Probabilistic Programming & Probabilistic Programming Languages

Yizhou Zhang
University of Waterloo
What is a Probabilistic Program?
What is a Probabilistic Program?

Drawing samples
What is a Probabilistic Program?

Drawing samples

Conditioning
specifies samples that are good
What is a Probabilistic Program?

- **Drawing samples**
  - Describes a distribution

- **Conditioning**
  - Specifies samples that are good

Describes a conditional distribution
What is a Probabilistic Program?

Drawing samples

Describes a distribution

Conditioning

specifies samples that are good

Describes a conditional distribution

\[ p(Z = z \mid X = x) \]
What is a Probabilistic Program?

Drawing samples

Conditioning

- Describes a distribution
- Describes a conditional distribution

$p(Z = z | X = x)$

specifies samples that are good

Latent $Z$  Observed $X$
What is a Probabilistic Program?

Drawing samples

Describes a distribution

Conditioning

specifies samples that are good

Bayes’ Theorem

\[ p(Z = z | X = x) = \frac{p(Z = z, X = x)}{p(X = x)} \]

latent observed
What is a Probabilistic Program?

**Drawing samples**
- Describes a distribution

**Conditioning**
- Specifies samples that are good

---

**Bayes’ Theorem**

$$p(z \mid x) = \frac{p(z, x)}{p(x)}$$

- **Posterior**
- **Joint**
- **Evidence**
What is a Probabilistic Program?

Drawing samples
Describes a distribution
Conditioning
specifies samples that are good

Bayes’ Theorem
\[ p(z \mid x) = \frac{p(z, x)}{p(x)} = \frac{p(x \mid z) p(z)}{p(x)} \]

Describes a conditional distribution
What is a Probabilistic Program?

**Drawing samples**
- Describes a distribution

**Conditioning**
- Specifies samples that are good

**Bayes’ Theorem**
\[
p(z | x) = \frac{p(z, x)}{p(x)} = \frac{p(x | z) p(z)}{\int p(z, x) \, dz}
\]
- Posterior
- Joint
- Likelihood
- Prior
- Evidence
- Marginal likelihood

\(z\)
\(X\)
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

\[ p(S \mid R) \]

\[ p(R) \]

\[ p(W \mid S, R) \]
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Aside on notation:

\[ p(W = T | S = T, R = F) \]

probability mass

\[ p(W | S = T, R = F) \]
distribution

\[ p(W | S, R) \]
family of distributions
Example: Rain-Sprinkler-Grass

Q1: Given that it rained, how likely is that the sprinkler was active?

\[ p(S \mid R) \]

\[ p(W \mid S, R) \]

\[ p(R) \]
Q1: Given that it rained, how likely is that the sprinkler was active?

Q2: Given that it rained, how likely is that the grass is wet?

\[
p(W | R = T) = \sum_s p(w, s | R = T) \cdot p(s | R = T)
\]

\[
= 0.99 \times 0.01 + 0.8 \times 0.99
\]

\[
= 0.8019
\]
Example: Rain-Sprinkler-Grass

Q1: Given that it rained, how likely is that the sprinkler was active?

Q2: Given that it rained, how likely is that the grass is wet?

Q3: Given that grass is wet, how likely is that it rained?
Example: Rain-Sprinkler-Grass

```javascript
var model = function() {
    // Pr(R)
    var r = sample(Bernoulli({p : 0.2}))

    // Pr(S|R=r)
    var s = sample(Bernoulli({p : r ? 0.01 : 0.4}))

    // Pr(W|R=r,S=s)
    var w = sample(Bernoulli({p :
        r ? (s ? 0.99 : 0.8) : (s ? 0.9 : 0.00)
    }))

    // condition model on W being true
    condition(w == true);

    return {R: r, S: s, W: w}
}

// apply Bayesian inference
var R_dist = Infer({
    method: 'enumerate',
    model: function() {
        var result = model()
        return result.R
    }
})
```

Q3: Given that grass is wet, how likely is that it rained?
Example: Rain-Sprinkler-Grass

```javascript
var model = function() {
  // Pr(R)
  var r = sample(Bernoulli({p : 0.2}));

  // Pr(S|R=r)
  var s = sample(Bernoulli({p : r ? 0.01 : 0.4}));

  // Pr(W|R=r,S=s)
  var w = sample(Bernoulli({p :
    r ? (s ? 0.99 : 0.8) : (s ? 0.9 : 0.0) 
  }));

  // condition model on W being true
  condition(w == true);

  return {R: r, S: s, W: w}
}

// apply Bayesian inference
var R_dist = Infer({
  method: 'enumerate',
  model: function() {
    var result = model()
    return result.R
  }
})
```

Q3: Given that grass is wet, how likely is that it rained?
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Q3: Given that grass is wet, how likely is that it rained?

Associational
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Q3: Given that grass is wet, how likely is that it rained?

associational
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Q3: Given that grass is wet, how likely is that it rained?

Q4: If we were to turn on the sprinkler, how likely would the grass be wet?

Ladder of Causation
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Q3: Given that grass is wet, how likely is that it rained?

Q4: If we were to turn on the sprinkler, how likely would the grass be wet?

Q5: Given that the sprinkler is active, had we turned off the sprinkler, how likely would the grass still be wet?

Ladder of Causation

Q3: Given that grass is wet, how likely is that it rained?

Q4: If we were to turn on the sprinkler, how likely would the grass be wet?

Q5: Given that the sprinkler is active, had we turned off the sprinkler, how likely would the grass still be wet?

Associational

Interventional

Counterfactual
Example: Rain-Sprinkler-Grass

https://en.wikipedia.org/wiki/Bayesian_network#Example

Q3: Given that grass is wet, how likely is that it rained?

Q4: If we were to turn on the sprinkler, how likely would the grass be wet?

Q5: Given that the sprinkler is active, had we turned off the sprinkler, how likely would the grass still be wet?

$p(S | R)$

$p(W | S, R)$

$p(R)$
$p(z \mid x) = \frac{p(x \mid z) \cdot p(z)}{p(x)}$
Example: Bouncing Balls

Ball rarely falls into bucket
Example: Bouncing Balls into Bucket

https://bit.ly/2Q8s88r

Ball often falls into bucket

Bumper positions
Applications of Probabilistic Programming

\[ p(z \mid x) = \frac{p(x \mid z) p(z)}{p(x)} \]

- Program source code
- Agent’s policy/plan
- Captcha letters

- Input-output examples
- Agent’s reward
- Captcha images
Applications of Probabilistic Programming

\[
p(z | x) = \frac{p(x | z) p(z)}{p(x)}
\]
Applications of Probabilistic Programming

\[ p(z \mid x) = \frac{p(x \mid z) p(z)}{p(x)} \]
Applications of Probabilistic Programming

\[ p(z \mid x) = \frac{p(x \mid z) p(z)}{p(x)} \]

- Input-output examples
- Agent’s reward
- Captcha images
- Program source code
- Agent’s policy/plan
- Captcha letters
PL as an Abstraction Layer

Programs

Interpreter / Compiler
PL as an Abstraction Layer

Programs

expression

Interpreter / Compiler

PL

solving
PPL as an Abstraction Layer

Probabilistic Programs

Probabilistic Inference

expression

PPL

solving
PPL as an Abstraction Layer

Probabilistic Programs

Probabilistic Inference

PPL

expression

solving

latent RVs

observed RVs
PPL as an Abstraction Layer

Probabilistic Programs

- Probabilistic Inference

- data-generation process
- generative model
- stochastic simulation
- decoders
- inductive bias

- latent RVs $Z$
- observed RVs $X$

- expression
- solving

PPL
PPL as an Abstraction Layer

Probabilistic Programs

\[ p(z \mid x) = \frac{p(x \mid z) p(z)}{p(x)} \]

Probabilistic Inference

Interpreters & Compilers

- data-generation process
- generative model
- stochastic simulation
- decoders
- inductive bias

- latent RVs
- observed RVs

PPL

expression

solving
What is hard about Bayesian inference?

**Bayes’ Theorem**

\[
p(z \mid x) = \frac{p(z, x)}{p(x)} = \frac{p(x \mid z)p(z)}{\int p(z, x) \, dz}
\]

- **Posterior**: \[p(z \mid x)\]
- **Joint**: \[p(z, x)\]
- **Likelihood**: \[p(x \mid z)\]
- **Prior**: \[p(z)\]
- **Evidence**: \[\int p(z, x) \, dz\]
- **Marginal Likelihood**: \[\int p(z, x) \, dz\]

Integration of joint over all execution traces

Enumerating all traces is unrealistic
What is hard about Bayesian inference?

**Bayes’ Theorem**

\[ p(z \mid x) = \frac{p(z, x)}{p(x)} = \frac{p(x \mid z) p(z)}{\int p(z, x) \, dz} \]

- **Posterior** \( p(z \mid x) \)
- **Joint** \( p(z, x) \)
- **Likelihood** \( p(x \mid z) \)
- **Prior** \( p(z) \)
- **Evidence** \( p(x) \)
- **Marginal Likelihood** \( \int p(z, x) \, dz \)

Integration of joint over all execution traces

Joint is defined by a program

Enumerating all traces is unrealistic

Integration rarely has analytical solutions
Bayes’ Theorem

\[ p(z \mid x) = \frac{p(z, x)}{p(x)} = \frac{p(x \mid z) p(z)}{\int p(z, x) \, dz} \]

- **Posterior**
- **Evidence**
- **Marginal Likelihood**

Integration of joint over all execution traces

- Joint is defined by a program
- Enumerating all traces is unrealistic
- Integration rarely has analytical solutions
Have to Approximate or Limit Expressivity

Approximate

Rejection Sampling
Likelihood Weighting
Importance Sampling
Sequential Monte Carlo (SMC)
Markov Chain Monte Carlo (MCMC)
Variational Inference
Have to Approximate or Limit Expressivity

Approximate
- Rejection Sampling
- Likelihood Weighting
- Importance Sampling
- Sequential Monte Carlo (SMC)
- Markov Chain Monte Carlo (MCMC)
- Variational Inference

Limit Expressivity
- Reduced expressive power
- Improved run-time efficiency
## Have to Approximate or Limit Expressivity

### Approximate
- Rejection Sampling
- Likelihood Weighting
- Importance Sampling
- Sequential Monte Carlo (SMC)
- Markov Chain Monte Carlo (MCMC)
- Variational Inference

### Limit Expressivity
- Reduced expressive power
- Improved run-time efficiency
- Most common restriction to impose:
  - Ban recursion/unbounded loops
    (think of finite graphical models)
- Examples: Stan, Infer.NET, …
PPL as an Abstraction Layer

Probabilistic Programs

Probabilistic Inference

expression

PPL

solving
Example: Reinforcement Learning

agent trajectory $Z$

high reward $X$
Example: Reinforcement Learning

**Environment**
- `def reward(state) // immediate reward`
- `def transition(state, action) // step the environment`

**Agent**
- `def MDP(state): // recursive MDP description`

Diagram:
- States: $s_1, s_2, s_3$
- Actions: $a_1, a_2, a_3$
- Reward
- High reward
- Agent trajectory
**Example: Reinforcement Learning**

**Environment**

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

**Agent**

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    MDP(nextState) // recurse
```
Example: Reinforcement Learning

Goal of inference:
a policy function \( \pi : \text{State} \rightarrow \text{Action} \)

Environment

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

Agent

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    MDP(nextState) // recurse
```
Example: Reinforcement Learning

Goal of inference:
a policy function $\pi : \text{State} \rightarrow \text{Action}$

Environment

```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

Agent

```
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    // condition on optimality
    MDP(nextState) // recurse
```
Example: Reinforcement Learning

Goal of inference:
a policy function $\pi : \text{State} \rightarrow \text{Action}$

Environment

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

Agent

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    // condition on optimality
    MDP(nextState) // recurse
```

$p(O_t = 1 | s_t) \overset{\text{def}}{=} \exp(r(s_t))$
Example: Reinforcement Learning

Goal of inference:
a policy function $\pi : \text{State} \rightarrow \text{Action}$

Environment

\[
\text{def } \text{reward(state)} \text{ // immediate reward}
\]
\[
\text{def } \text{transition(state, action)} \text{ // step the environment}
\]

Agent

\[
\text{def } \text{MDP(state)} : \text{ // recursive MDP description}
\]
\[
\text{if } (\text{terminal(state)})
\]
\[
\text{return}
\]
\[
\text{action} = \text{sample}(...) \text{ // sample action from prior}
\]
\[
\text{nextState} = \text{transition(state, action)}
\]
\[
\text{factor}(\text{reward(nextState)}) \text{ // condition on optimality}
\]
\[
\text{MDP(nextState)} \text{ // recurse}
\]

\[
p(O_t = 1 | s_t)^{\text{def}} = \exp(r(s_t))
\]

agent trajectory

optimality

variables

states

actions

optimality

Goal of inference:

$\pi$:

$\pi(s) = \arg \max_{a} Q(s,a)$

$p(O_t = 1 | s_t)^{\text{def}} = \exp(r(s_t))$
Example: Reinforcement Learning

Goal of inference:
a policy function $\pi : \text{State} \rightarrow \text{Action}$ leading to optimal trajectory, rather than a trajectory per se

Environment

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

Agent

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    factor(reward(nextState)) // condition on optimality
    MDP(nextState) // recurse
```

$p(O_t = 1 | s_t) \overset{\text{def}}{=} \exp(r(s_t))$
Example: Reinforcement Learning

Goal of inference:
a policy function $\pi : \text{State} \rightarrow \text{Dist}[\text{Action}]$ leading to optimal trajectory, rather than a trajectory per se

Environment

```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

Agent

```
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    factor(reward(nextState)) // condition on optimality
    MDP(nextState) // recurse
```
Example: Reinforcement Learning

Goal of inference: A policy function \( \pi(s) = q(a | s; \phi) \) leading to optimal trajectory, rather than a trajectory per se.

**Environment**

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

**Agent**

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(...) // sample action from prior
    nextState = transition(state, action)
    factor(reward(nextState)) // condition on optimality
    MDP(nextState) // recurse
```

\[ p(O_t = 1 | s_t) \overset{\text{def}}{=} \exp(r(s_t)) \]
Example: Reinforcement Learning

**Goal of inference:** \( \pi(s) = q(a \mid s; \phi) \)
a policy function \( \pi : \text{State} \rightarrow \text{Dist}[\text{Action}] \) leading to
optimal trajectory, rather than a trajectory per se

**Environment**

```python
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

**Agent**

```python
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    p(a_t) ← action = sample(...) // sample action from prior
    p(s_{t+1} \mid s_t, a_t) ← nextState = transition(state, action)
    p(O_t = 1 \mid s_t) \overset{def}{=} \exp(r(s_t)) ← factor(reward(nextState)) // condition on optimality
    MDP(nextState) // recurse
```
Example: Reinforcement Learning

**Goal of inference:** \( \pi(s) = q(a \mid s; \phi) \)

A policy function \( \pi : \text{State} \rightarrow \text{Dist}[\text{Action}] \) leading to optimal trajectory, rather than a trajectory per se.

\[
p(s_1, a_1, \ldots \mid \text{optimality}) \propto p(s_1) \prod_t p(a_t) p(s_{t+1} \mid s_t, a_t) p(O_t = 1 \mid s_t)
\]

### Agent

```python
def MDP(state):  # recursive MDP description
    if (terminal(state))
        return
    p(a_t) ← action = sample(...)  # sample action from prior
    p(s_{t+1} \mid s_t, a_t) ← nextState = transition(state, action)
    p(O_t = 1 \mid s_t) \overset{\text{def}}{=} \exp(r(s_t)) ← factor(reward(nextState))  # condition on optimality
    MDP(nextState)  # recurse
```
Example: Reinforcement Learning

Goal of inference: $\pi(s) = q(a \mid s; \phi)$
a policy function $\pi : \text{State} \to \text{Dist}[\text{Action}]$ leading to optimal trajectory, rather than a trajectory per se

$$p(s_1, a_1, \ldots \mid \text{optimality}) \propto p(s_1) \prod_t p(a_t) p(s_{t+1} \mid s_t, a_t) p(O_t = 1 \mid s_t)$$

$$q(s_1, a_1, \ldots, s_t, a_t; \phi) = p(s_1) \prod_t q(a_t \mid s_t; \phi) p(s_{t+1} \mid s_t, a_t)$$

Agent

```
def MDP(state):  // recursive MDP description
    if (terminal(state))
        return
    action = sample(...)  // sample action from prior
    nextState = transition(state, action)
    p(a_t) ← action
    p(s_{t+1} \mid s_t, a_t) ← nextState = transition(state, action)
    p(O_t = 1 \mid s_t) ≜ \exp(r(s_t)) ← factor(reward(nextState))  // condition on optimality
    MDP(nextState)  // recurse
```
Example: Reinforcement Learning

**Goal of inference:**

\[ \pi(s) = q(a \mid s; \phi) \]

A policy function \( \pi : \text{State} \rightarrow \text{Dist}[\text{Action}] \) leading to an optimal trajectory, rather than a trajectory per se.

\[
p(s_1, a_1, \ldots \mid \text{optimality}) \propto p(s_1) \prod_t p(a_t) p(s_{t+1} \mid s_t, a_t) p(O_t = 1 \mid s_t)
\]

\[
q(s_1, a_1, \ldots, s_t, a_t; \phi) = p(s_1) \prod_t q(a_t \mid s_t; \phi) p(s_{t+1} \mid s_t, a_t)
\]

\[
\min_{\phi} D_{KL} \left( q(s_1, a_1, \ldots, s_t, a_t; \phi) \mid \mid p(s_1, a_1, \ldots \mid \text{optimality}) \right)
\]
Example: Two-Player Game

https://agentmodels.org/chapters/7-multi-agent.html
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};
```
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}
```
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}

var simulate = function(state, action, player) {
    var nextState = transition(state, action, player);
    if (isTerminal(nextState)) {
        return nextState;
    } else {
        var nextPlayer = otherPlayer(player);
        return sample(Infer({ model() {
            var nextMove = act(nextState, nextPlayer);
            return simulate(nextState, nextMove nextPlayer);
        } });)
    }
};
```
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}

var simulate = function(state, action, player) {
    var nextState = transition(state, action, player);
    if (isTerminal(nextState)) {
        return nextState;
    } else {
        var nextPlayer = otherPlayer(player);
        return sample Infer(model) {
            var nextMove = act(nextState, nextPlayer);
            return simulate(nextState, nextMove nextPlayer);
        });
    }
};
```

https://agentmodels.org/chapters/7-multi-agent.html
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}

var simulate = function(state, action, player) {
    var nextState = transition(state, action, player);
    if (isTerminal(nextState)) {
        return nextState;
    } else {
        var nextPlayer = otherPlayer(player);
        return sample Infer { model() {
            var nextMove = act(nextState, nextPlayer); // (Green circle)
            return simulate(nextState, nextMove, nextPlayer);
        }};
    }
};
```

Inference inside inference => Thinking about thinking
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}

var simulate = function(state, action, player) {
    var nextState = transition(state, action, player);
    if (isTerminal(nextState)) {
        return nextState;
    } else {
        var nextPlayer = otherPlayer(player);
        return sample(Infer({ model() {
            var nextMove = act(nextState, nextPlayer);
            return simulate(nextState, nextMove nextPlayer);
        }}));
    }
};
```
Example: Two-Player Game

```javascript
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

var utility = function(state, move, player) {
    var outcome = simulate(state, move, player);
    if (hasWon(state, player)) { return 10; }
    else if (isDraw(state)) { return 0; }
    else { return -10; }
}

var simulate = function(state, action, player) {
    var nextState = transition(state, action, player);
    if (isTerminal(nextState)) {
        return nextState;
    } else {
        var nextPlayer = otherPlayer(player);
        return sample(Infer({ model() {
            var nextMove = act(nextState, nextPlayer);
            return simulate(nextState, nextMove,nextPlayer);
        } })));
    }
};
```

https://agentmodels.org/chapters/7-multi-agent.html
A Glimpse into Formal Semantics

terms \( t ::= x \) variable
| \( \lambda x. t \) abstraction
| \( t_1 \ t_2 \) application
| \( r \) real number
| \( \text{op}_n(t_1, \ldots, t_n) \) \( n \)-ary operation invocation
| \text{sample} sampling
| \text{factor} t conditioning

\[
\rho_n(\ell, t, V) \overset{\text{def}}{=} \begin{cases} r \quad \langle \ell \mid t \rangle \xrightarrow{n} \langle \epsilon \mid v \rangle \cdot r \text{ and } v \in V \\ 0 \quad \text{otherwise} \end{cases}
\]

\[
\mu_n(t, V) \overset{\text{def}}{=} \int \rho_n(\ell, t, V) \, d\ell
\]

\[
\mu(t, V) \overset{\text{def}}{=} \lim_{n \to \infty} \mu_n(t, V)
\]
Takeaway messages

PPLs are powerful tools for probabilistic modeling and inference

Exciting area of ongoing research