Program Logics for Probabilistic Programs

Motivation: Analogy between probability and mutable state

Probability and mutable state have more in common than you think

This is ongoing work, started with a POPL 2020 paper from Bartha, Hsu, Liao, "Probabilistic Separation Logic"

$$S \models P * Q$$

$$\exists\ (s_1,h_1) \models P \quad (s_2,h_2) \models Q$$

$$(s_1,h_1) \perp (s_2,h_2)$$

└ decomp. operator

"Lilac"
John Li, Amal Ahmed
Steven Holtzen

Separation Logic:

$$\{T\}\ x := \text{alloc } 10;$$
$$\{x \mapsto 10\}$$
$$y := \text{alloc } 20;$$
$$\{x \mapsto 10 * y \mapsto 20\}$$

Separation Logic has been quite helpful for imperative heap manipulating programs.

Now, lets consider an analogous program

$$\{T\}\ x := \text{Flip } 1/2;\ (\text{x is a random following Bernoulli } 1/2)$$
$$\{x \sim \text{Bernoulli } 1/2\}$$
$$y := \text{Flip } 1/2;$$
$$\{x \sim \text{Bernoulli } 1/2 * y \sim \text{Bernoulli } 1/2\}$$

x and y are independent random variables each following bernoulli 1/2

Do we get all the same properties from separation logic?

$$\frac{\{P\}\ c\ \{Q\}}{\{P * F\}\ c\ \{Q * F\}}$$

Can extend props in smaller context to a larger context.

Composing a larger heap from a set of disjoint smaller heaps

Today, we're gonna try to get intuition about props like x tilde Bern 1/2

Separation Logic for Heap Programs

(s,h)

s: names -> heap loc

h: loc -> values (partial map)

Base cases

$$(s,h) \models T \quad \text{true works for all heaps}$$
$$(s,h) \not\models F \quad \text{false for none}$$

More

$$(s,h) \models P \wedge Q \text{ iff } (s,h) \models P \text{ and } (s,h) \models Q$$

Configuration disjointness: if heaps dont reference same locations

$$(s_1,h_1) \perp (s_2,h_2) \text{ iff } dom(h_1) \cap dom(h_2) = \emptyset$$

$$(s_1,h_1) \uplus (s_2,h_2) \quad \text{partial function} \quad \ell \mapsto \begin{cases} h_1(\ell) & \text{if } \ell \in dom(h_1) \\ h_2(\ell) & \text{if } \ell \in dom(h_2) \end{cases}$$

$$(s,h) \models P \wedge Q \text{ iff } (s,h) \models P \text{ and } (s,h) \models Q$$

$$(s,h) \models [x \mapsto v] \text{ iff } h(s(x)) = v$$

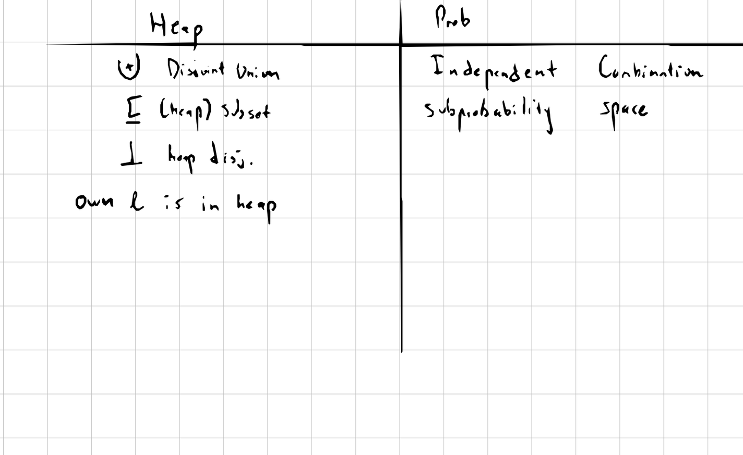
Table of Analogies

Heap	Prob	
$\uplus$ Disjoint Union	Independent	Combination
$\sqsubseteq$ (heap) subset	subprobability	space
$\perp$ heap disj.		
own ℓ is in heap		

A portion of heap is owned, if it can be referred to with a proposition

We're going to build a model that again is by analogy with heaps and stores.

$$(s, (\Omega, F, \mu)) \models T$$

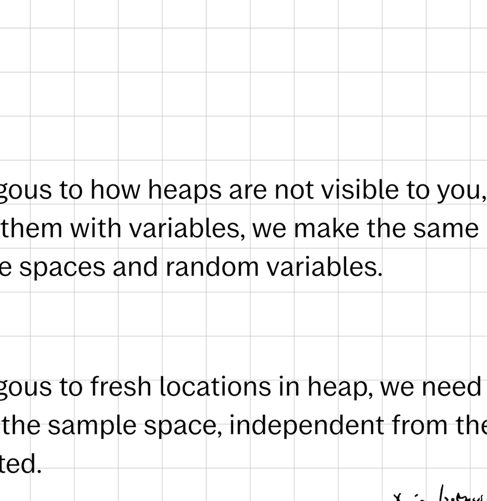


$$x : \Omega \to \{T, F\}$$

- x behaves like a pointer/location
-

Analogous to how heaps are not visible to you, and you only touch them with variables, we make the same idea here with sample spaces and random variables.

Analogous to fresh locations in heap, we need a fresh location out of the sample space, independent from the ones I've already allocated.



x is behaving like a pointer location

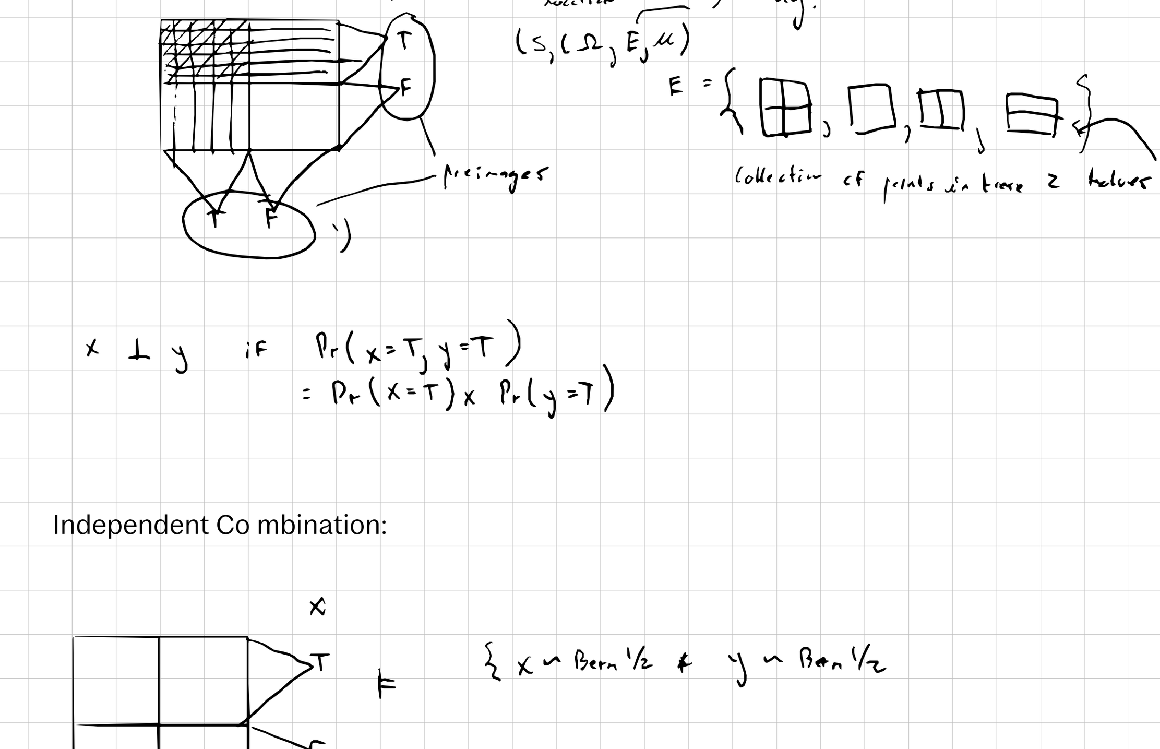
$$(s, (\Omega, E, \mu))$$

$$E = \{ \dots \}$$

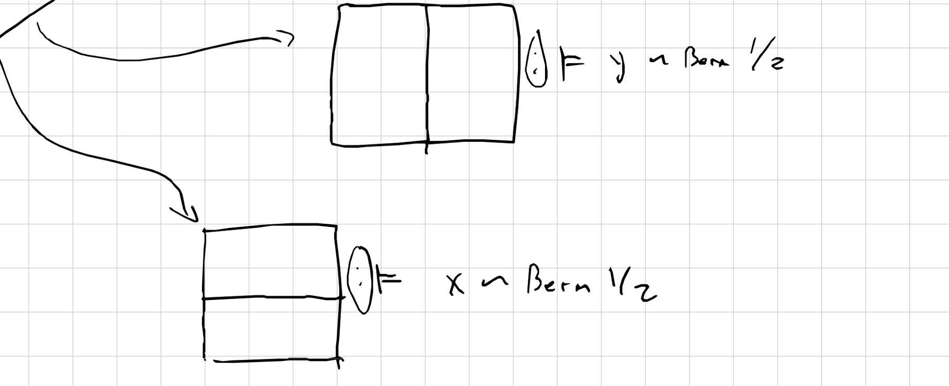
Collection of points in there 2 halves

$$x \perp y \text{ if } Pr(x=T, y=T) = Pr(x=T) \times Pr(y=T)$$

Independent Co mbination:



Disjoint Union



$$(\Omega_1, E_1, \mu_1) \bullet (\Omega, E_2, \mu_2) = (\Omega, E_1 \sqcup E_2, \rho)$$

$$\forall e_1 \in E_1,\ e_2 \in E_2 \quad \rho(e_1 \wedge e_2) = \mu_1(e_1) \times \mu_2(e_2)$$

(Partial Function)

Question: is there a nice way of choosing an appropriate omega?

- I like this one.
- It's like asking "how do I choose the kind of heap shapes for my program"
- Its a large set of possible intervals I can carve out for my new randomness
- Behaves very strangely due to real numbers, can always carve out from it.

Question: what is mu

- a function from events to 0,1

$$\mu : \text{events} \to [0, 1]$$

Question: How do I allocate another fresh source of randomness in this square:

- Have to be clever about it.
- Something perpendicular maybe.

Independent combination is a function.

Disjoint union is a partial function that takes two probability spaces and tries to combine them into one, preserving the measure factorization structure.

That gives me my interpretation for disjoint union.

Definitions are in the paper.

It is important to be able to pull back x

$$x \perp y \text{ if } Pr(x=T, y=T) = Pr(x=T) \times Pr(y=T)$$

$$S : \text{names} \to (\Omega \to \text{values})$$

Heap

$$(s, (\Omega, E, \mu)) \models x \sim \text{Bern } \theta \text{ iff}$$
$$s^{-1}(x=T) \in E$$
$$s^{-1}(x=F) \in E$$
$$\mu(s^{-1}(T)) = \theta$$

x is measurable

"Ownership is Measurability"

A nice aspect here is we've managed to take well studied portions of probability theory and encoded them in our logic.

The main point I wanna make is every single piece is in close analogy to separation logic.

Question: Is there an analagous heap metaphor for conditional independence?

- Conditioning needs to be in our logic
-

$$C_{X \leftarrow x}\ P \quad \text{"P holds conditionally on X"}$$

$$C_{z \leftarrow Z}\ (x \sim \text{Bern } 1/2, * \ y \sim \text{Bern } 1/2$$

Semantics are spicy

Lilac paper, with title "A modal separation logic for conditional probability"

It gets hairy, dont wanna say much more

Question: Does it behave like diamond?

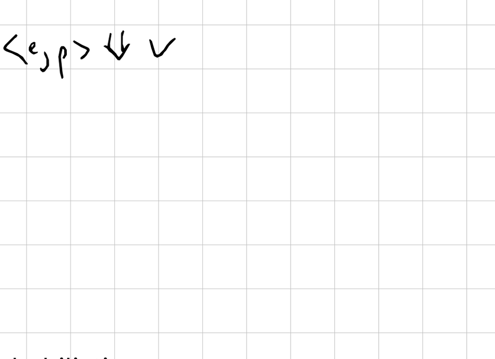
- Interesting question
- Paper, Bao Blubell, has a different way to define conditional probability
- For our purposes, we proved laboriously the specific rules we needed for our examples
- Defining different structures could be quite useful.

Credits "John" for a lot of this, don't know who that is.

Perspectives

Want to conclude with big challenges in probabilistic programming

1. Scalability
   - We have this underlying hardness thats difficult to avoid
   - Hopeful that a better deductive logic, better sampling will help
   - It's a big barrier to using them in real programs
   - Right now we're on the order of 1000 lines
   - Scalable one will require a synthesis of lots of ideas.
2. Usability
   - Probabilistic programs are not very fun to use
   - No debuggers, profilers, ide tooling
   - They'll often just output a wrong answer instead of failing
   - PPLs are the domain of the experts, which is not the point.
   - There are research questions here, I don't know how to do the debugging
   - Profilers are obvious, there's interesting research here about which characteristics we check
3. Core semantic challenges
   - I want something that takes a term, and out comes a prob, something that looks like a probability distribution (denotation function)
   - I showed you a very simple language, no loops, which does let us do this
   - If we add higher-order functions, recursion, other stuff, adds complexity.
   - Paper: Staton
   - It is hard to make denotational arguments about prob, even though we know lots about probability.
   - Open questions here.
4. Inference (debatably falls under performance)
   - Only showed a sliver of the tools and capabilities
   - Need to PL'ify this
   - Don't have good foundational principles for reasoning about this



$$\Theta \vdash \langle e, \rho \rangle \Downarrow v$$

5. Reasoning (other questions)
   - Automation, other kinds of separation
   -

Question: Can we combine heap manipulation and probabilistic separation?

- In principle its possible
- Will be hard, layers of separation
- Concurrent separation logic will have something that could help

Question: Has there been work on embedding PP in theorem provers?

- With the idea of reasoning about programs i
- Fun question, sounds like fun.
- Theres lots of work, Joe Tasarotti
-