto y) \wedge (y \mapsto x)$

$(x \mapsto y) * (y \mapsto x)$



value	10	42
location	42	10

Adapted from an image in "Separation Logic" in CACM

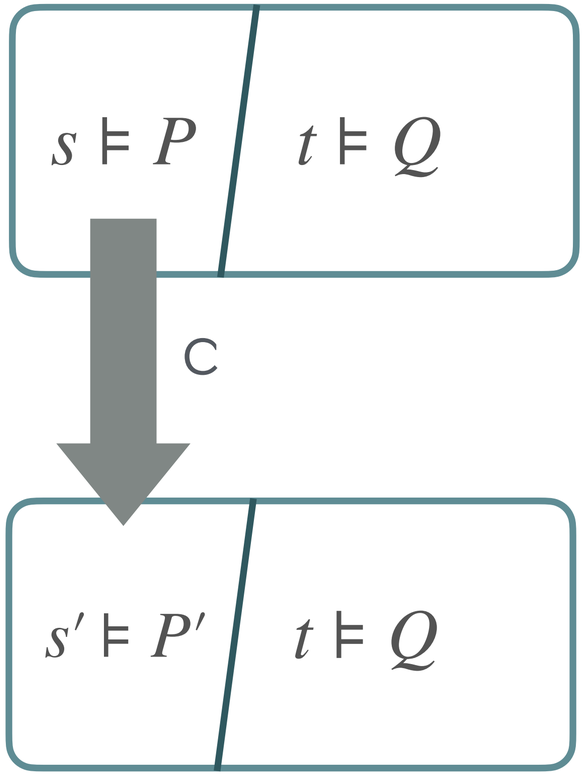
Separation Logic (Reynolds 2002)

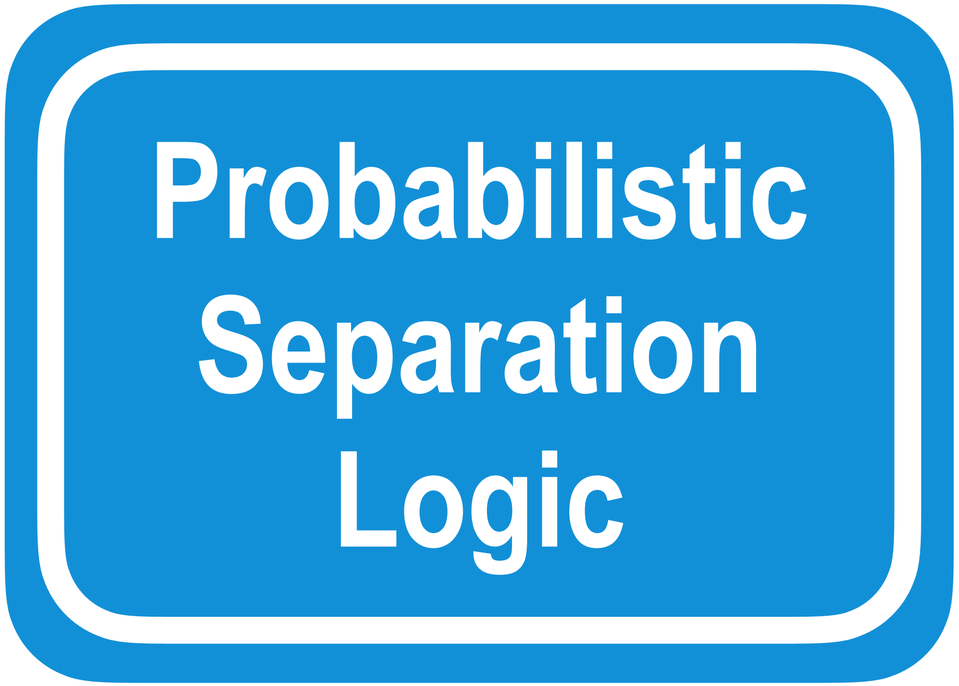
$\{\phi\}\mathsf{C}\{\psi\}$ where C is a program and ϕ, ψ are bunched logic formulas

Signature "Frame Rule"

$$\frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad c \text{ does not modifies } FV(\eta) \qquad \text{side conditions}}{\vdash \{\phi\ *\ \eta\}\ c\ \{\psi\ *\ \eta\}}$$

Modular Reasoning of Program States




Probabilistic
Separation
Logic

Kripke Semantics

- We need a commutative monoid M
  - Let M be the set of distributions. For distributions $f : X \to [0,1]$ and $g : Y \to [0,1]$, define $f \circ g$ to be their independent product. (Barthe et al. 2020 and Li et al. 2023)
- We need a valuation on atomic propositions:
  - Define $m \vDash \langle X \rangle$ iff variables in X are in dom(m)
- **Theorem**. $m \vDash \langle X \rangle * \langle Y \rangle$ iff variables X, Y are independent in m.

Program Logic Rules

PWhile

$$
\begin{aligned}
c ::= \ & \textbf{skip} \\
\mid \ & x := a \\
\mid \ & c_1; c_2 \\
\mid \ & \textbf{if } b \textbf{ then } c_1 \textbf{ else } c_2 \\
\mid \ & \textbf{while } b \textbf{ do } c \\
\mid \ & x :\sim \mu
\end{aligned}
$$

$$\textsc{RAssn}\ \frac{x_r \notin FV(e_r)}{\vdash \{\top\}\ x_r \leftarrow e_r\ \{x_r \sim e_r\}} \qquad \textsc{RSamp}\ \frac{}{\vdash \{\top\}\ x_r \xleftarrow{\$} \mathbf{U}_S\ \{\mathbf{U}_S[x_r]\}}$$

$$\textsc{RDCond}\ \frac{\vdash \{\phi \wedge b \sim tt\}\ c\ \{\psi\} \qquad \vdash \{\phi \wedge b \sim ff\}\ c'\ \{\psi\} \qquad \models \phi \to (b \sim tt \vee b \sim ff)}{\vdash \{\phi\}\ \textbf{if}_R\ b\ \textbf{then}\ c\ \textbf{else}\ c'\ \{\psi\}}$$

$$\textsc{RCond}\ \frac{\vdash \{\phi * b \sim tt\}\ c\ \{\psi * b \sim tt\} \qquad \vdash \{\phi * b \sim ff\}\ c'\ \{\psi * b \sim ff\} \qquad \psi \in \mathrm{SP}}{\vdash \{\phi * \mathbf{D}[b]\}\ \textbf{if}_R\ b\ \textbf{then}\ c\ \textbf{else}\ c'\ \{\psi * \mathbf{D}[b]\}}$$

$$\textsc{Weak}\ \frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad \models \phi' \to \phi \wedge \psi \to \psi'}{\vdash \{\phi'\}\ c\ \{\psi'\}} \qquad \textsc{True}\ \frac{}{\vdash \{\top\}\ c\ \{\top\}}$$

$$\textsc{Conj}\ \frac{\vdash \{\phi_1\}\ c\ \{\psi_1\} \qquad \vdash \{\phi_2\}\ c\ \{\psi_2\}}{\vdash \{\phi_1 \wedge \phi_2\}\ c\ \{\psi_1 \wedge \psi_2\}} \qquad \textsc{Case}\ \frac{\vdash \{\phi_1\}\ c\ \{\psi_1\} \qquad \vdash \{\phi_2\}\ c\ \{\psi_2\}}{\vdash \{\phi_1 \vee \phi_2\}\ c\ \{\psi_1 \vee \psi_2\}}$$

$$\textsc{RCase}\ \frac{\vdash \{\phi * b \sim tt\}\ c\ \{\psi * b \sim tt\} \qquad \vdash \{\phi * b \sim ff\}\ c\ \{\psi * b \sim ff\} \qquad \psi \in \mathrm{SP}}{\vdash \{\phi * \mathbf{D}[b]\}\ c\ \{\psi * \mathbf{D}[b]\}}$$

$$\textsc{Const}\ \frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad FV(\eta) \cap MV(c) = \emptyset}{\vdash \{\phi \wedge \eta\}\ c\ \{\psi \wedge \eta\}}$$

$$\textsc{Frame}\ \frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad FV(\eta) \cap MV(c) = \emptyset \qquad FV(\psi) \subseteq T \cup RV(c) \cup WV(c) \qquad \models \phi \to \mathbf{D}[T \cup RV(c)]}{\vdash \{\phi * \eta\}\ c\ \{\psi * \eta\}}$$

Samp

$$\frac{}{\vdash \{\ \top\ \}\ x :\sim \mu\ \{x \sim \mu\}}$$

Frame

$$\frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad c \text{ does not modifies } FV(\eta) \qquad \text{side conditions}}{\vdash \{\phi\ *\ \eta\}\ c\ \{\psi\ *\ \eta\}}$$

One-time Pad Encryption

$\{ \top \}$

m :~ 

$\{m \sim \mathrm{Bern}(0.5)\}$

$\{m \sim \mathrm{Bern}(0.5) * \top\}$

k ~ 

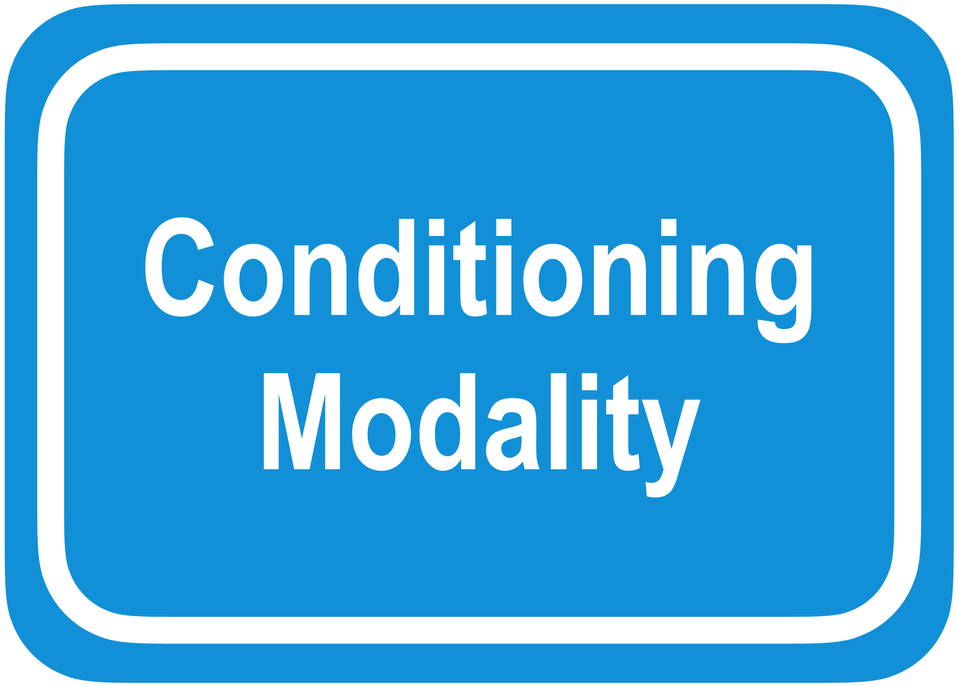
$\{m \sim \mathrm{Bern}(0.5) * k \sim \mathrm{Bern}(0.5)\}$

c = m xor k

$$\frac{}{\vdash \{ \top \}\ x :\sim \mu\ \{x \sim \mu\}}\ \text{Samp}$$

$$\frac{}{P \dashv\vdash P * I}\ *\text{-Unit}$$

$$\frac{\vdash \{\phi\}\ c\ \{\psi\} \qquad \cdots}{\vdash \{\phi\ *\ \eta\}\ c\ \{\psi\ *\ \eta\}}\ \text{Frame}$$

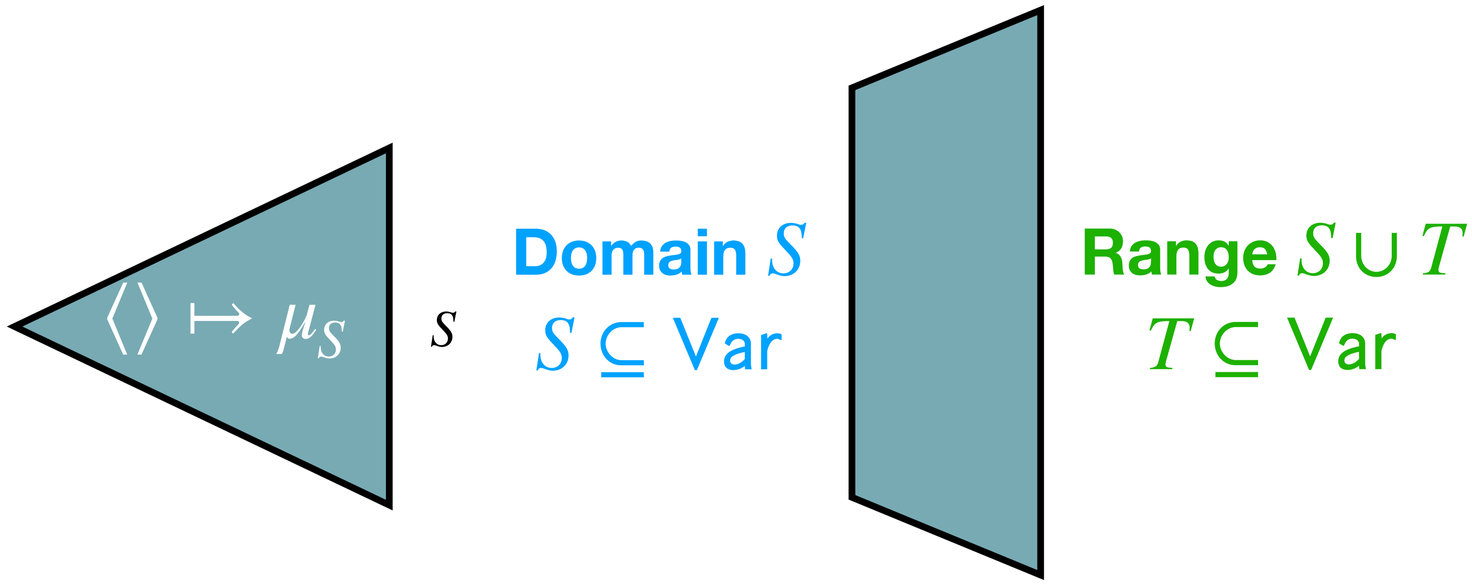

Conditioning
Modality

Decomposition by Conditioning

Given a distribution f over variables $S \cup T$, we can decompose f into

- The marginal distribution μ_S of f on S
- A Markov kernel κ from outcomes on S to distributions over $S \cup T$
  - $\kappa(\omega) = f\,|_{\omega}$

Write $f = (\mu_S \texttt{ >>= } \kappa)$.



Conditioning Modality (Bluebell, in submission)

- New formula: $\mathsf{C}_{\mu} v \,.\, P(v)$
- Define a distribution $f \vDash \mathsf{C}_{\mu} v \,.\, P(v)$ if there exists κ such that

  $f = (\mu \ggg= \kappa)$ and for all value v in the support of μ, $\kappa(v) \vDash P(v)$.
- How to make sure that κ is f conditioned on some variable X?

Almost Sure Equivalence (Li et al. 2023)

- New formula: $[e_1 = e_2]$
- Define a distribution $f \vDash [e_1 = e_2]$ if for any outcome ω in the support of f, the value of e_1, e_2 evaluated on ω are equal.
- Importantly, $[e_1 = e_2] \wedge P$ iff $[e_1 = e_2] * P$

Conditioning Modality (Bluebell, in submission)

- How to make sure that κ is f conditioned on some variable X?
  - Assert $\mathsf{C}_{\mu} v \,.\, [X = v] * P(v)$

Rules for Conditioning Modality (Bluebell, in submission)

- It commutes with many connectives in the logic.

C-FRAME

$$P * \boldsymbol{C}_\mu\, v.\, K(v) \vdash \boldsymbol{C}_\mu\, v.\, (P * K(v))$$

C-FOR-ALL

$$\boldsymbol{C}_\mu\, v.\, \forall x : X.\, Q(v, x) \vdash \forall x : X.\, \boldsymbol{C}_\mu\, v.\, Q(v, x)$$

C-AND

$$\frac{\mathrm{idx}(K_1) \cap \mathrm{idx}(K_2) = \emptyset}{\boldsymbol{C}_\mu\, v.\, K_1(v) \wedge \boldsymbol{C}_\mu\, v.\, K_2(v) \vdash \boldsymbol{C}_\mu\, v.\, (K_1(v) \wedge K_2(v))}$$

C-SKOLEM

$$\frac{\mu : \mathbb{D}(\Sigma_A)}{\boldsymbol{C}_\mu\, v.\, \exists x : X.\, Q(v, x) \vdash \exists f : A \to X.\, \boldsymbol{C}_\mu\, v.\, Q(v, f(v))}$$

SURE-STR-CONVEX

$$\boldsymbol{C}_\mu\, v.\, (K(v) * \lceil E\langle i \rangle \rceil) \vdash \lceil E\langle i \rangle \rceil * \boldsymbol{C}_\mu\, v.\, K(v)$$

Rules for Conditioning Modality (Bluebell, in submission)

- It inherits rules from monadic laws.

C-UNIT-L

$$C_{\delta_{v_0}}\, v.\, K(v) \dashv\vdash K(v_0)$$

C-UNIT-R

$$E\langle i\rangle \sim \mu \dashv\vdash C_{\mu}\, v.\, \lceil E\langle i\rangle = v\rceil$$

C-ASSOC

$$\frac{\mu_0 = \mathbf{bind}(\mu, \boldsymbol{\lambda} v.\ (\mathbf{bind}(\kappa(v), \boldsymbol{\lambda} w.\ \mathbf{return}(v, w))))}{C_{\mu}\, v.\, C_{\kappa(v)}\, w.\, K(v, w) \vdash C_{\mu_0}\, (v, w).\, K(v, w)}$$

C-UNASSOC

$$C_{\mathbf{bind}(\mu,\kappa)}\, w.\, K(w) \vdash C_{\mu}\, v.\, C_{\kappa(v)}\, w.\, K(w)$$

Rules for Conditioning Modality (Bluebell, in submission)

- It also has other interesting rules. For example,

$$\textsc{C-TRANSF}\quad \frac{f : \ \mathrm{supp}(\mu') \to \mathrm{supp}(\mu) \ \ \text{bijective} \qquad \forall b \in \mathrm{supp}(\mu').\, \mu'(b) = \mu(f(b))}{\boldsymbol{C}_{\mu}\, a.\, K(a) \vdash \boldsymbol{C}_{\mu'}\, b.\, K(f(b))}$$

One-time Pad Encryption

$\{\mathsf{m} \sim \mathsf{Bern}(0.5) * \mathsf{k} \sim \mathsf{Bern}(0.5) * \mathsf{T}$

c = m xor k

$\{\mathsf{m} \sim \mathsf{Bern}(0.5) * \mathsf{k} \sim \mathsf{Bern}(0.5) * [\mathsf{c}$

$\{\complement_{\mathsf{Bern}(0.5)} w \,.\, [m = w] * \complement_{\mathsf{Bern}(0.5)} v \,.$

C-TRANSF

$$\frac{f : \ \mathrm{supp}(\mu') \to \mathrm{supp}(\mu) \ \ \text{bijective} \qquad \forall b \in \mathrm{supp}(\mu').\, \mu'(b) = \mu(f(b))}{\boldsymbol{C}_{\mu}\, a.\, K(a) \vdash \boldsymbol{C}_{\mu'}\, b.\, K(f(b))}$$

$$\left\{ \complement_{\mathsf{Bern}(0.5)} w \,.\, [m = w] * \begin{cases} [w = 1] \implies \complement_{\mathsf{Bern}(0.5)} v \,.\, [c = v] \\ [w = 0] \implies \complement_{\mathsf{Bern}(0.5)} v \,.\, [c = \neg v] \end{cases} \right\}$$

By C-Cons

$\{\complement_{\mathsf{Bern}(0.5)} w \,.\, [m = w] * \complement_{\mathsf{Bern}(0.5)} v \,.\, [c = v]\}$

By C-Transf

One-time Pad Encryption

$$\{\complement_{\mathsf{Bern}(0.5)} w \,.\, [m = w] \ * \ \complement_{\mathsf{Bern}(0.5)} v \,.\, [c = k]\}$$

SURE-MERGE

$$\lceil E_1\langle i\rangle\rceil * \lceil E_2\langle i\rangle\rceil \dashv\vdash \lceil (E_1 \wedge E_2)\langle i\rangle\rceil$$

$$\{\complement_{\mathsf{Bern}(0.5)} w \,.\, \complement_{\mathsf{Bern}(0.5)} v \,.\, [m = w \wedge c = v]\}$$

By C-Frame, Sure-Merge

$$\{\complement_{\mathsf{Bern}(0.5)\otimes\mathsf{Bern}(0.5)} v \,.\, [m = w \wedge c = v]\}$$

By C-Assoc

$$\{(m, k) \sim \mathsf{Bern}(0.5) \otimes \mathsf{Bern}(0.5)\}$$

By C-Unit

$$\{m \sim \mathsf{Bern}(0.5) \ * \ k \sim \mathsf{Bern}(0.5)\}$$

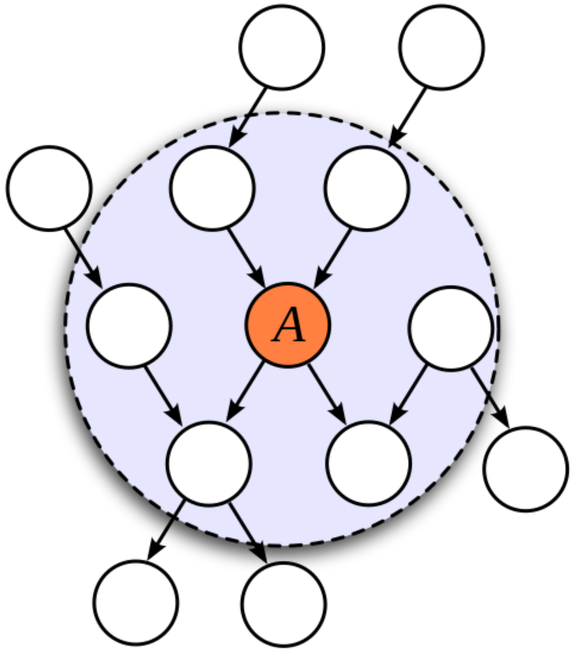
By Prod-Split

PROD-SPLIT

$$(E_1\langle i\rangle, E_2\langle i\rangle) \sim \mu_1 \otimes \mu_2 \vdash E_1\langle i\rangle \sim \mu_1 * E_2\langle i\rangle \sim \mu_2$$

More Examples

- Security of a multi-party addition protocol
- Quality of random bits generated by Von Neumann Extractor
- Correctness of a Reservoir Sampling algorithm
- Probabilistic Relational Hoare Logic style reasoning
- Bound of a Monte-carlo min and max estimation algorithm
- Conditional Independence in Markov Blanket



Current Status

Separation Logic for (In)dependencies

- **Independence:** Probabilistic Separation Logic (Hsu, Barthe and Liao, 2020)
- **Conditional Independence:**
  - A Bunched Logic for Conditional Independence (Bao, Docherty, Hsu and Silva 2021)
  - Lilac: A Modal Separation Logic for Conditional Probability (Li, Ahmed and Holtzen 2023)
  - Bluebell: An Alliance of Relational Lifting and Independence For Probabilistic Reasoning (Bao, D'osualdo and Farzan, in submission)
- Other (in)dependencies:
  - A Separation Logic for **Negative Dependence** (Bao, Gaboardi, Hsu, Tussoretti 2022)
  - On Separation Logic, **Computational Independence**, and Pseudorandomness (Dal Lago, Davol and Kapron 2024)

On the Theory Side

- A Nominal Approach to Probabilistic Separation Logic (Li, Aytac, Johnson-Freyd, Ahmed and Holtzen, 2024)
- A Categorical Approach to DIBI Models (Gu, Bao, Hsu, Silva and Zanasi, 2024)

On the Implementation Side

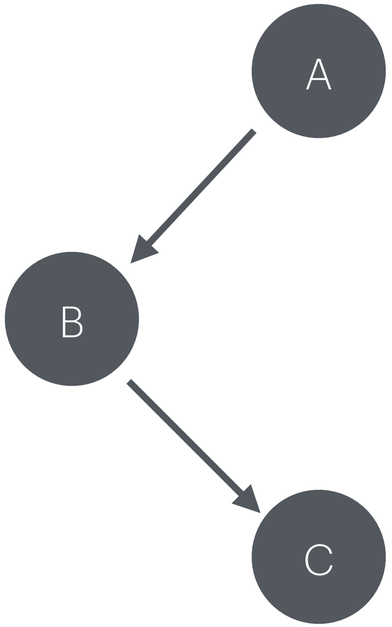
- Mechanization using Coq
  - Combining Classical and Probabilistic Independence Reasoning to Verify the Security of Oblivious Algorithms (Yan et al. 2024)
- I have an ongoing work on automating probabilistic separation logic for independence (without conditioning modality)

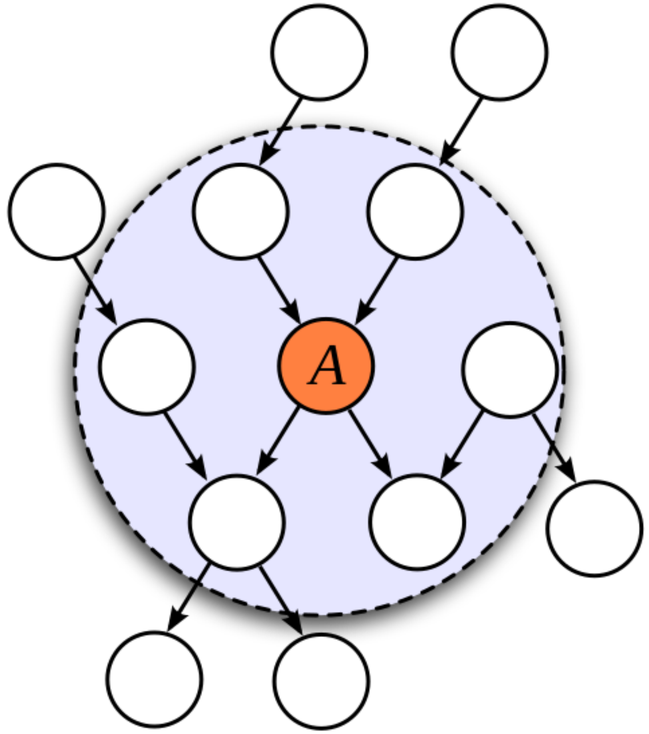
Discussion

Automating the Proof Construction using AI?

- Why it may be interesting to train AI to do it?
  - Soundness of a formal logic guarantees no hallucinations
- Potential advantages compared to using general theorem prover's language
  - Encapsulation of key concepts lead to more succinct proof
  - Locality of proofs due to the frame rule
- Limitation:
  - Tailored for specific properties, though different models of one logic can share one automation.

Usage in Variable Elimination?






Thanks!