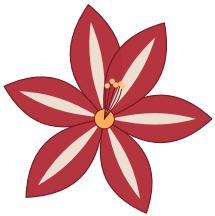
First Steps Towards Probabilistic Iris: Harmonizing Independence, Conditioning, and Dynamic Heap Allocation

Iris Workshop 2026

Janine Lohse, Tim Rohde, Jimmy Xin, Niklas Mück, Iona Kuhn
Deepak Garg, Derek Dreyer, Emanuele D'Osualdo

June 9, 2026



Probabilistic Program Logics

Distributional Probabilistic Logics (DPLs)	**Specialized Probabilistic Logics (SPLs)**
Native **assertions over distribution** of program output/state	Traditional assertions over output/state plus **probabilistic resources**
PSL [Barthe et al. 2020] Lilac [Li et al. 2023] Bluebell [Bao et al. 2025] pcOL [Zilberstein et al. 2026]	Eris [Aguirre et al. 2024] Tachis [Haselwarter et al. 2024] Clutch [Gregersen et al. 2024] Coneris [Li et al. 2025]
$\{\mathit{True}\}\ \mathit{flip}()\ \{V.\ V \sim \mathit{Ber}_{1/2}\}$	$\{\frac{1}{2}(0.5)\}\ \mathit{flip}()\ \{v.v = 1\}$

Probabilistic Program Logics

Distributional Probabilistic Logics (DPLs)	**Specialized Probabilistic Logics (SPLs)**
Native **assertions over distribution** of program output/state	Traditional assertions over output/state plus **probabilistic resources**
Can support **wide range of properties** including (conditional) independence	focus on **specific property** of interest e.g. expected values, error bounds
	often superior language feature support + fancy Iris reasoning techniques + strong modularity

DPLs have potential for being a **general-purpose foundation** for probabilistic logics, but so far are **lagging behind** SPLs in e.g. supported language features

Independence as Separation [Barthe et al. 2020]

Example: one time pad encryption: M message, K key:

$$M \sim Ber_p * K \sim Ber_{1/2}$$

means that M and K are **probabilistically independent**

Separating Conjunction

$\phi * \psi :=$ can separate probability space into independent parts satisfying ϕ and ψ

One time pad encryption correctness:

$$M \sim Ber_p * K \sim Ber_{1/2} \vdash M \sim Ber_p * (K \text{ xor } M) \sim Ber_{1/2}$$

The Conditioning Modality [Li et al. 2023]

Conditioning (discrete setting): probabilistic *case distinction* $C_\mu v.\ \Phi(v)$

$$M \sim Ber_p * K \sim Ber_{1/2} \vdash C_{Ber_p} v. \begin{cases} [M=0] * [K \text{ xor } M = K] * K \sim Ber_{1/2} & \text{if } v = 0 \\ [M=1] * [K \text{ xor } M = \neg K] * K \sim Ber_{1/2} & \text{else} \end{cases}$$

$$\vdash C_{Ber_p} v. \begin{cases} [M=0] * (K \text{ xor } M) \sim Ber_{1/2} & \text{if } v = 0 \\ [M=1] * (K \text{ xor } M) \sim Ber_{1/2} & \text{else} \end{cases}$$

$$\vdash (K \text{ xor } M) \sim Ber_{1/2} * C_{Ber_p} v. \begin{cases} [M=0] & \text{if } v = 0 \\ [M=1] & \text{else} \end{cases}$$

$$\vdash (K \text{ xor } M) \sim Ber_{1/2} * M \sim Ber_p$$

The Gap

All existing logics with support for independence reasoning
do not support dynamic memory allocation
...let alone more sophisticated Iris-style ghost state

The Challenge of Dynamic Allocation

Probabilistic Logics
Separation = independence
$M \sim Ber_p * K \sim Ber_{1/2}$

$+?$

Classic Separation Logic
Separation = heap disjointness
$\ell_1 \mapsto v_1 * \ell_2 \mapsto v_2$

Approach by PSL and pcOL: Separation means independence **and** heap disjointness:

Separating Conjunction in PSL/pcOL

$\phi * \psi \triangleq$ can separate probability space into two independent parts **over disjoint parts of the storage** that satisfy ϕ and ψ

$\Rightarrow$ in PSL/pcOL: $\phi * \psi$ requires heap disjointness **across all random outcomes**

The Challenge of Dynamic Allocation

in PSL/pcOL: $\phi * \psi$ requires heap disjointness across all random outcomes

Definition does not work with dynamic allocation!

Example

$\{\top\}$ (**if** flip() **then ref** 0); (**ref** flip(), **ref** flip()) $\{(L_1, L_2).\ L_1 \rightsquigarrow \mathit{Ber}_{1/2} * L_2 \rightsquigarrow \mathit{Ber}_{1/2}\}$

Assume det. allocator that allocates $\ell_0, \ell_1, \ell_2, \ldots$ in order

$\Rightarrow$ First flip false $\rightarrow (\ell_0, \ell_1)$ returned, first flip true $\rightarrow (\ell_1, \ell_2)$ returned

$\Rightarrow$ ℓ_1 can belong to either L_1 or L_2: no global disjointness!

$\Rightarrow$ Triple is not valid with the PSL/pcOL model of separation!

Key Contributions

Amaryllis

First logic to support independence + **dynamic heap allocation**

- General construction for lifting ownership of Iris resource to probabilistic **ownership per random outcome**
- Instance: Novel model that requires heap disjointness only per random outcome instead over all of them
- Fully mechanized in Rocq, including Iris proof mode

Limitations: Only finite distributions → only terminating programs.
No step indexing, invariants, concurrency

Amaryllis' Model of Per-Outcome Disjointness

Key Idea

Require heap disjointness only **per random outcome** instead over all of them

Amaryllis' Model explicitly tracks random choices (indexed valuations)

RID: set of *random choice identifiers*

Model

$$\mathbb{P}(\mathsf{RID} \to \mathbb{B}) \times RV_{\mathsf{Heap}} \qquad \text{where } RV_T \triangleq (\mathsf{RID} \to \mathbb{B}) \to T$$

$(\phi * \psi)(\mathcal{P}, R) \triangleq$ can separate $\mathcal{P}$ into independent $\mathcal{P}_1, \mathcal{P}_2$ and R into two pointwise disjoint R_1, R_2 s.t. $\phi(\mathcal{P}_1, R_1)$ and $\psi(\mathcal{P}_2, R_2)$

Logic Assertions

Model

$$\mathbb{P}(\mathsf{RID} \to \mathbb{B}) \times RV_{\mathsf{Heap}} \qquad \text{where } RV_T \triangleq (\mathsf{RID} \to \mathbb{B}) \to T$$

$$(V \sim \mu)(\mathcal{P}, R) \triangleq V \text{ is distributed as } \mu \text{ in } \mathcal{P}$$

$$(L \mapsto V)(\mathcal{P}, R) \triangleq \forall \rho \in \mathsf{supp}(\mathcal{P}).\ (L(\rho) \mapsto V(\rho))(R(\rho))$$

$$L \rightsquigarrow \mu \triangleq \exists V.\ V \sim \mu \wedge L \mapsto V$$

Amaryllis Logic

Idea: Use random expression variable in the middle of triple!

	Classic Separation Logic	**Amaryllis**
Points-tos	$\ell \mapsto v$ with $\ell : \mathit{Loc}, v : \mathit{Val}$	$L \mapsto V$ with $L : \mathit{RV}_{\mathit{Loc}}, V : \mathit{RV}_{\mathit{Val}}$
Allocation Rule	$\{\top\}\ \mathbf{ref}\ v\ \{\ell.\ \ell \mapsto v\}$	$\{\top\}\ \rho.\ \mathbf{ref}\ V(\rho)\ \{L.\ L \mapsto V\}$
Load	$\{\ell \mapsto v\}\ !\ell\ \{w.\ v = w\}$	$\{L \mapsto V\}\ \rho.\ !L(\rho)\ \{W.\ V \approx W\}$
Store	$\{\ell \mapsto v\}\ \ell \leftarrow w\ \{\ell \mapsto w\}$	$\{L \mapsto V\}\ \rho.\ L(\rho) \leftarrow W(\rho)\ \{L \mapsto W\}$
Sample	-	$\{\top\}\ \mathrm{flip}(p)\ \{V.\ V \sim \mathit{Ber}_p\}$

Example Verification in Amaryllis

Example

$\{\top\}\ (\textbf{if}\ \mathsf{flip}()\ \textbf{then}\ \textbf{ref}\ 0);\ (\textbf{ref}\ \mathsf{flip}(), \textbf{ref}\ \mathsf{flip}())\ \{(L_1, L_2).\ L_1 \rightsquigarrow \mathit{Ber}_{1/2} * L_2 \rightsquigarrow \mathit{Ber}_{1/2}\}$

Proof idea:
Condition on outcome of first flip and reason **reason case-by-case** using C-lift!

$$\text{C-LIFT}\ \frac{\forall v.\ \{\Phi(v)\}\ E\ \{\Psi(v)\}}{\{C_\mu v.\ \Phi(v)\}\ E\ \{C_\mu v.\ \Psi(v)\}}$$

First, show:

$$\{\top\}\ (\mathbf{ref}\ \mathsf{flip}(), \mathbf{ref}\ \mathsf{flip}())\ \{\exists L_1, L_2.\ V \approx (L_1, L_2) * L_1 \rightsquigarrow \mathit{Ber}_{1/2} * L_2 \rightsquigarrow \mathit{Ber}_{1/2}\}$$

Then:

$$\begin{aligned}
\{\top\}\ (\mathbf{if}\ \mathsf{flip}()\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{V.\ \exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{V_f \sim \mathit{Ber}_{1/2}\}\ (\mathbf{if}\ V_f\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{V.\ \exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{C_{\mathit{Ber}_{1/2}} v.\ V_f \approx v\}\ (\mathbf{if}\ V_f\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{V.\ C_{\mathit{Ber}_{1/2}} v.\ \exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{V_f \approx v\}\ (\mathbf{if}\ V_f\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{\exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{\top\}\ (\mathbf{if}\ v\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{\exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\}
\end{aligned}$$

Do case analysis on v:

$$\begin{aligned}
\{\top\}\ (\mathbf{if}\ \mathsf{true}\ \mathbf{then}\ \mathbf{ref}\ 0);\ \ldots \quad & \{\exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{\top\}\ \mathbf{ref}\ 0;\ (\mathbf{ref}\ \mathsf{flip}(), \mathbf{ref}\ \mathsf{flip}()) \quad & \{\exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\} \dashv \\
\{L \mapsto 0\}\ (\mathbf{ref}\ \mathsf{flip}(), \mathbf{ref}\ \mathsf{flip}()) \quad & \{\exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots\}
\end{aligned}$$

Example Verification in Amaryllis

Example

$\{\top\}\ (\textbf{if}\ \mathsf{flip}()\ \textbf{then}\ \textbf{ref}\ 0);\ (\textbf{ref}\ \mathsf{flip}(), \textbf{ref}\ \mathsf{flip}())\ \{(L_1, L_2).\ L_1 \rightsquigarrow \mathit{Ber}_{1/2} * L_2 \rightsquigarrow \mathit{Ber}_{1/2}\}$

Key Proof Step

In Amaryllis Model, the postcondition is **convex**:

$$C_{\mathit{Ber}_{1/2}} v.\ \exists L_1, L_2.\ V \approx (L_1, L_2) * \cdots \vdash \exists L_1, L_2.\ V \approx (L_1, L_2) * \ldots$$

Convexity in general not trivial, also challenging on paper

To ease this: Amaryllis' proof mode automates away std cases like this via typeclasses

Summary

- Amaryllis combines independence, conditioning, and dynamic allocation in one logic
- Key idea to combine separation=independence and separation=heap disjointness:
  Require only **per-outcome disjointness** of the heap
- In Amaryllis' Hoare triples $\{\Phi\}\ E\ \{V.\ \Psi(V)\}$, E and V are **random variables**
- Amaryllis' model is parametric in resource algebra, not specific to heaps

What else is in the paper?

- Generalize Iris' update modality to DPL setting
  - Novel notion: *probabilistic frame-preserving update* that preserves conditionings
  - General concept for achieving modular reasoning in DPLs
- First to apply Iris methodology to a DPL:
  - Re-interpret *authorative resource algebra* for DPL setting
  - Define *base logic* (including conditioning modality) then derive *weakest precondition modality* and *program logic* on top