New foundations for probabilistic separation logic

Northeastern University
Khoury College of Computer Sciences

John Li (li.john@northeastern.edu), Amal Ahmed (amal@ccs.neu.edu), Steven Holtzen (s.holtzen@northeastern.edu)

Probabilistic programs are getting big, and their verification at scale requires modularity 1.
Parallels between probability and mutable state suggest the use of separation logic 2.

1 Statistical independence is a fundamental modularity principle: probabilistic reasoning frequently proceeds by decomposition into independent pieces.

2 Both sampling and allocation are generative: sampling yields a random variable independent of previous ones, just as allocation yields a fresh chunk of memory.

Some separation logics support independence substructurally 3, but they overapproximate 4.

3 In these logics, propositions are sets of distributions on stores, and $\mu \vDash P * Q$ if μ factors into independent distributions μ_1 and μ_2 on stores with disjoint domain.

4 Some common independences are not expressible:

$$X = \texttt{flip } 1/2; Y = \texttt{flip } 1/2; Z = X + Y$$
$$\{(Z - X) * (Z - Y)\}$$

This postcondition is unprovable because Z appears twice.

We present a new model of separation logic where separating conjunction is interpreted by the *independent combination* of probability spaces 5. This novel combining operation is the probabilistic analogue of disjoint union of heaps 6.

5 Fix a sample space Ω. The *independent combination* of two probability spaces $(\mathcal{F}, \mu)$ and $(\mathcal{G}, \nu)$ is

$$(\mathcal{F}, \mu) \cdot (\mathcal{G}, \nu) = (\sigma(\mathcal{F}, \mathcal{G}), \rho)$$

where $\rho(F \cap G) = \mu(F)\nu(G)$ for all $F \in \mathcal{F}$, $G \in \mathcal{G}$.

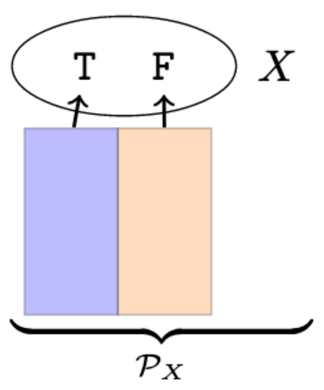
6 Independent combination forms a Kripke resource monoid, giving the expected interpretation of separating conjunction:

$(\mathcal{E}, \rho) \vDash P * Q \iff$ there exists $(\mathcal{F}, \mu)$ and $(\mathcal{G}, \nu)$ with $(\mathcal{F}, \mu) \vDash P$ and $(\mathcal{G}, \nu) \vDash Q$ and $(\mathcal{E}, \rho) = (\mathcal{F}, \mu) \cdot (\mathcal{G}, \nu)$.

Using independent combination, one can read probabilistic programs operationally: sampling literally allocates probability spaces, and conditioning performs destructive update 8.

8

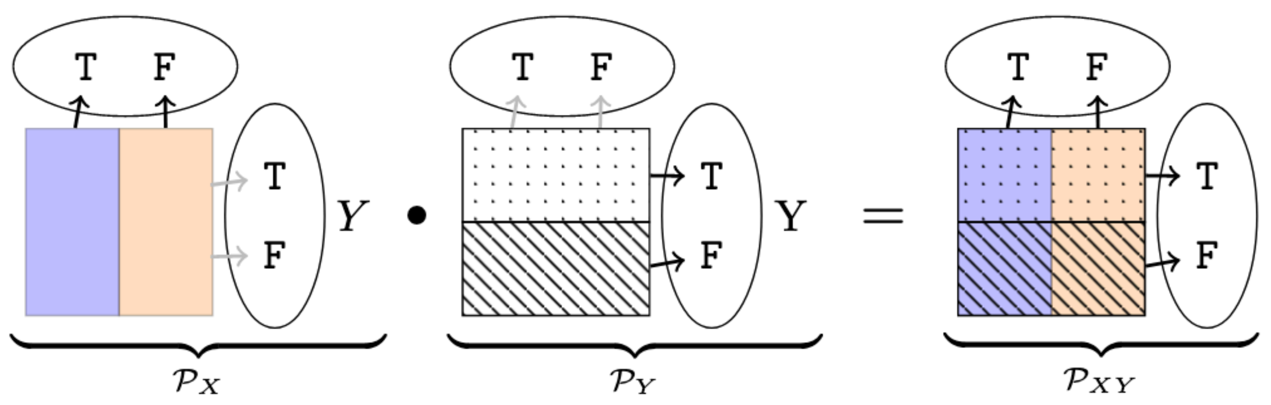
$X = \texttt{flip } 1/2;$



Shaded regions denote events, and their areas denote probabilities.

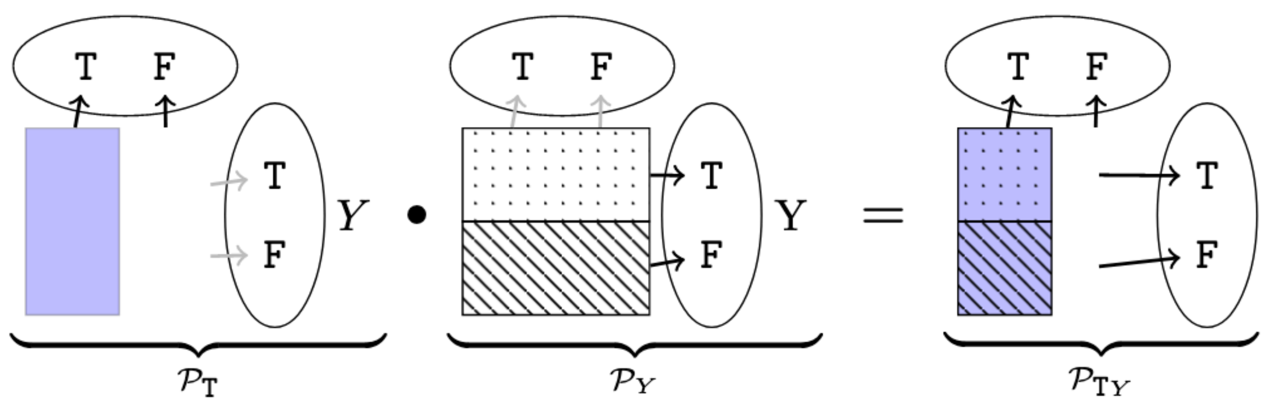
The 1st `flip` allocates a probability space $\mathcal{P}_X$ with blue-orange σ-algebra and a $\mathcal{P}_X$-measurable random variable X with distribution $\mathrm{Ber}\ 1/2$.

$Y = \texttt{flip } 1/2;$



The 2nd `flip` allocates $\mathcal{P}_Y$ with the dotted-dashed σ-algebra and a $\mathcal{P}_Y$-measurable variable Y, producing the independent combination $\mathcal{P}_X \cdot \mathcal{P}_Y$.

`observe` $X = \mathrm{T};$



Finally, `observe` destructively updates the measure stored in $\mathcal{P}_X$ so that $\mathrm{P}(X = \mathrm{T}) = 1$.

This new separation logic precisely captures independence 7, enjoys a completely standard frame rule, and easily supports continuous random variables. Using disintegration theory, we extend the base logic with a *conditioning modality* 8, and use this modal logic to verify a challenging randomized algorithm 9.

7 $(\mathcal{F}, \mu) \vDash X_1 * \ldots * X_n$ iff $X_1, \ldots, X_n$ mutually independent.

8 $\mathrm{D}_{x \leftarrow X} P$ says P holds conditional on $X = x$ for all x. Propositions have intuitive "conditional" readings under D. For example, $\mathrm{D}_{x \leftarrow X}(Y * Z)$ expresses conditional independence of Y and Z given X.

9 Sampling from a finite distribution can be done in constant space given access to continuous variables. The correctness proof uses D to condition on a continuous variable and a derived rule expressing the law of total expectation. A crucial step exploits independences automatically preserved through each loop iteration by the frame rule.

https://john-ml.github.io/lilac.pdf