CS 152 Programming Languages

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

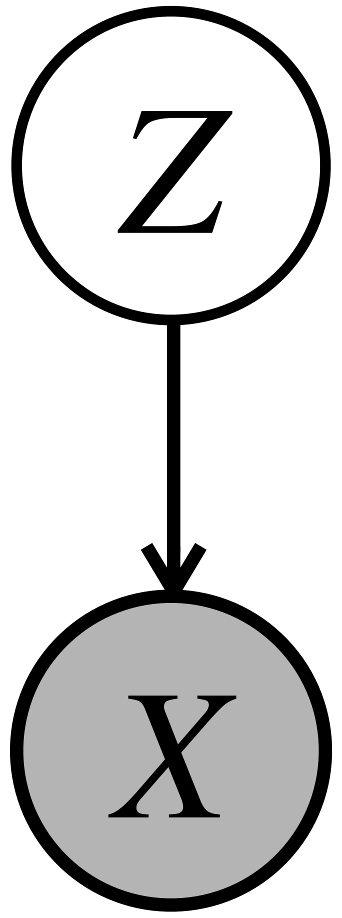
What is a Probabilistic Program?

Drawing samples → Describes a distribution

Conditioning

specifies samples that are good

→ Describes a conditional distribution



What is a Probabilistic Program?

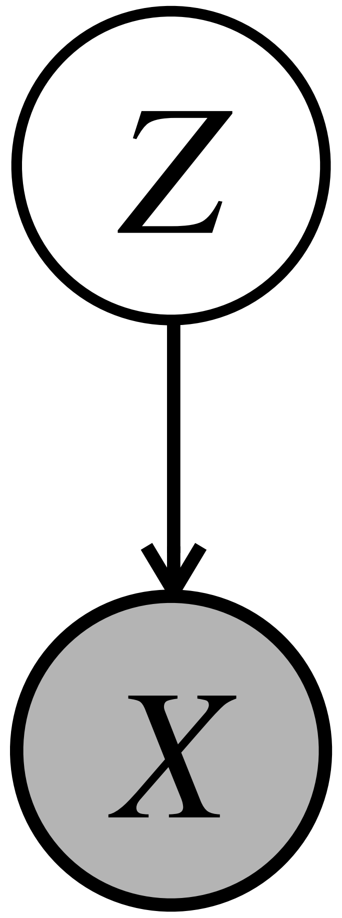
Drawing samples → Describes a distribution

Conditioning

specifies samples that are good

→ Describes a conditional distribution

$$p(Z = z \mid X = x)$$



What is a Probabilistic Program?

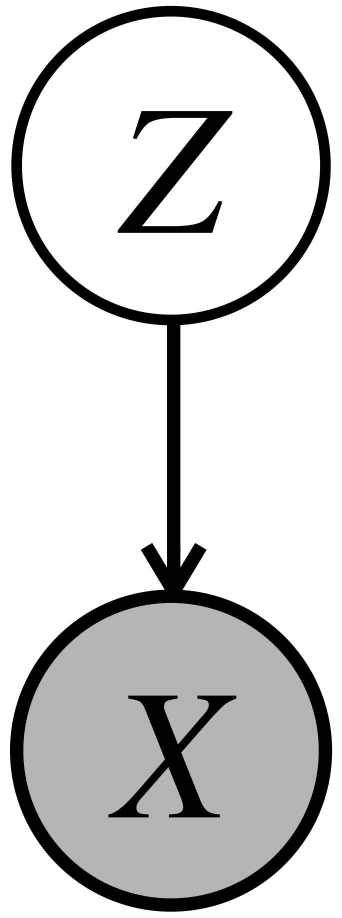
Drawing samples ⇨ Describes a distribution

Conditioning

specifies samples that are good

⇨ Describes a conditional distribution

$$p(\underbrace{Z}_{\text{latent}} = z \mid \underbrace{X}_{\text{observed}} = x)$$



What is a Probabilistic Program?

Drawing samples ⇨ Describes a distribution

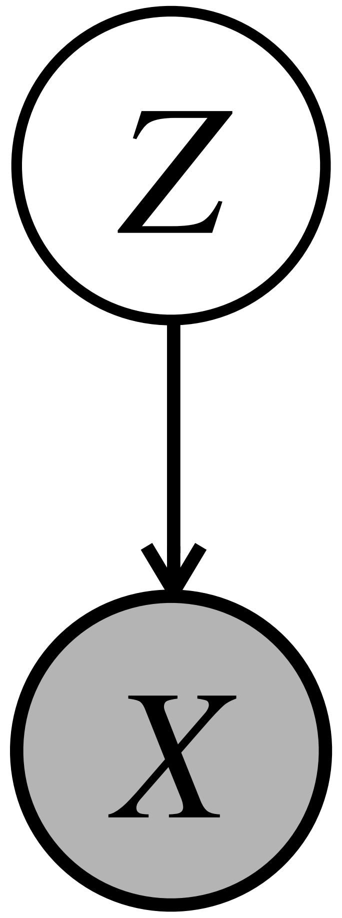
Conditioning

specifies samples that are good

⇨ Describes a conditional distribution

Bayes' Theorem

$$p(\underbrace{Z}_{\text{latent}} = z \mid \underbrace{X}_{\text{observed}} = x) = \frac{p(Z = z, X = x)}{p(X = x)}$$



What is a Probabilistic Program?

Drawing samples ⇨ Describes a distribution

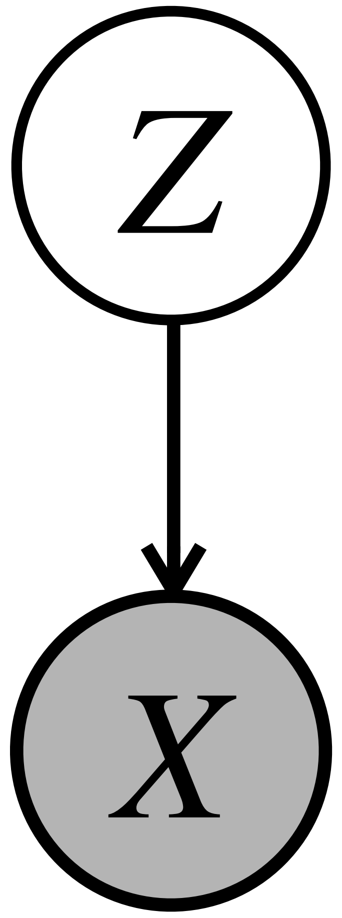
Conditioning

specifies samples that are good

⇨ Describes a conditional distribution

Bayes' Theorem

$$\underbrace{p(z \mid x)}_{\text{posterior}} = \frac{\overbrace{p(z, x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}}$$



What is a Probabilistic Program?

Drawing samples → Describes a distribution

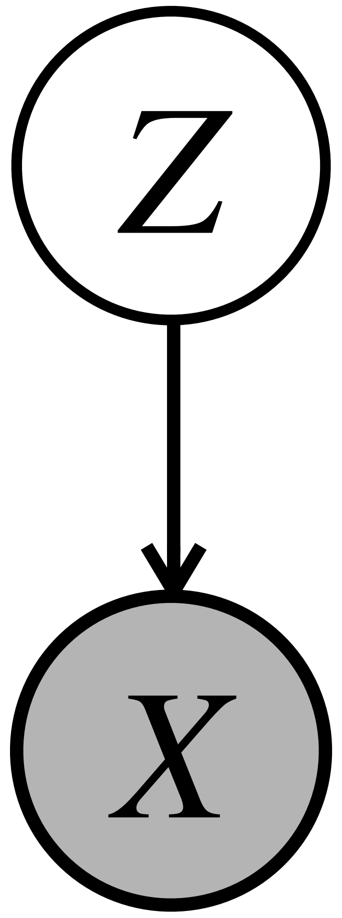
Conditioning

specifies samples that are good

→ Describes a conditional distribution

Bayes' Theorem

$$\underbrace{p(z \mid x)}_{\text{posterior}} = \frac{\overbrace{p(z, x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}} = \frac{\overbrace{p(x \mid z)}^{\text{likelihood}} \overbrace{p(z)}^{\text{prior}}}{p(x)}$$



What is a Probabilistic Program?

Drawing samples

Describes a distribution

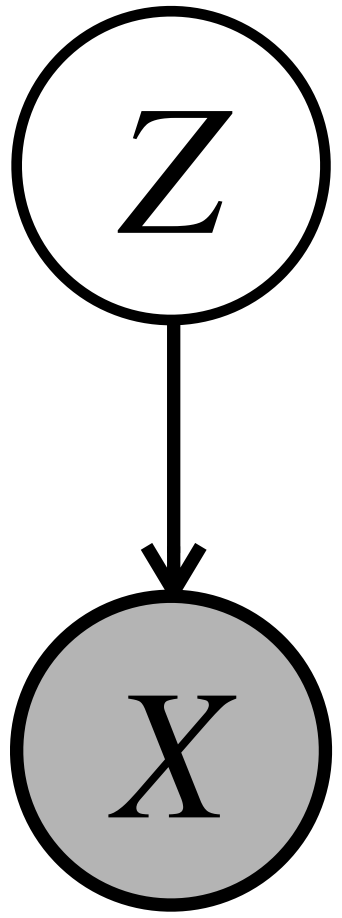
Conditioning

specifies samples that are good

Describes a conditional distribution

Bayes' Theorem

$$\underbrace{p(z \mid x)}_{\text{posterior}} = \frac{\overbrace{p(z, x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}} = \frac{\overbrace{p(x \mid z)}^{\text{likelihood}}\,\overbrace{p(z)}^{\text{prior}}}{\underbrace{\int p(z, x)\,\mathrm{d}z}_{\text{marginal likelihood}}}$$

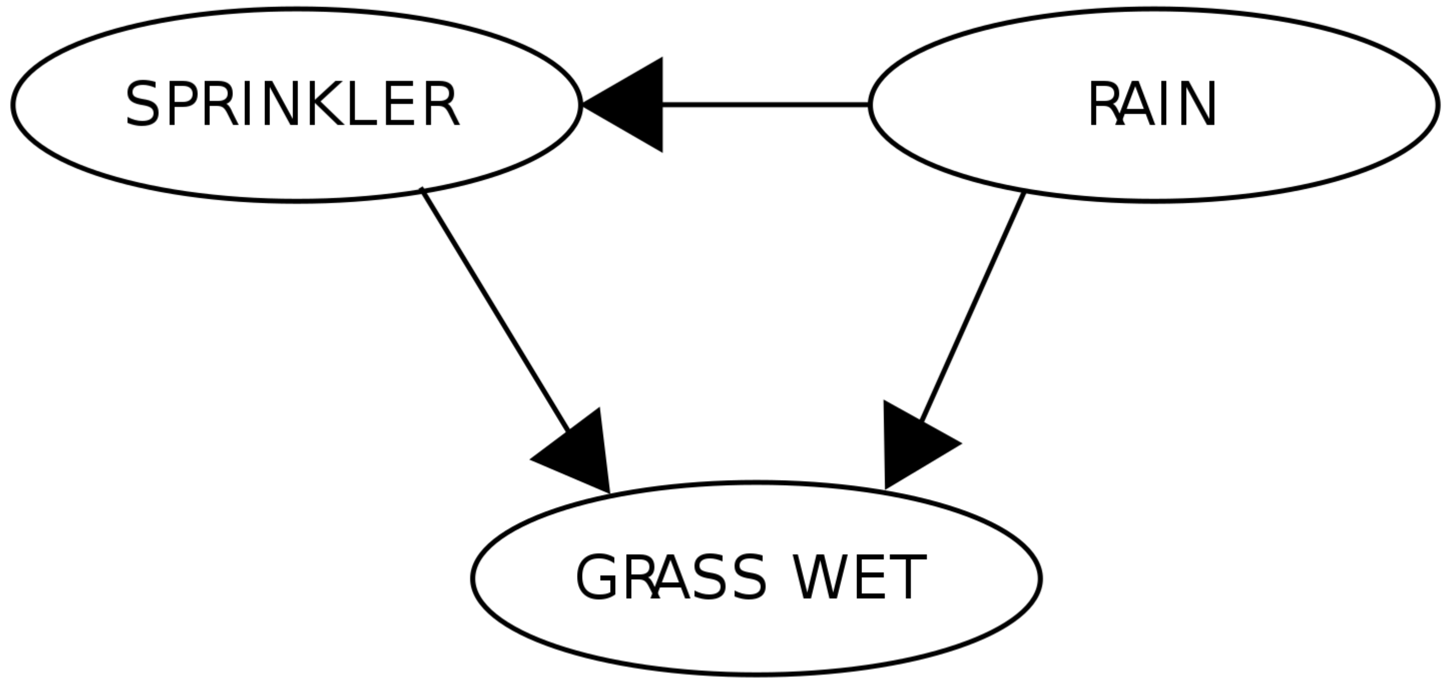


Example: Rain-Sprinkler-Grass

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

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

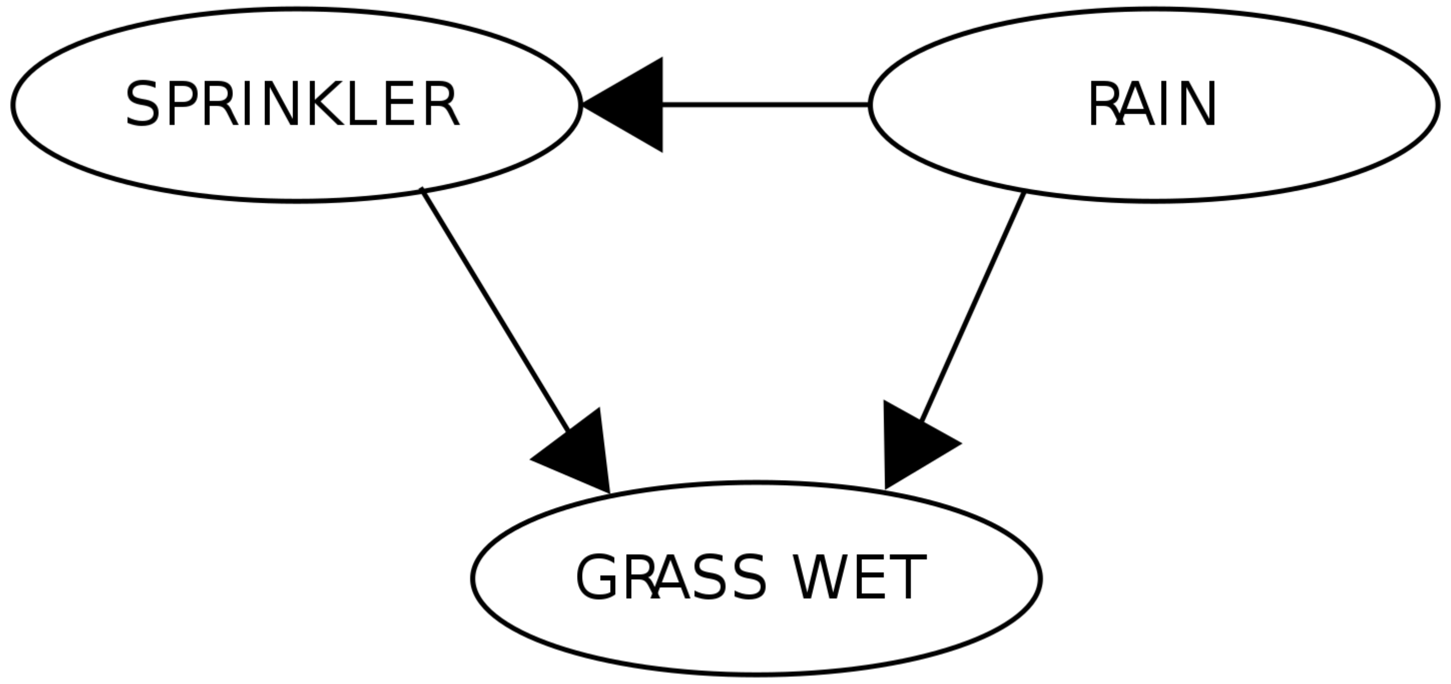
$p(W \mid S, R)$

Example: Rain-Sprinkler-Grass

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

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99

$p(S\,|\,R)$



RAIN T	RAIN F
0.2	0.8

$p(R)$

$p(W\,|\,S, R)$

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Aside on notation:

$p(W = \mathsf{T}\,|\,S = \mathsf{T}, R = \mathsf{F})$

probability mass

$p(W\,|\,S = \mathsf{T}, R = \mathsf{F})$

distribution

$p(W\,|\,S, R)$

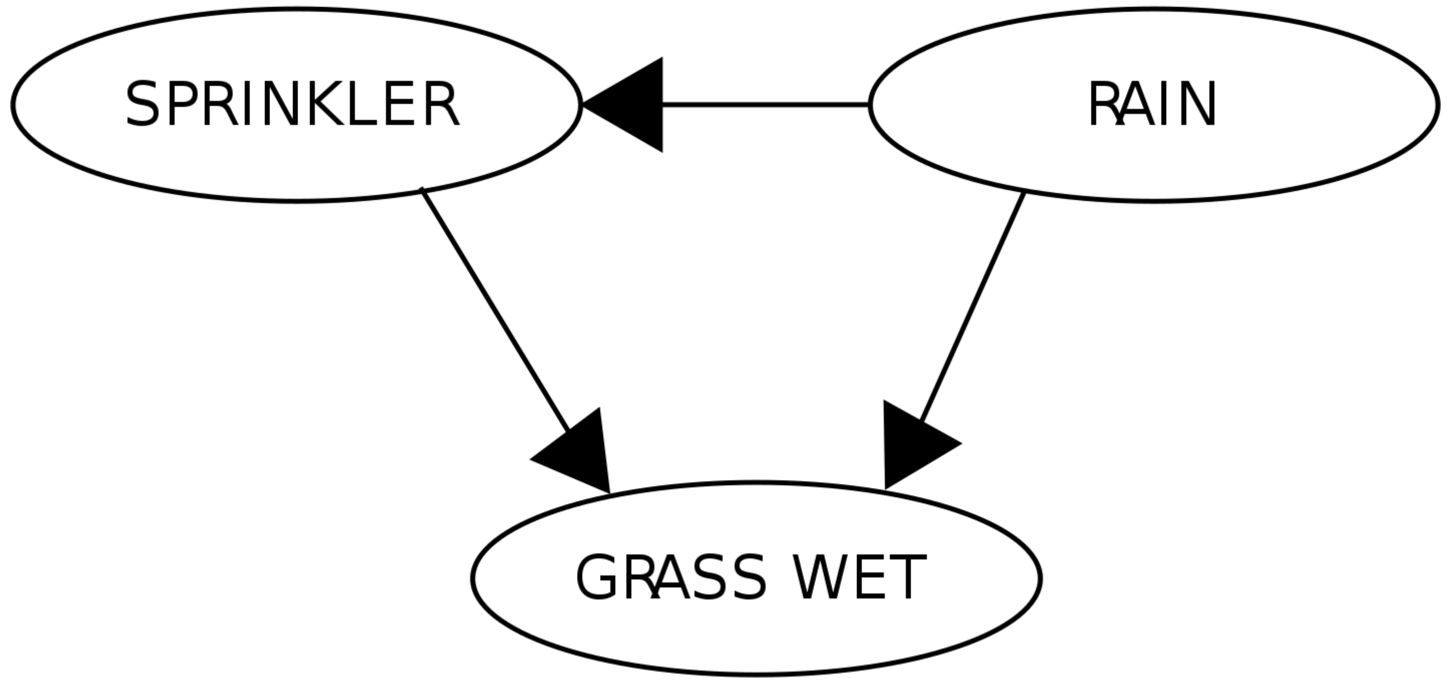
family of distributions

Example: Rain-Sprinkler-Grass

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

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$p(W \mid S, R)$

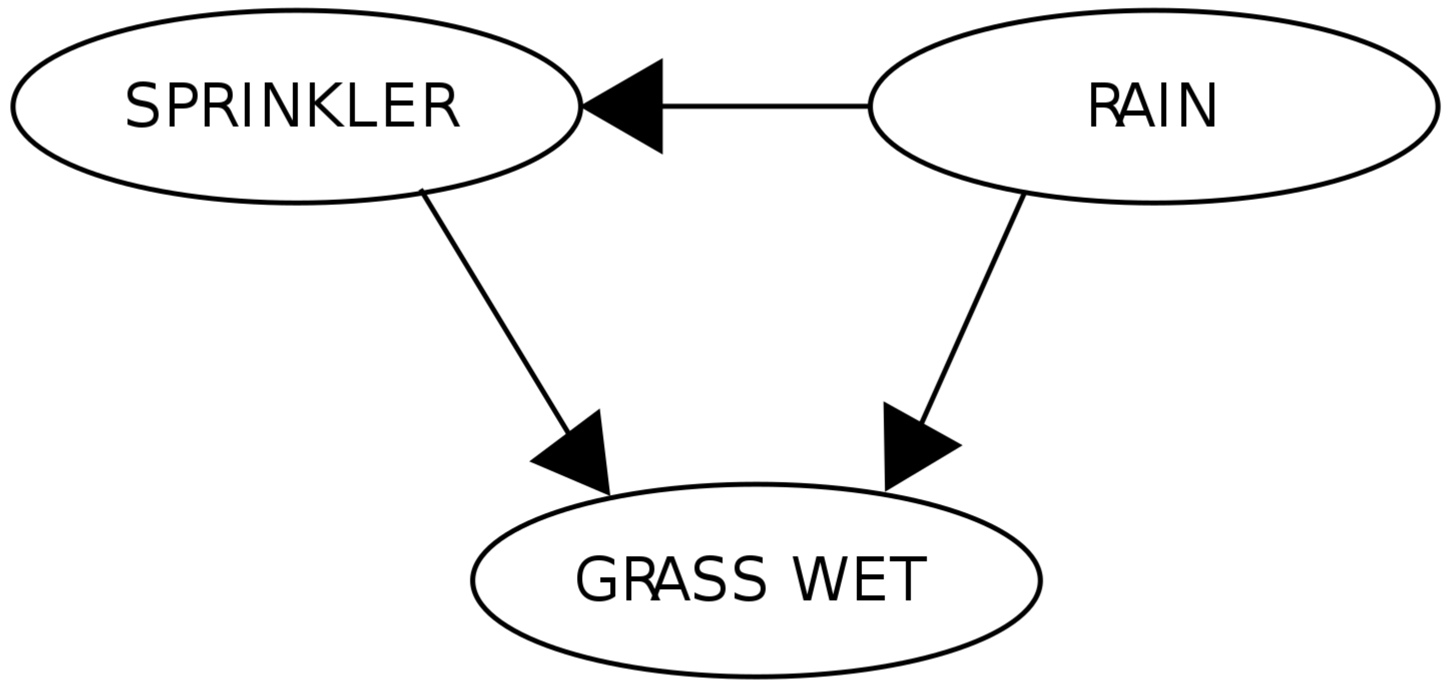
Q1: Given that it rained, how likely is that the sprinkler was active?

Example: Rain-Sprinkler-Grass

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

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$p(W \mid S, R)$

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?

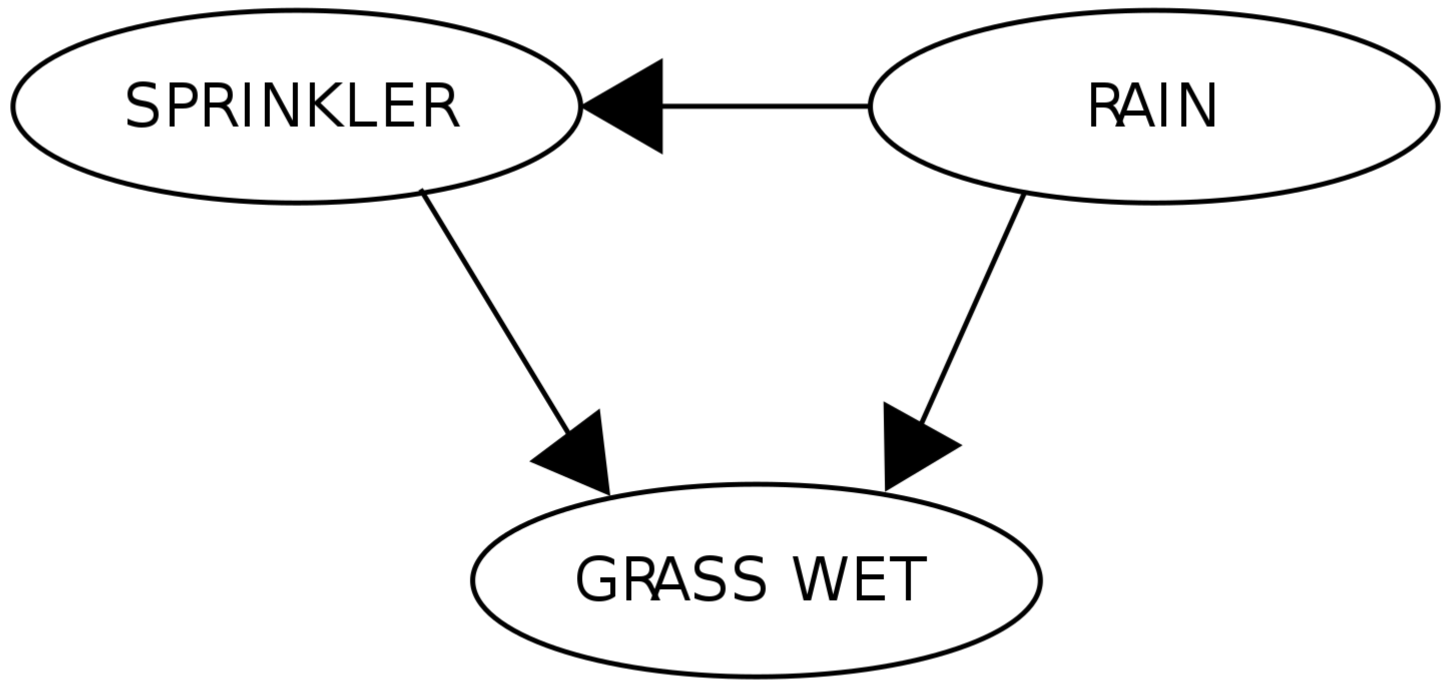
$$p(w \mid R=T)$$
$$= \sum_s p(w, s \mid R=T)$$
$$= \sum_s p(w \mid S=s, R=T)\, p(s \mid R=T)$$
$$= 0.99 \times 0.01 + 0.8 \times 0.99$$
$$= 0.8019$$

Example: Rain-Sprinkler-Grass

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

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$p(W \mid S, R)$

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?

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

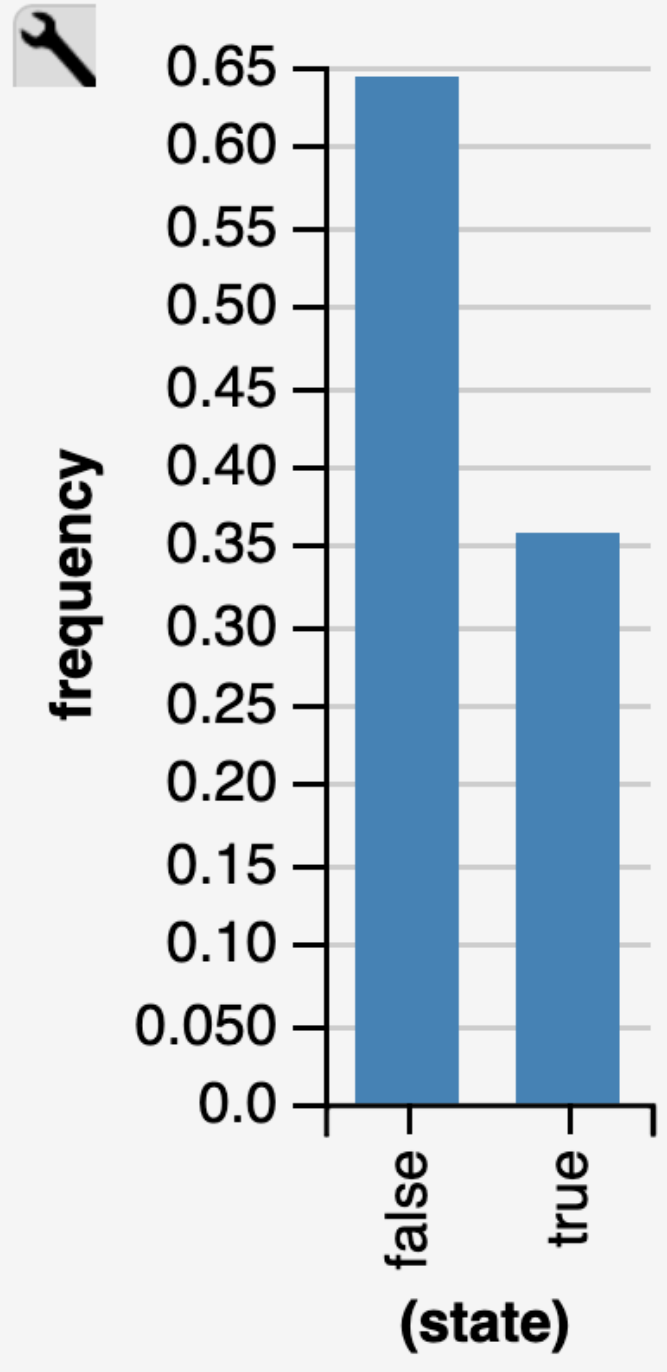
Example: Rain-Sprinkler-Grass

```
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

```
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
  }
})
```




```
Chance of rain is 35.76876756322762%
```

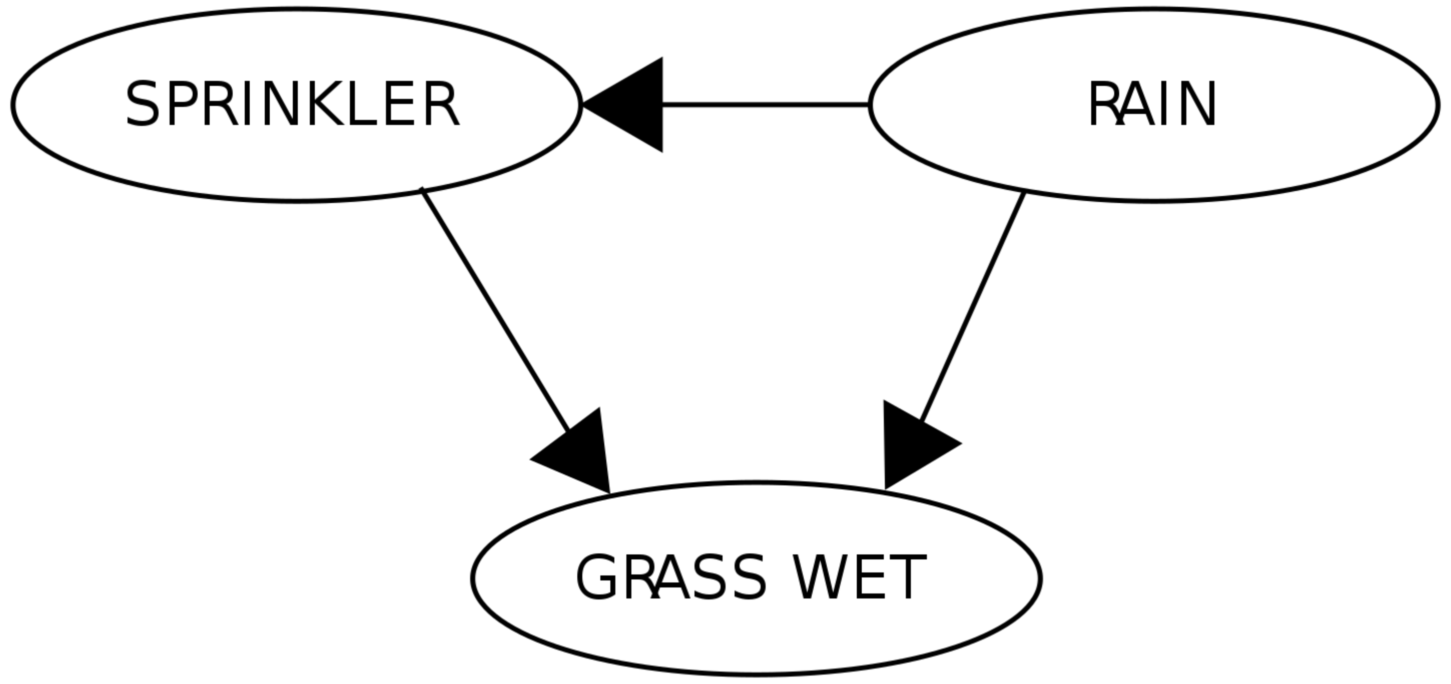
Q3: Given that grass is wet, how likely is that it rained?

Example: Rain-Sprinkler-Grass

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

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN T	RAIN F
0.2	0.8

$p(R)$

$p(W \mid S, R)$

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

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

associational

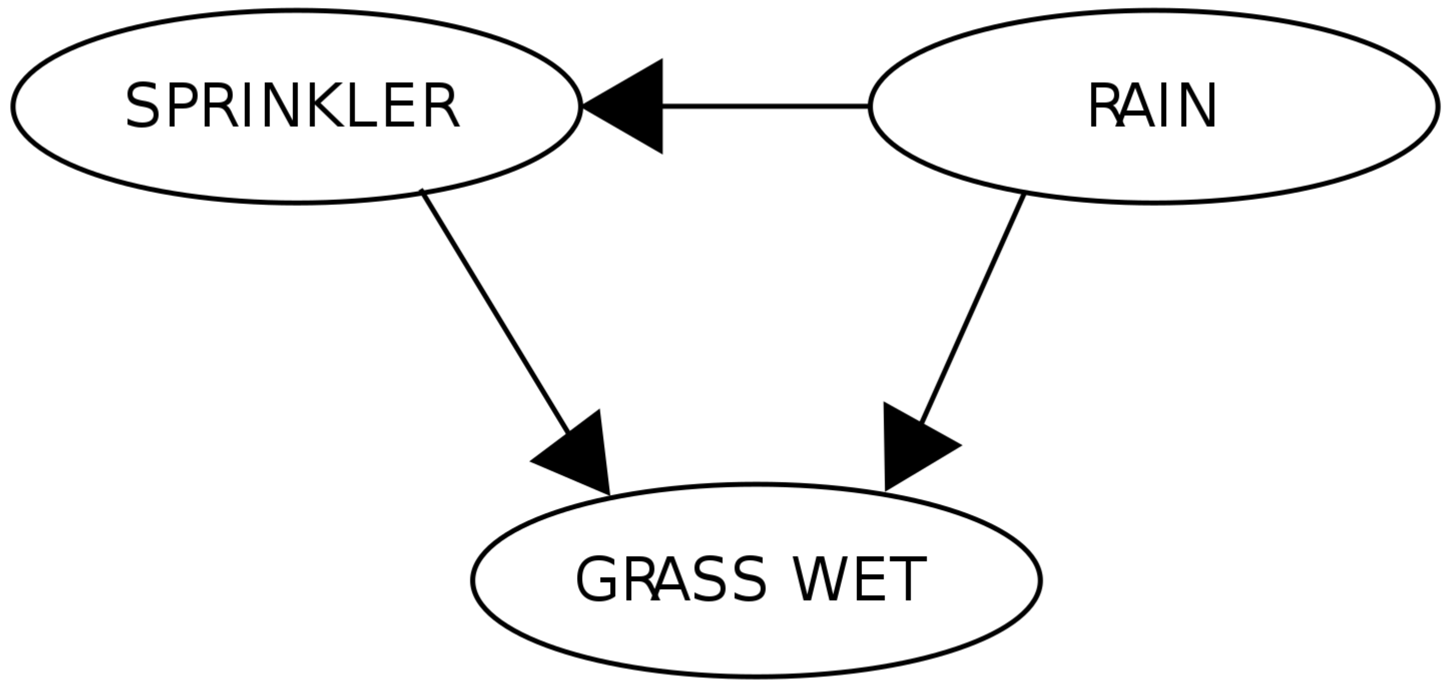
Example: Rain-Sprinkler-Grass

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

Ladder of Causation

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

$p(W \mid S, R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

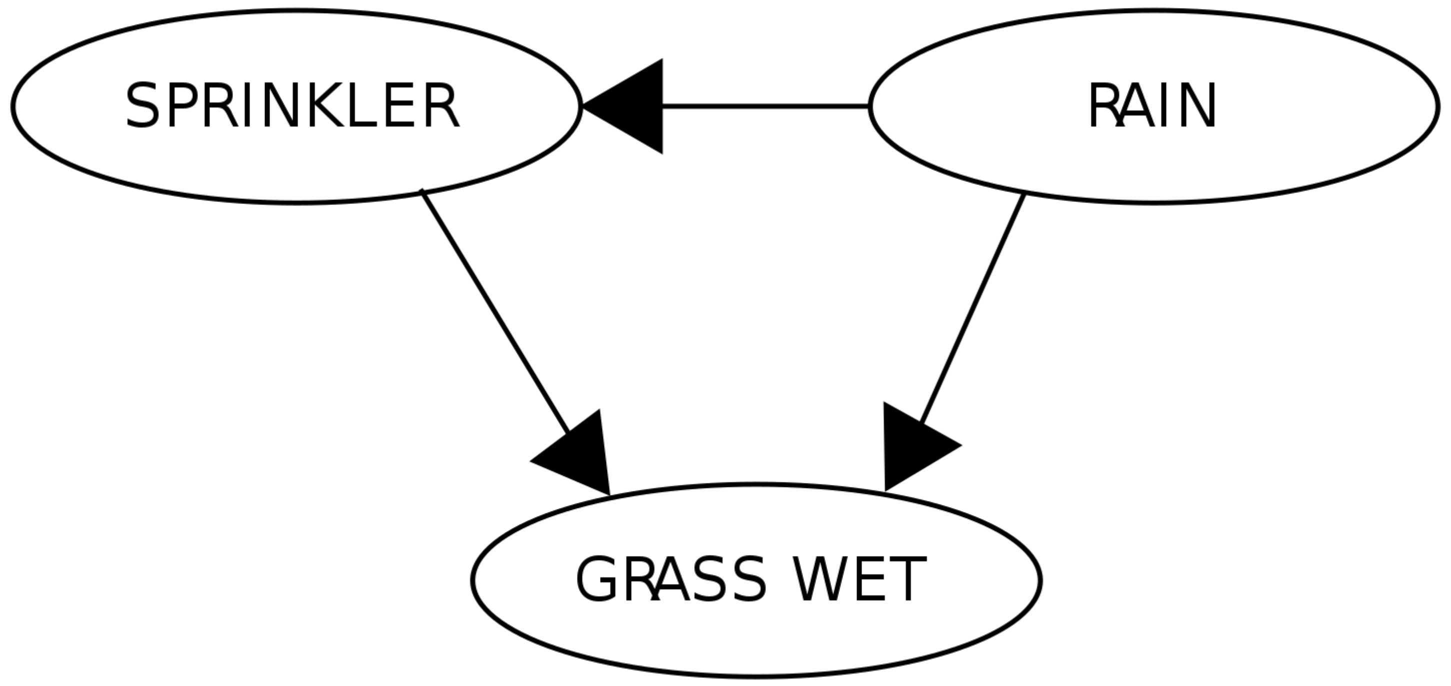
Q3: Given that grass is wet, how likely is that it rained?
associational

Example: Rain-Sprinkler-Grass

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

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN T	RAIN F
0.2	0.8

$p(R)$

$p(W \mid S, R)$

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

Ladder of Causation

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

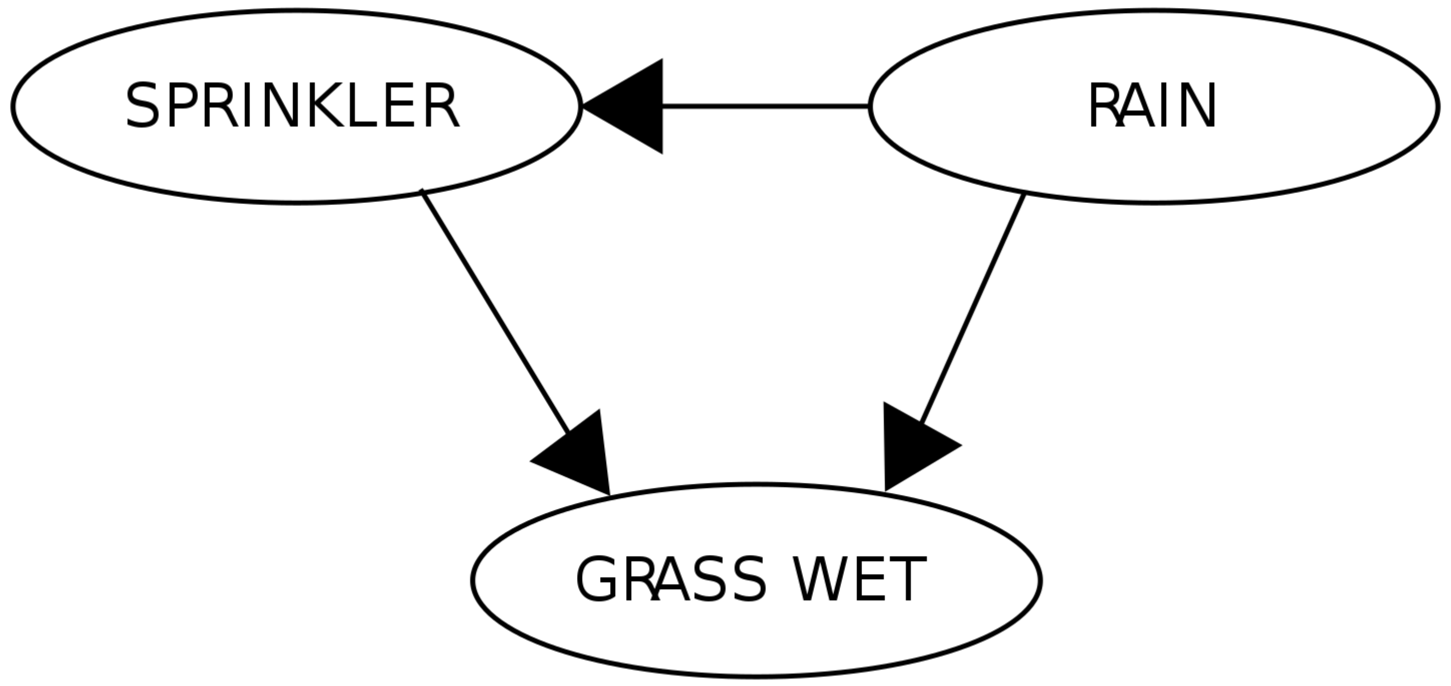
Q3: Given that grass is wet, how likely is that it rained?
associational

Example: Rain-Sprinkler-Grass

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

RAIN	SPRINKLER T	SPRINKLER F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN T	RAIN F
0.2	0.8

$p(R)$

SPRINKLER	RAIN	GRASS WET T	GRASS WET F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$p(W \mid S, R)$

Ladder of Causation

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

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

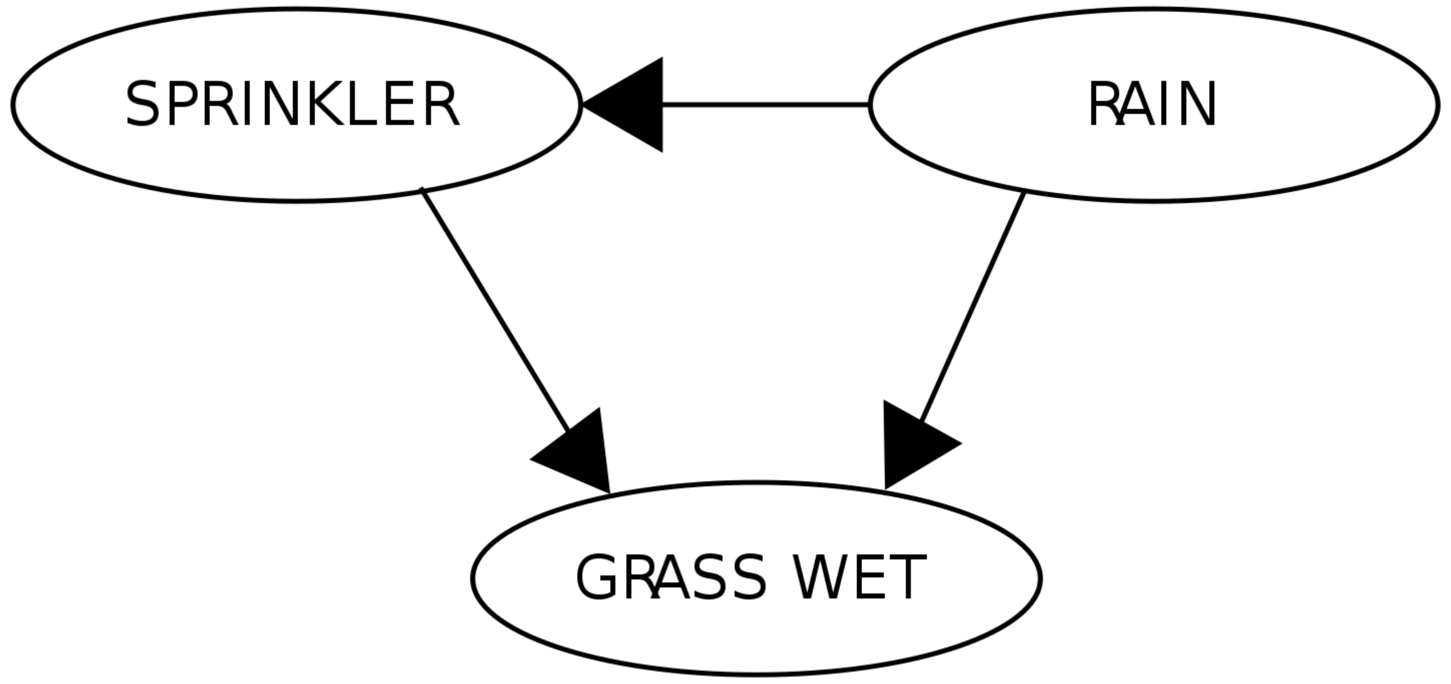
Q3: Given that grass is wet, how likely is that it rained?
associational

Example: Rain-Sprinkler-Grass

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

	SPRINKLER	
RAIN	T	F
F	0.4	0.6
T	0.01	0.99

$p(S \mid R)$



RAIN	
T	F
0.2	0.8

$p(R)$

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$p(W \mid S, R)$

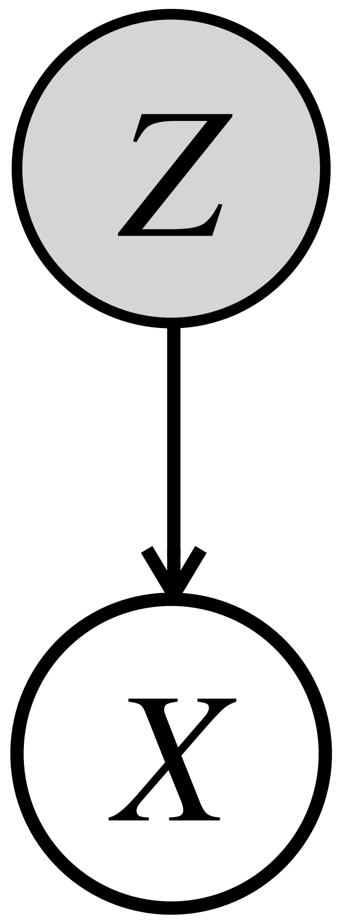
Ladder of Causation

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

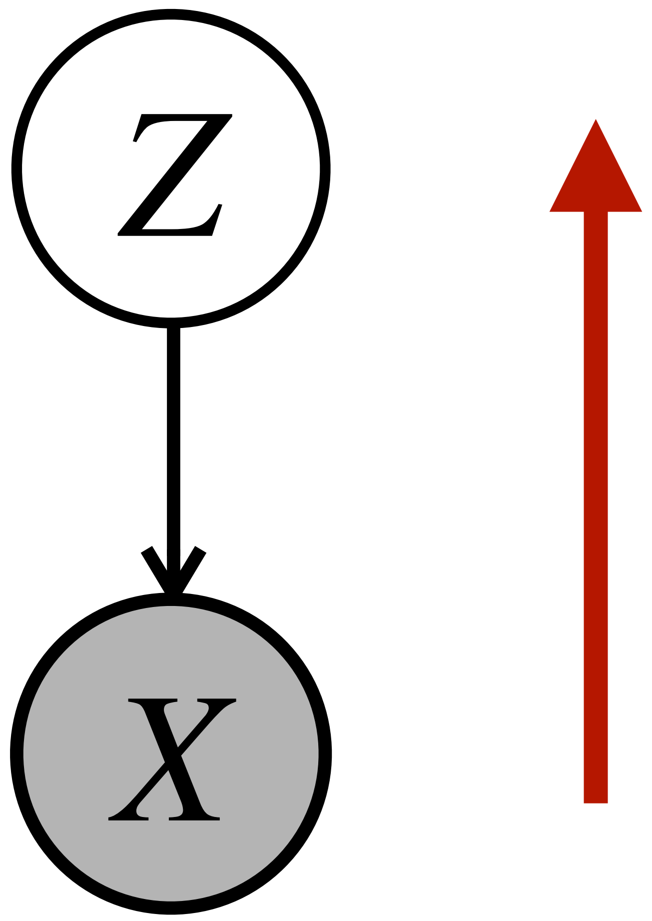
Q4: If we were to turn on the sprinkler, how likely would the grass be wet?
interventional

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

Programming

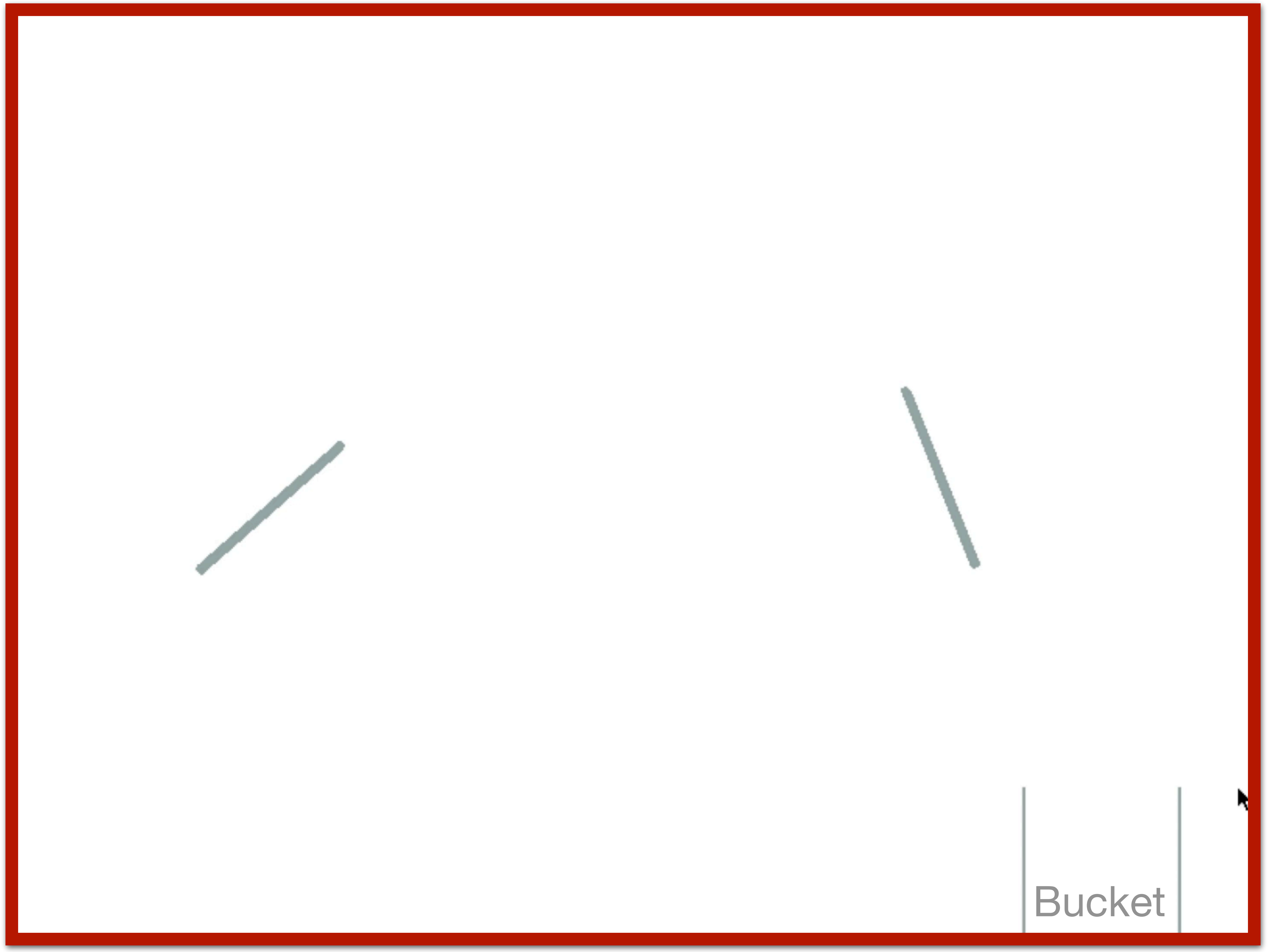


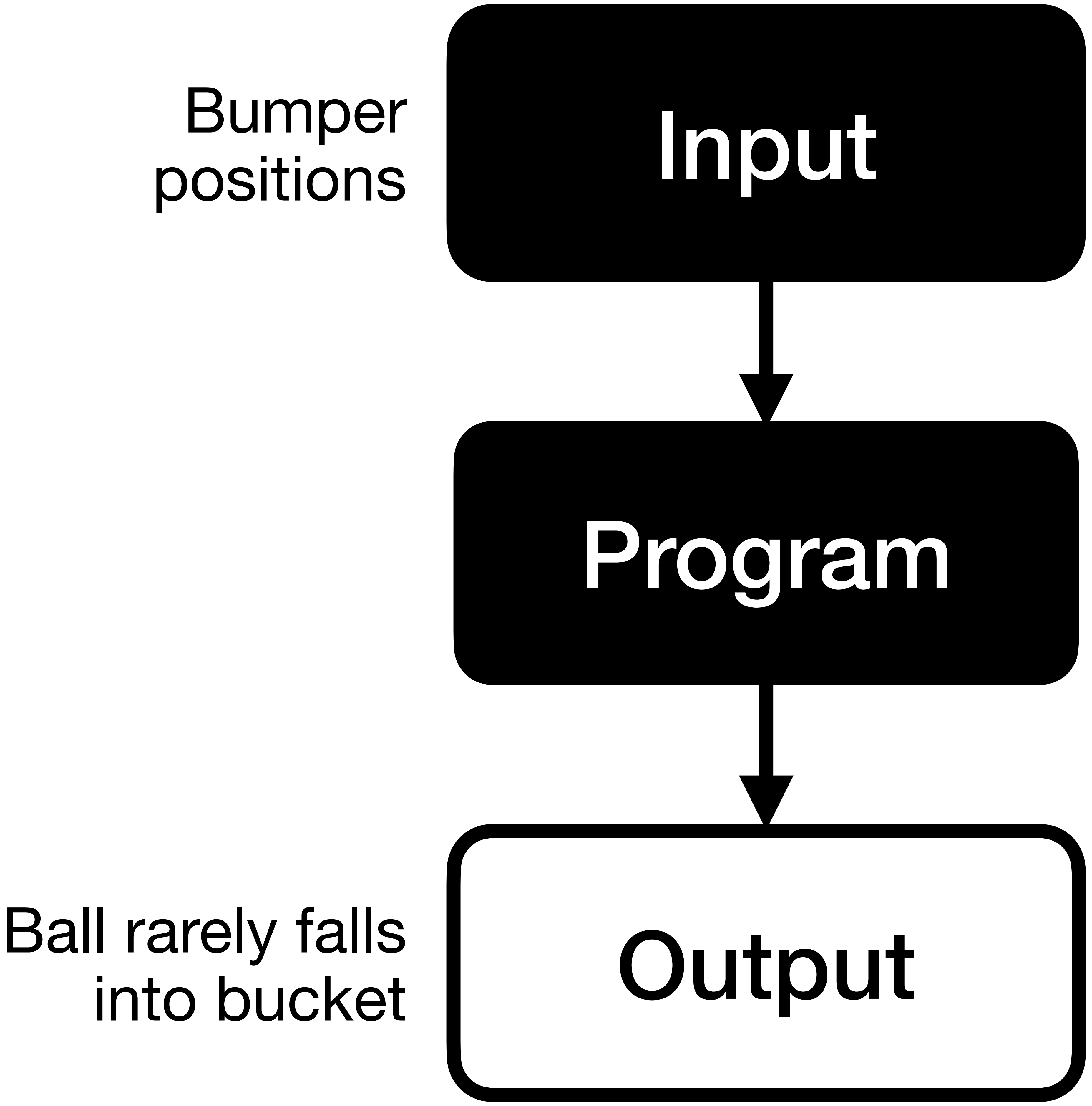
Probabilistic Programming



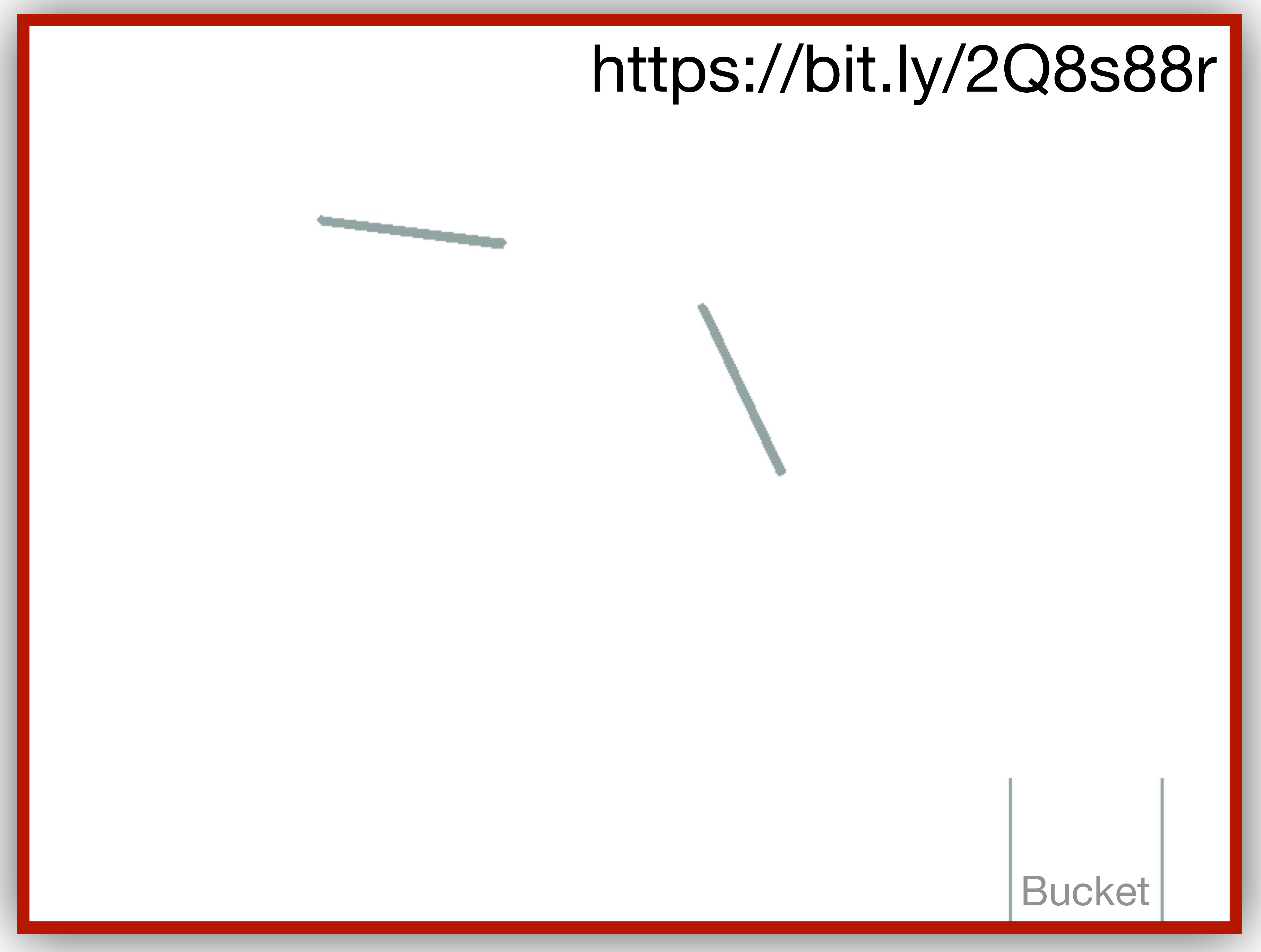
$$p(z\,|\,x) = \frac{p(x\,|\,z)\,p(z)}{p(x)}$$

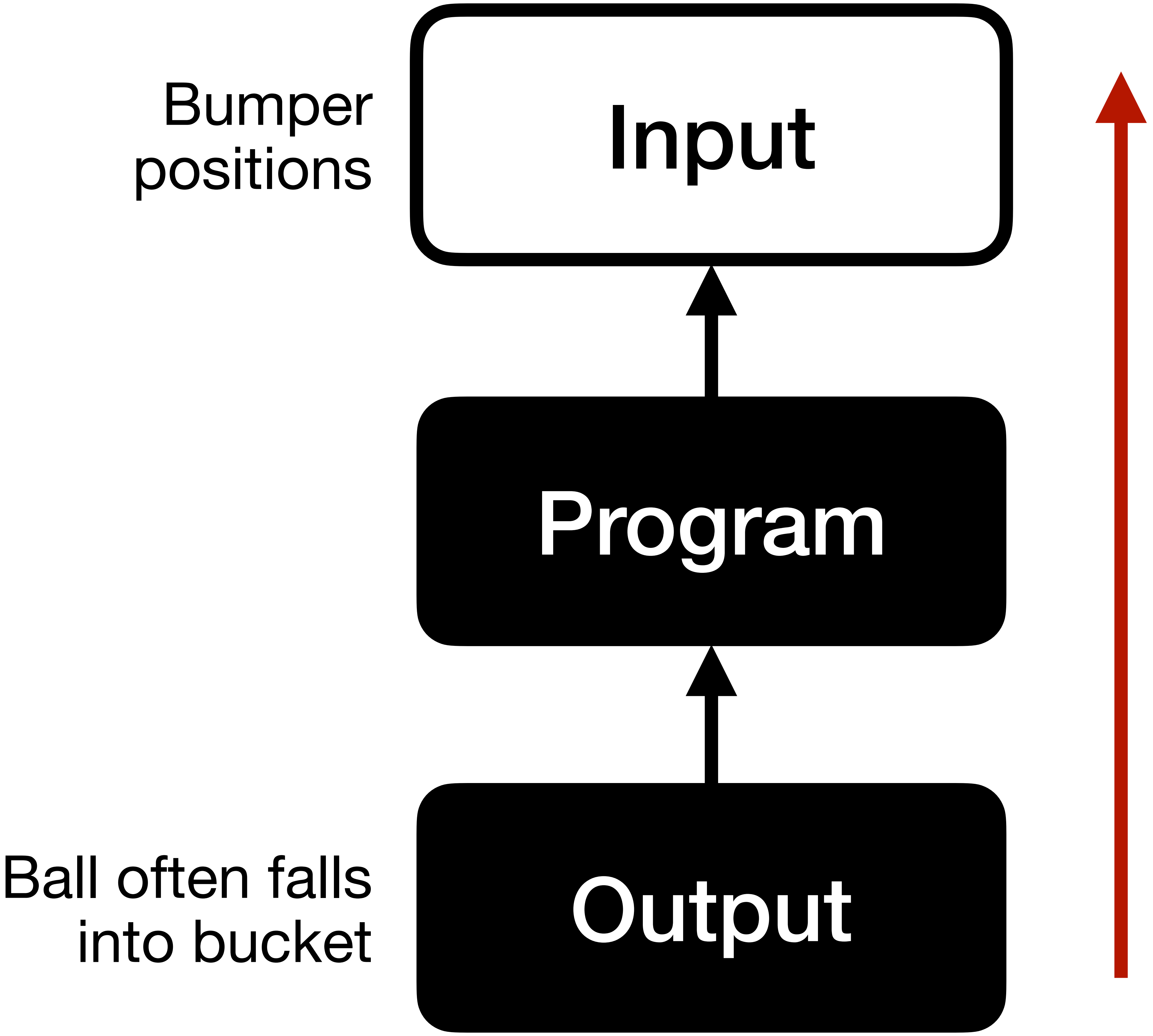
Example: Bouncing Balls



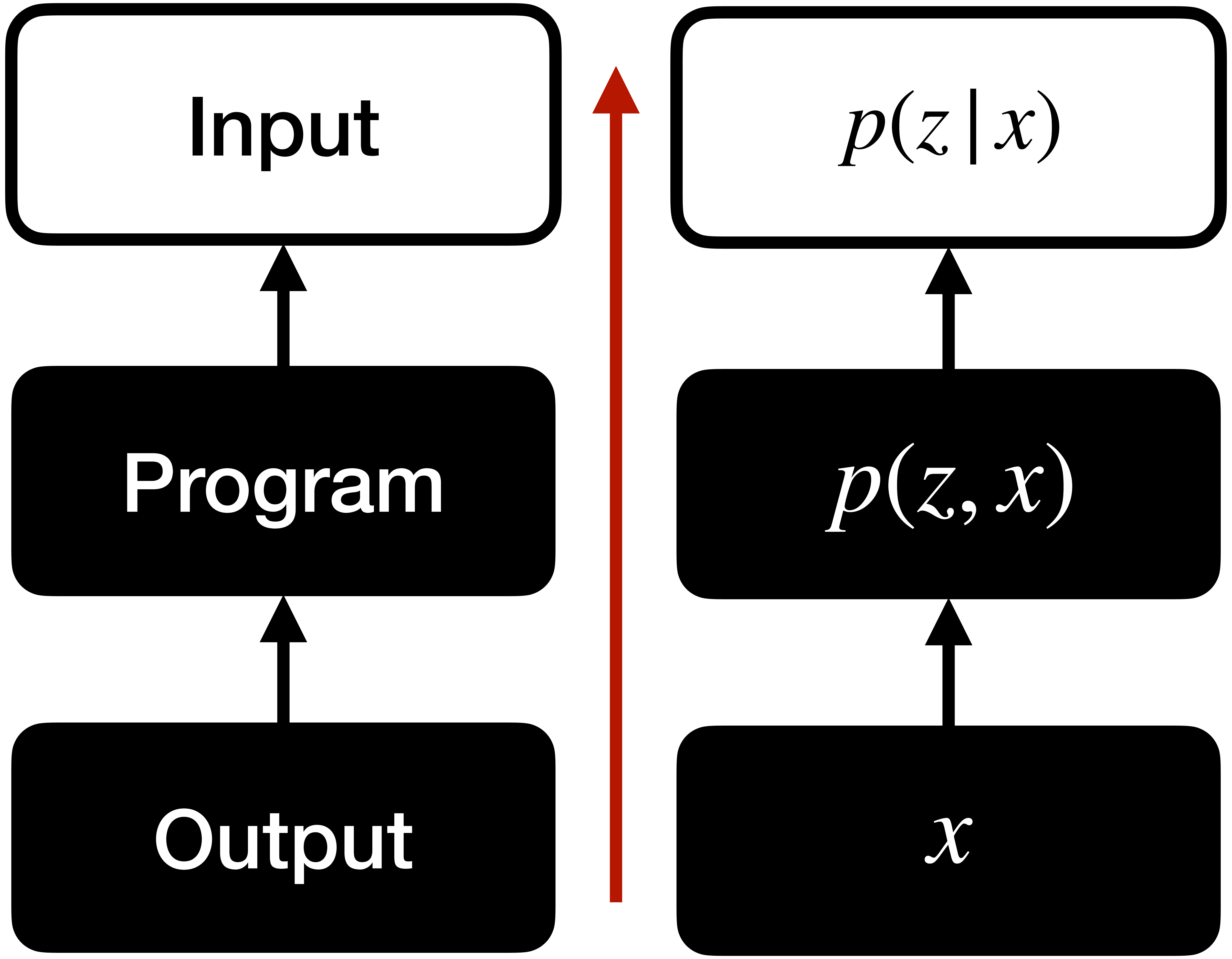


Example: Bouncing Balls into Bucket



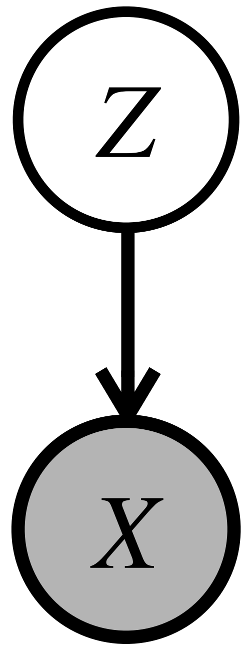


Applications of Probabilistic Programming



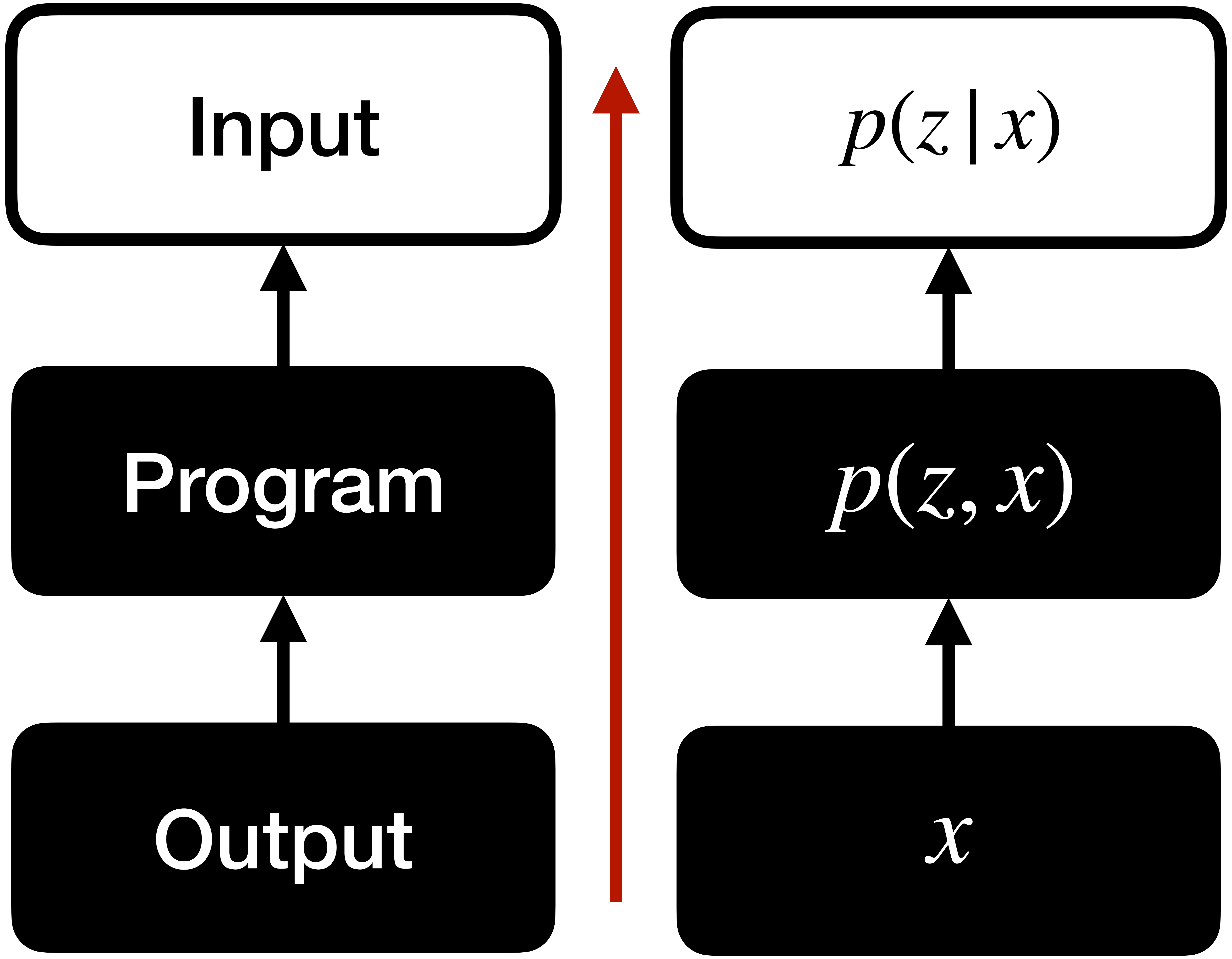
Program source code | Agent's policy/plan | Captcha letters

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



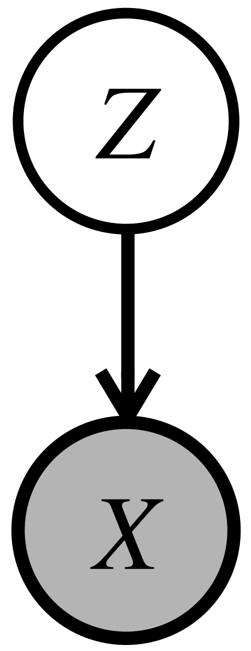
Input-output examples | Agent's reward | Captcha images

Applications of Probabilistic Programming



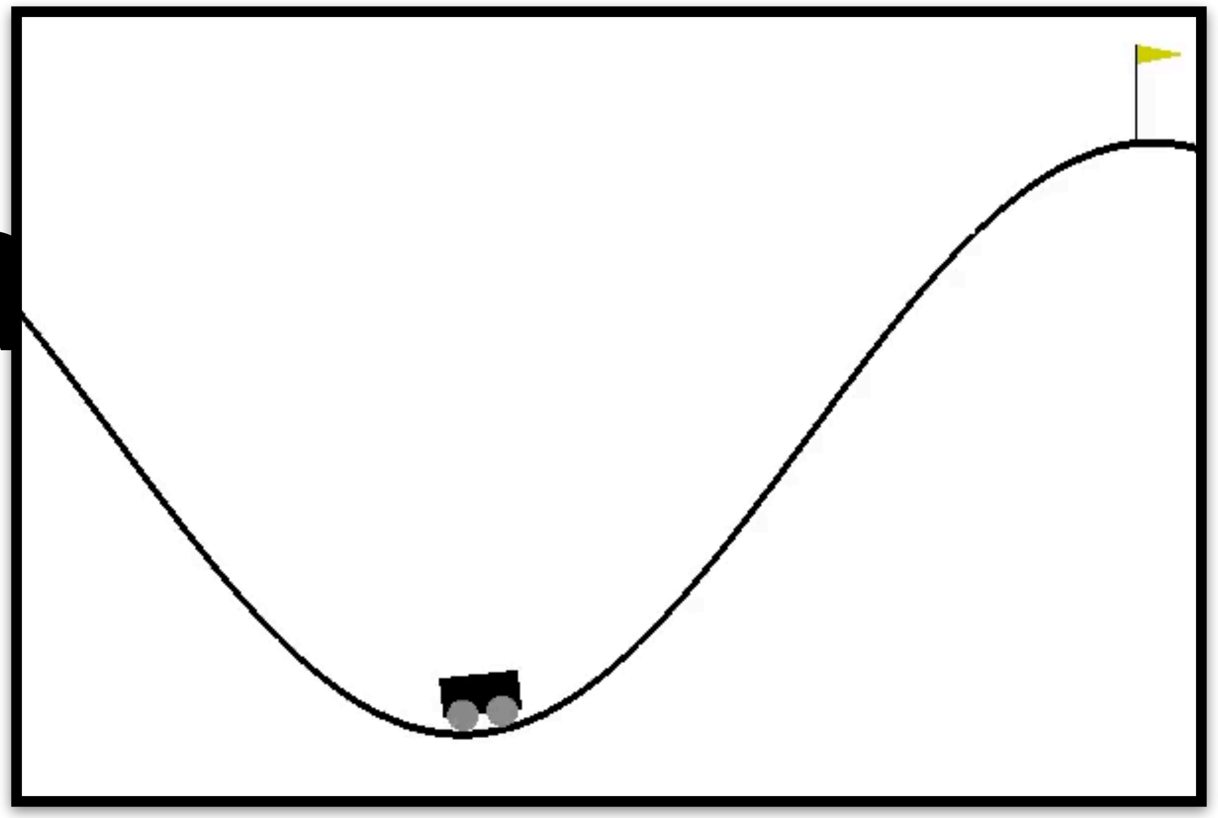
Program source code | Agent's policy/plan | Captcha letters

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



Input-output examples | Agent's reward | Captcha images

Applications of Probabilistic Progra



Input

Program

Output

$p(z \mid x)$

$p(z, x)$

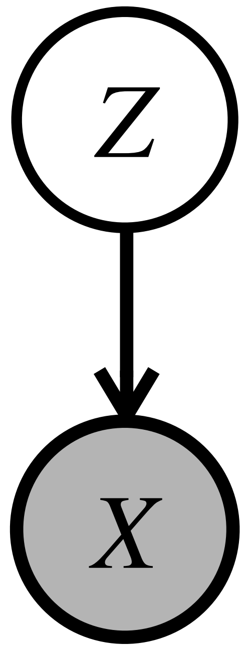
x

Program source code

Agent's policy/plan

Captcha letters

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

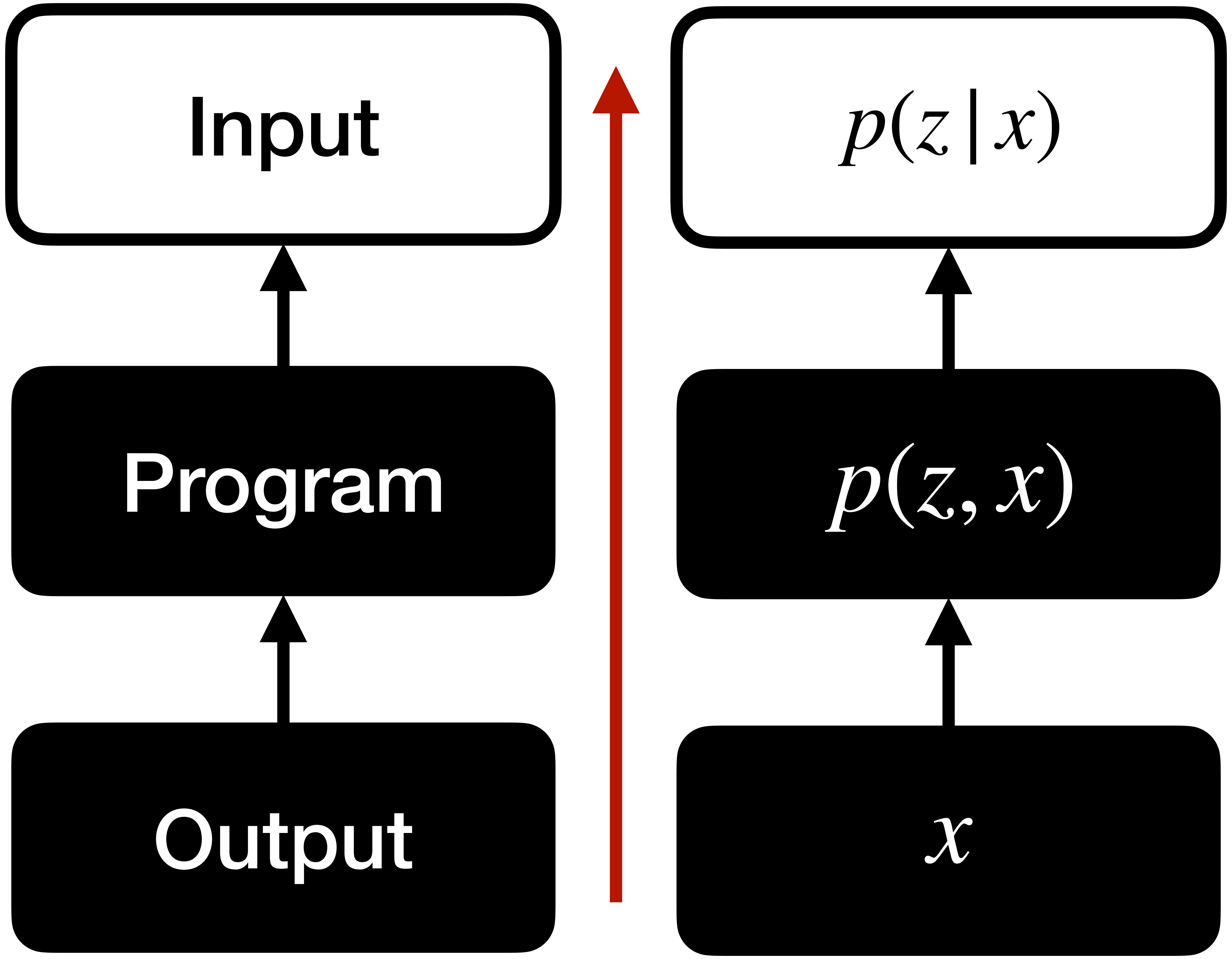


Input-output examples

Agent's reward

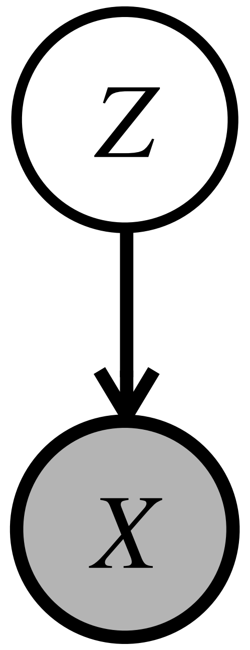
Captcha images

Applications of Probabilistic Programming



Program source code	Agent's policy/plan	Captcha letters

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



Input-output examples	Agent's reward	Captcha images

PL as an Abstraction Layer

Programs

PL

Interpreter / Compiler

PL as an Abstraction Layer

Programs | expression

PL

Interpreter / Compiler | solving

PPL as an Abstraction Layer

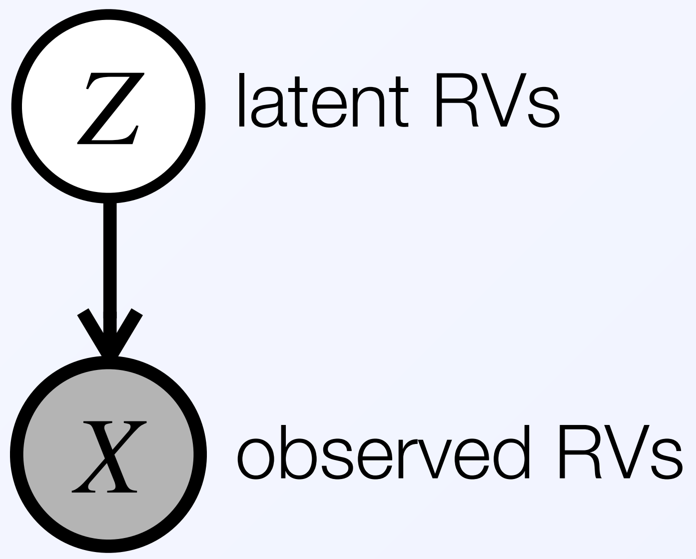
Probabilistic Programs — expression

PPL

Probabilistic Inference — solving

PPL as an Abstraction Layer

Probabilistic Programs



expression

PPL

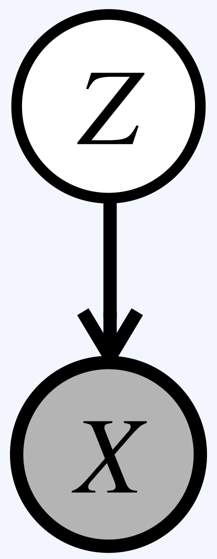
Probabilistic Inference

solving

PPL as an Abstraction Layer

data-generation process
generative model
stochastic simulation
decoders
inductive bias

Probabilistic Programs



latent RVs

observed RVs

expression

PPL

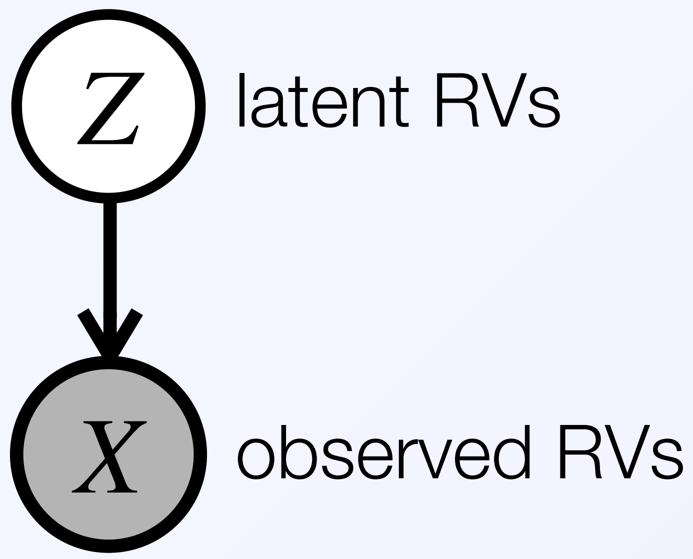
Probabilistic Inference

solving

PPL as an Abstraction Layer

data-generation process
generative model
stochastic simulation
decoders
inductive bias

Probabilistic Programs



expression

PPL

Probabilistic Inference

Interpreters & Compilers

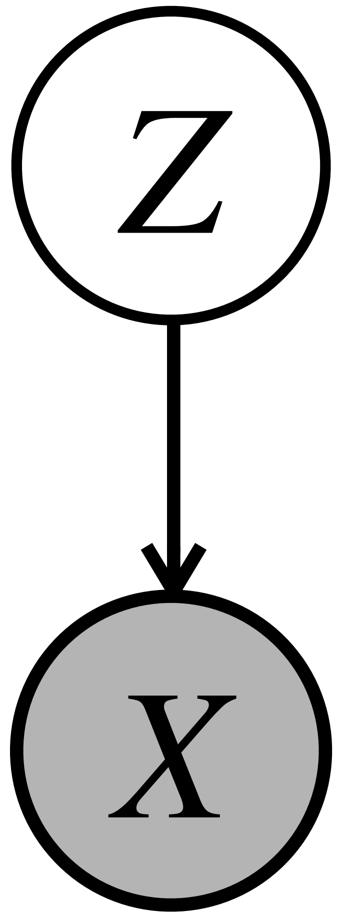
solving

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

What is hard about Bayesian inference?

Bayes' Theorem

$$\underbrace{p(z \mid x)}_{\text{posterior}} = \frac{\overbrace{p(z, x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}} = \frac{\overbrace{p(x \mid z)}^{\text{likelihood}}\,\overbrace{p(z)}^{\text{prior}}}{\underbrace{\boxed{\int p(z, x)\,\mathrm{d}z}}_{\text{marginal likelihood}}}$$

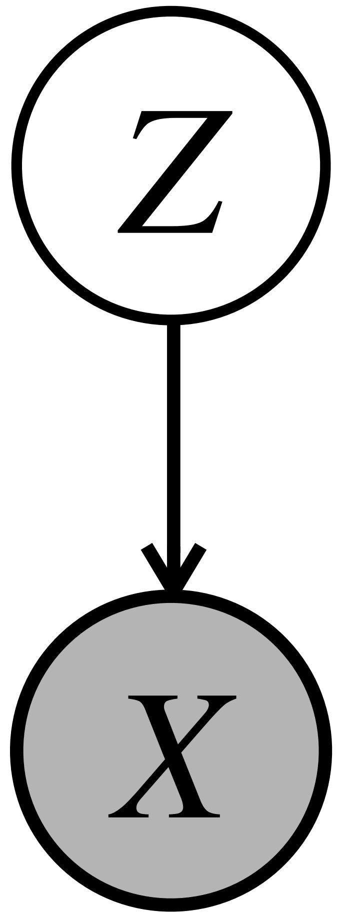


Integration of joint over all execution traces $\Rightarrow$ Enumerating all traces is unrealistic

What is hard about Bayesian inference?

Bayes' Theorem

$$\underbrace{p(z \mid x)}_{\text{posterior}} = \frac{\overbrace{p(z, x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}} = \frac{\overbrace{p(x \mid z)}^{\text{likelihood}}\,\overbrace{p(z)}^{\text{prior}}}{\underbrace{\boxed{\int p(z, x)\,\mathrm{d}z}}_{\text{marginal likelihood}}}$$



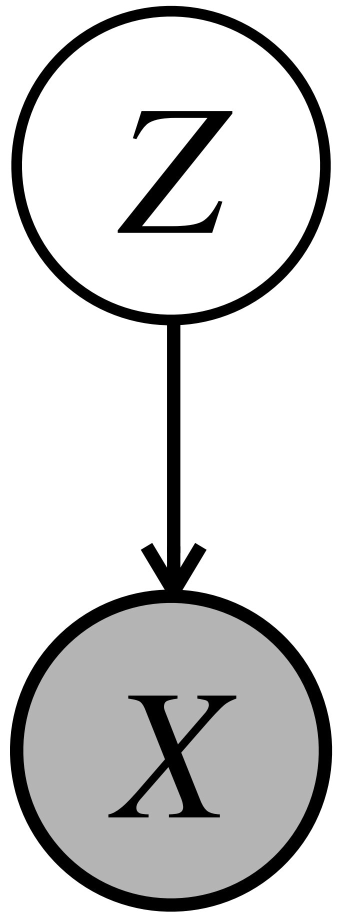
Integration of joint over all execution traces ⇨ Enumerating all traces is unrealistic

Joint is defined by a program ⇨ Integration rarely has analytical solutions

Have to Approximate or Limit Expressivity

Bayes' Theorem

$$\underbrace{p(z\,|\,x)}_{\text{posterior}} = \frac{\overbrace{p(z,x)}^{\text{joint}}}{\underbrace{p(x)}_{\text{evidence}}} = \frac{\overbrace{p(x\,|\,z)}^{\text{likelihood}}\,\overbrace{p(z)}^{\text{prior}}}{\underbrace{\int p(z,x)\,\mathrm{d}z}_{\text{marginal likelihood}}}$$



Integration of joint over all execution traces ⇨ Enumerating all traces is unrealistic

Joint is defined by a program ⇨ 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

$\Rightarrow$ 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, …

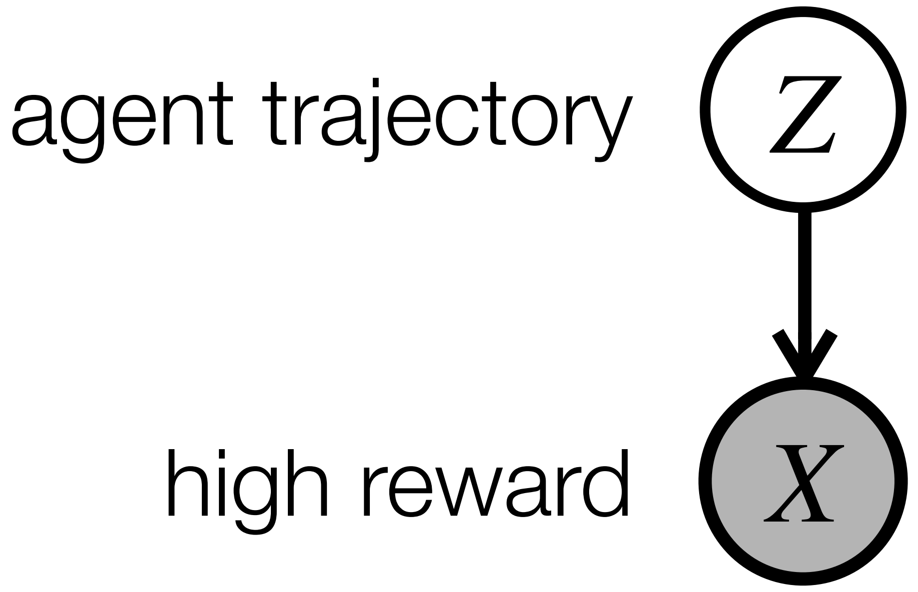
PPL as an Abstraction Layer

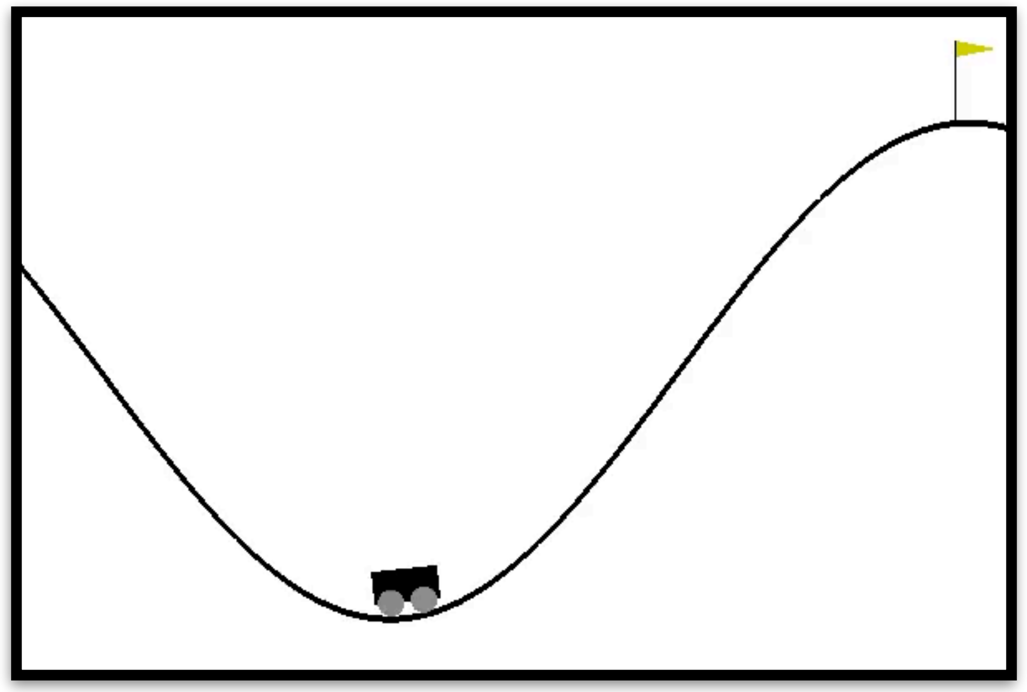
Probabilistic Programs — expression

PPL

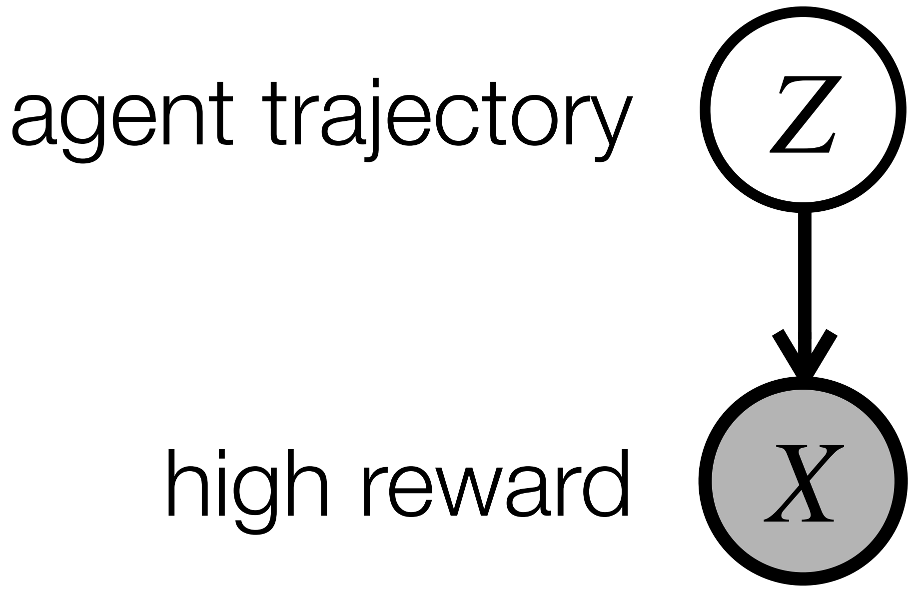
Probabilistic Inference — solving

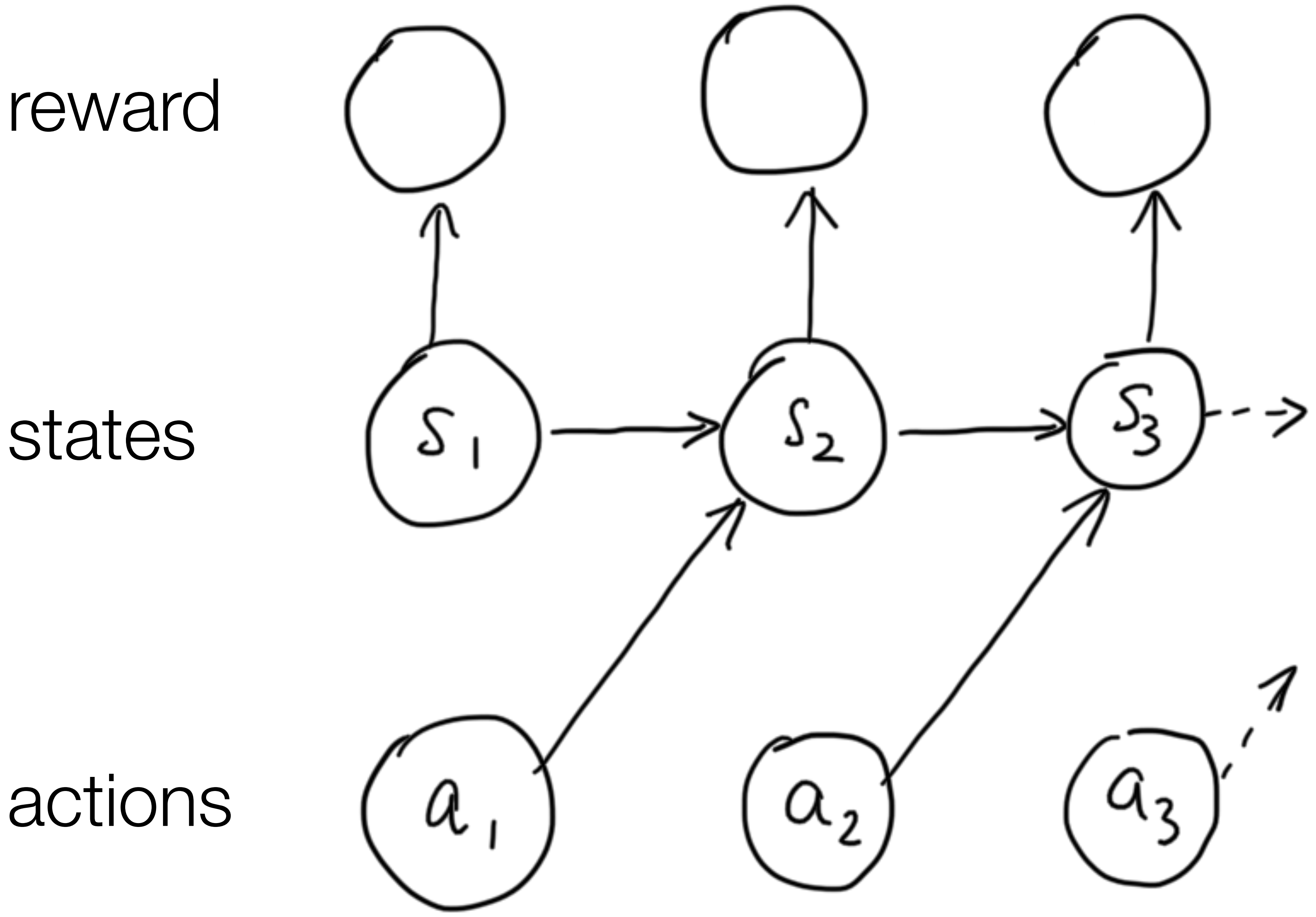
Example: Reinforcement Learning





Example: Reinforcement Learning



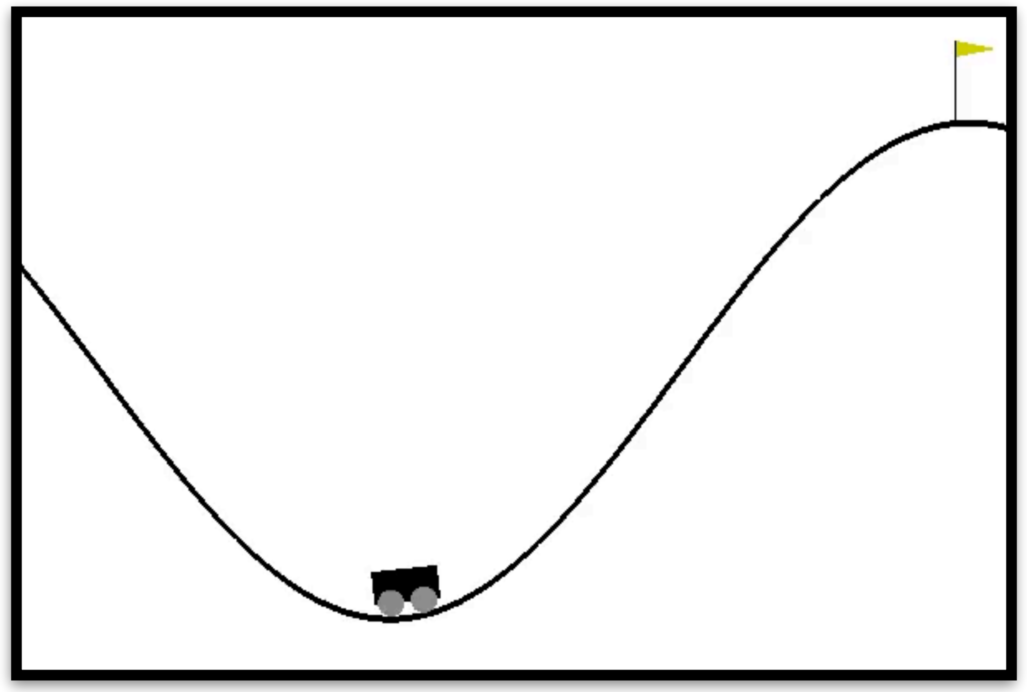


Environment

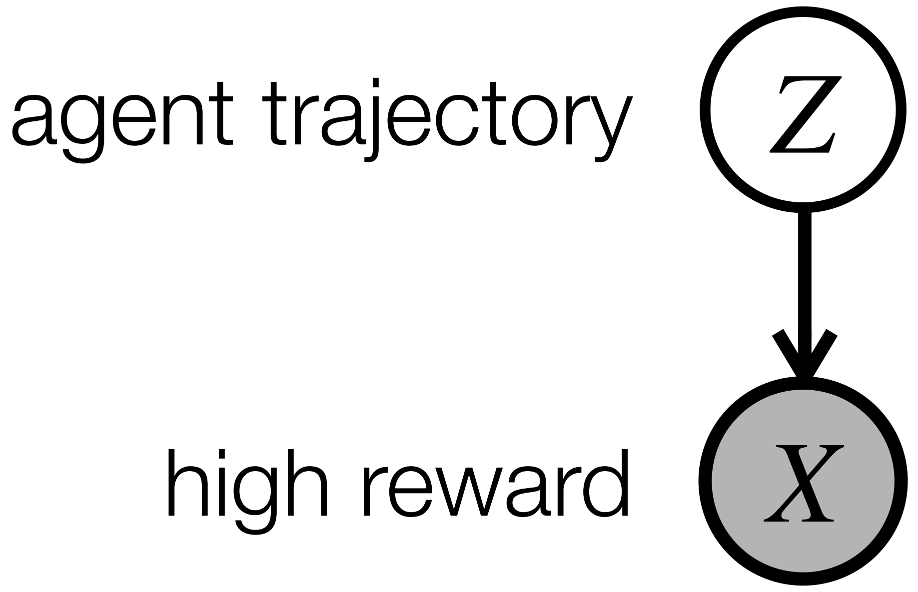
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

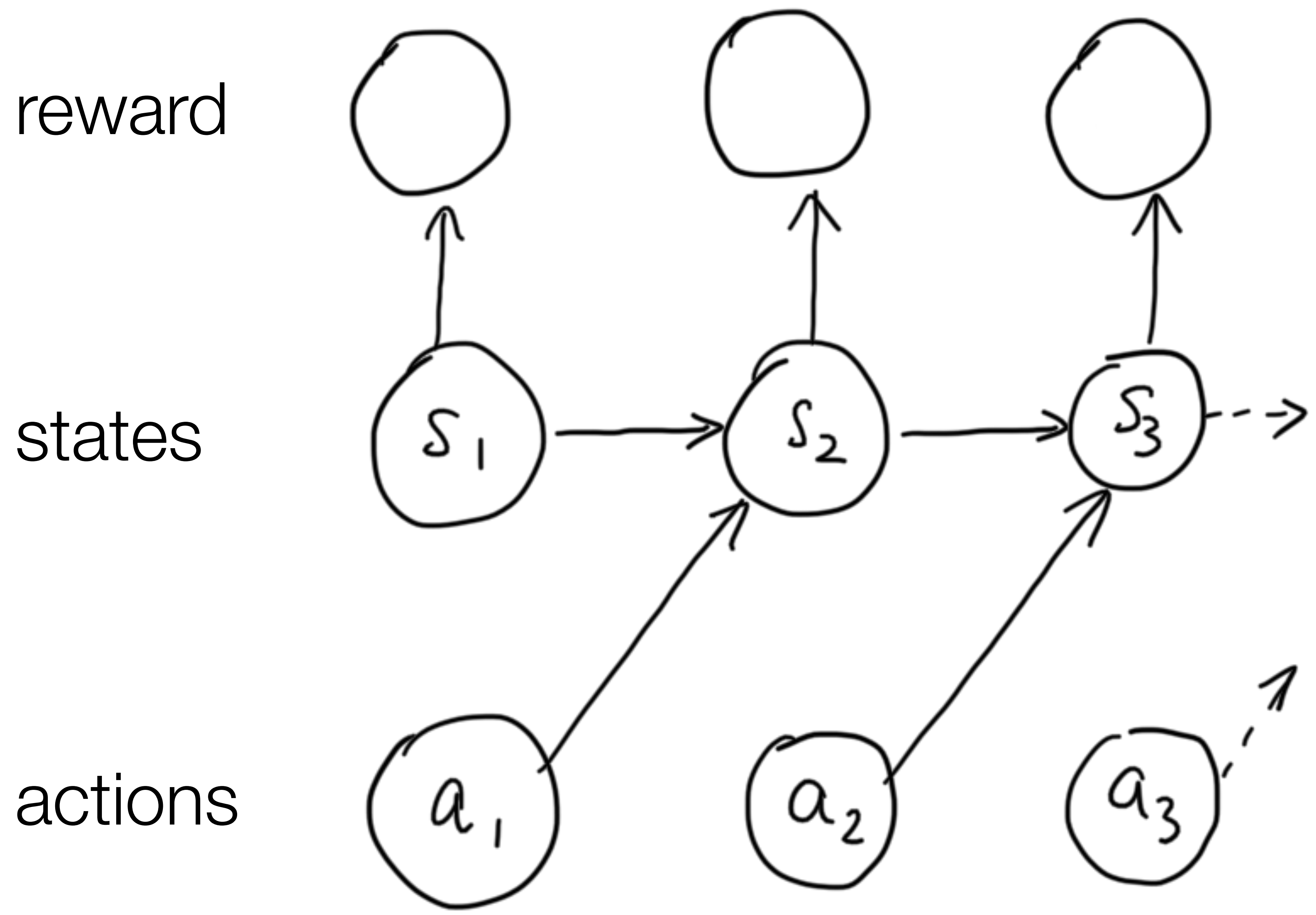
Agent

```
def MDP(state): // recursive MDP description
```



Example: Reinforcement Learning



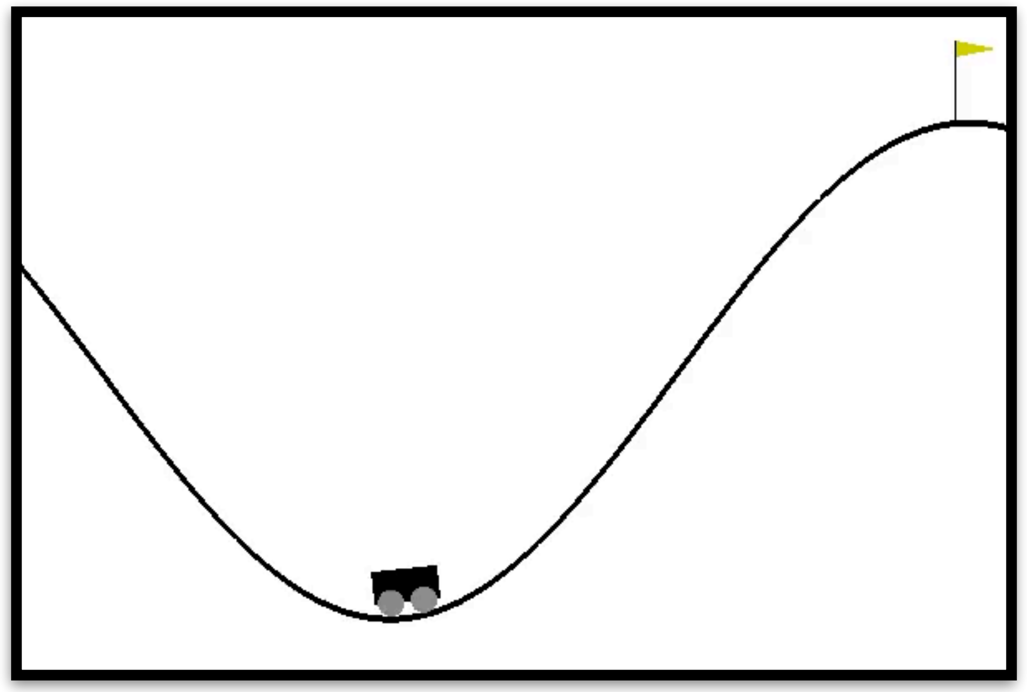


Environment

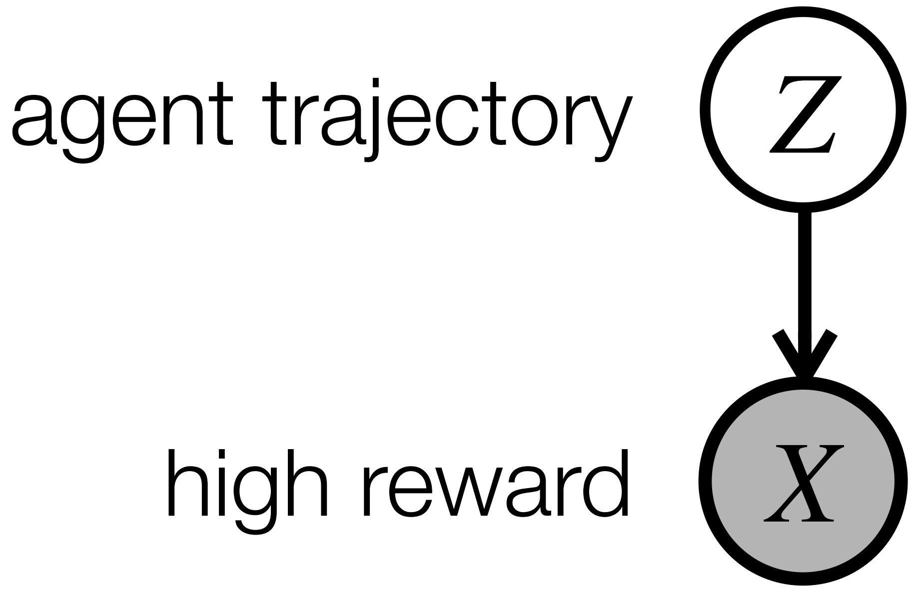
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

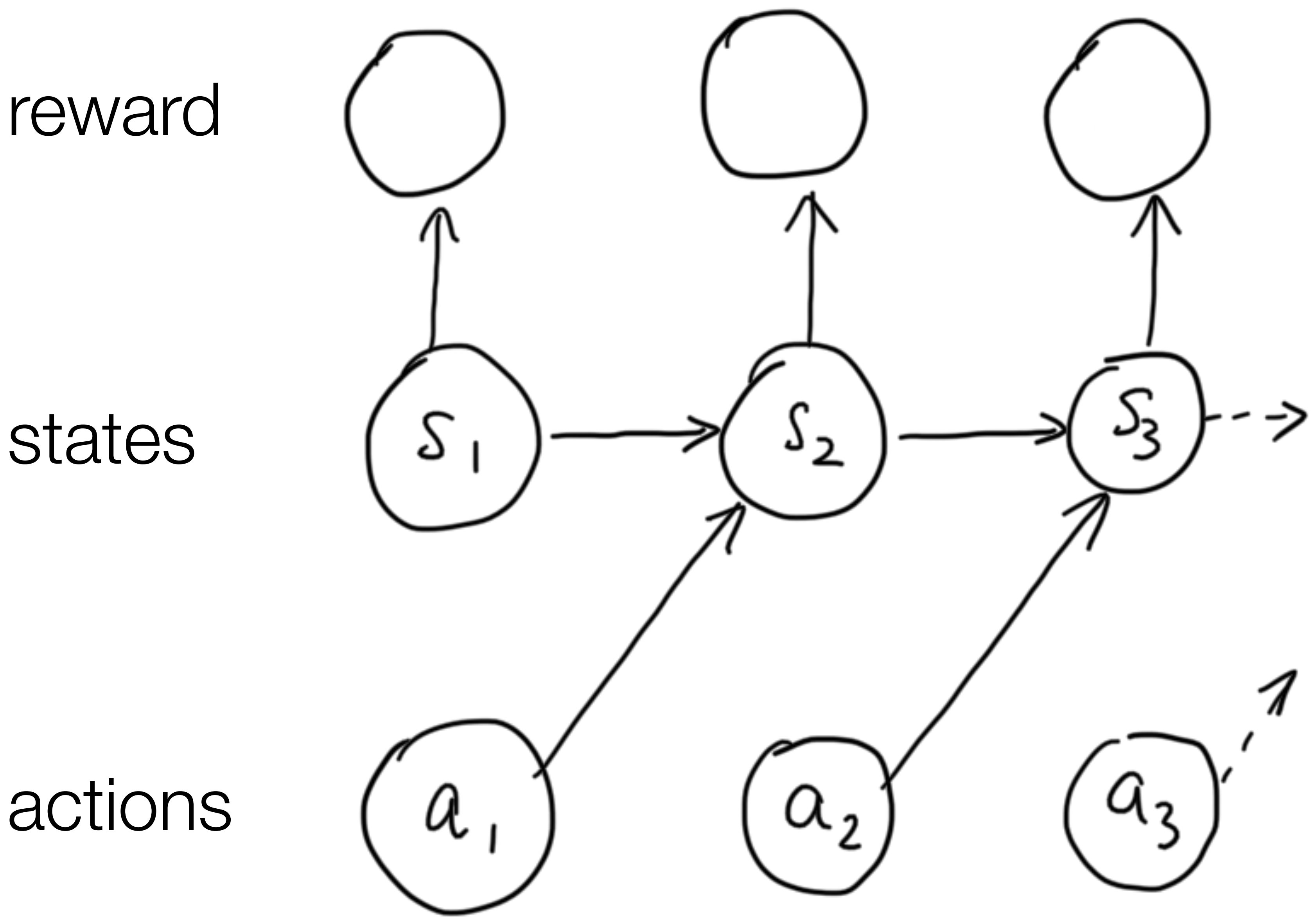
Agent

```
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(…) // sample action from prior
    nextState = transition(state, action)

    MDP(nextState) // recurse
```



Example: Reinforcement Learning





Goal of inference:

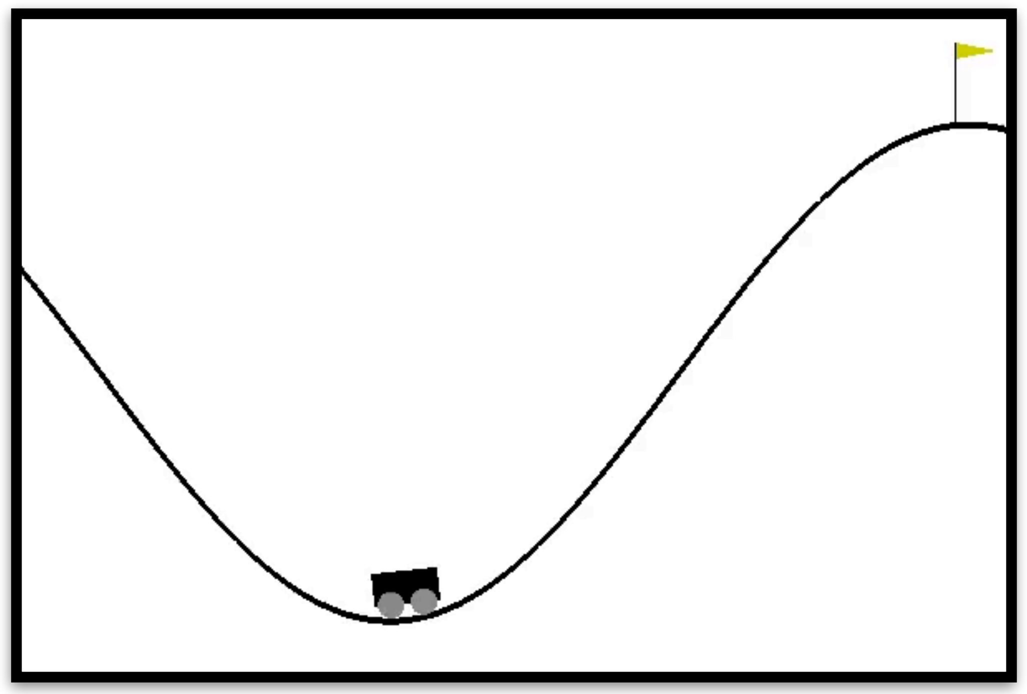
a policy function $\pi : \mathrm{State} \rightarrow \mathrm{Action}$

Environment

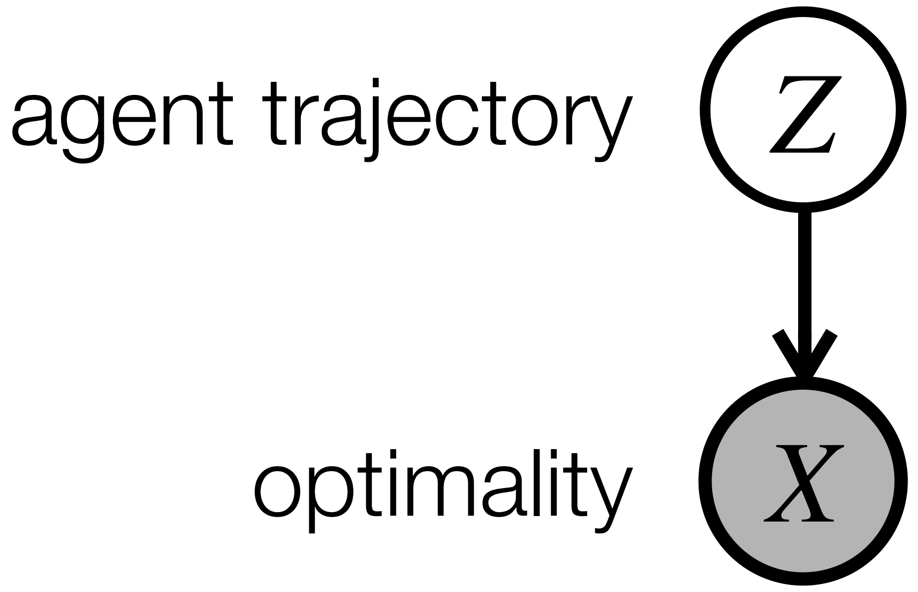
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

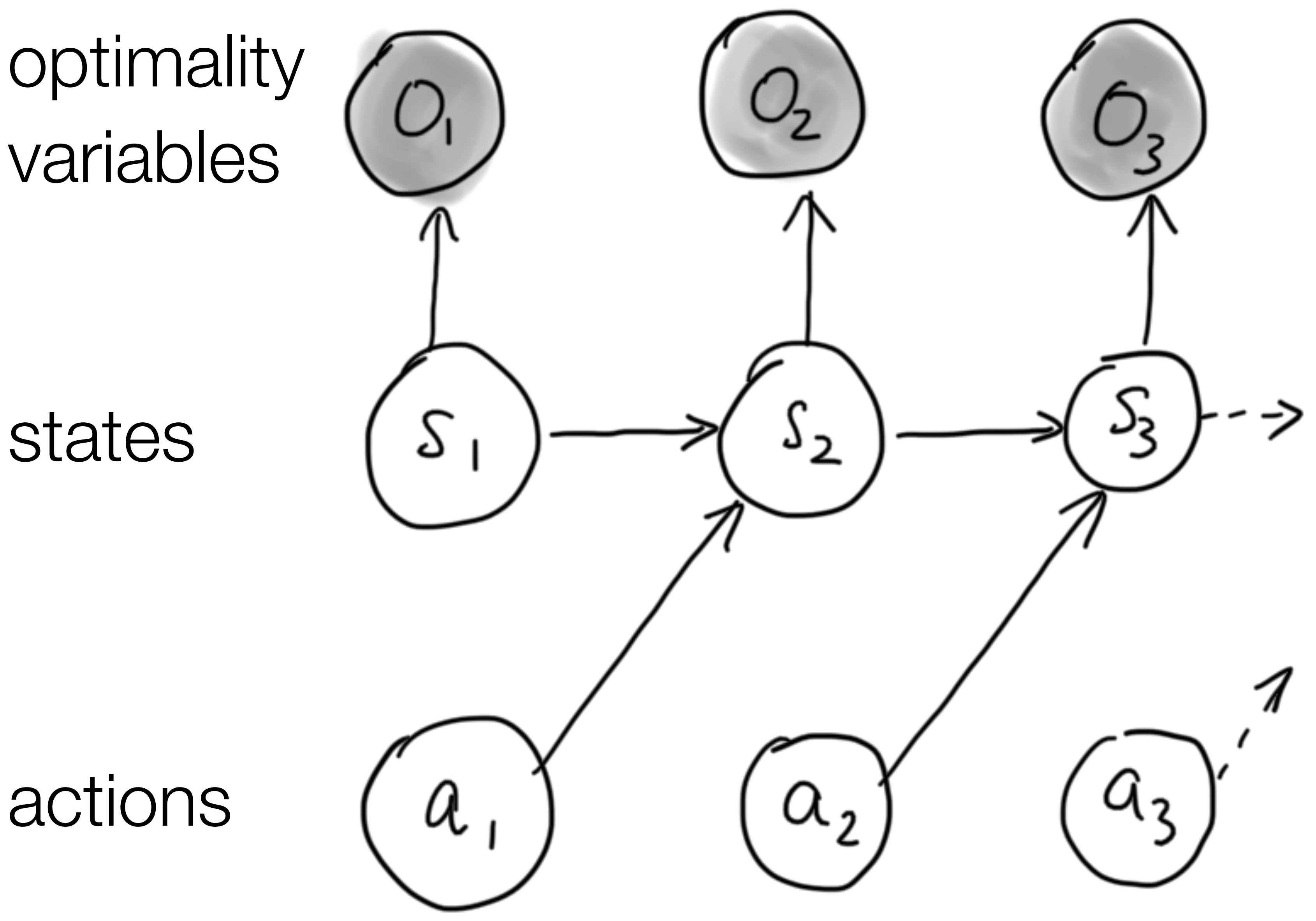
Agent

```
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(…) // sample action from prior
    nextState = transition(state, action)

    MDP(nextState) // recurse
```



Example: Reinforcement Learning





$$p(O_t = 1 \mid s_t) \stackrel{def}{=} ?$$

Goal of inference:

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

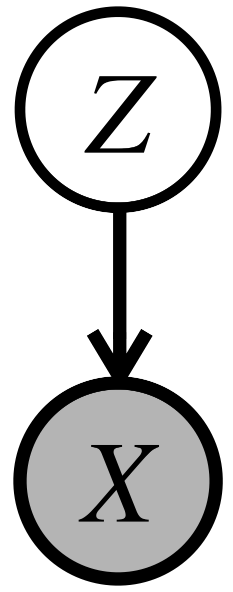
Environment

```
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

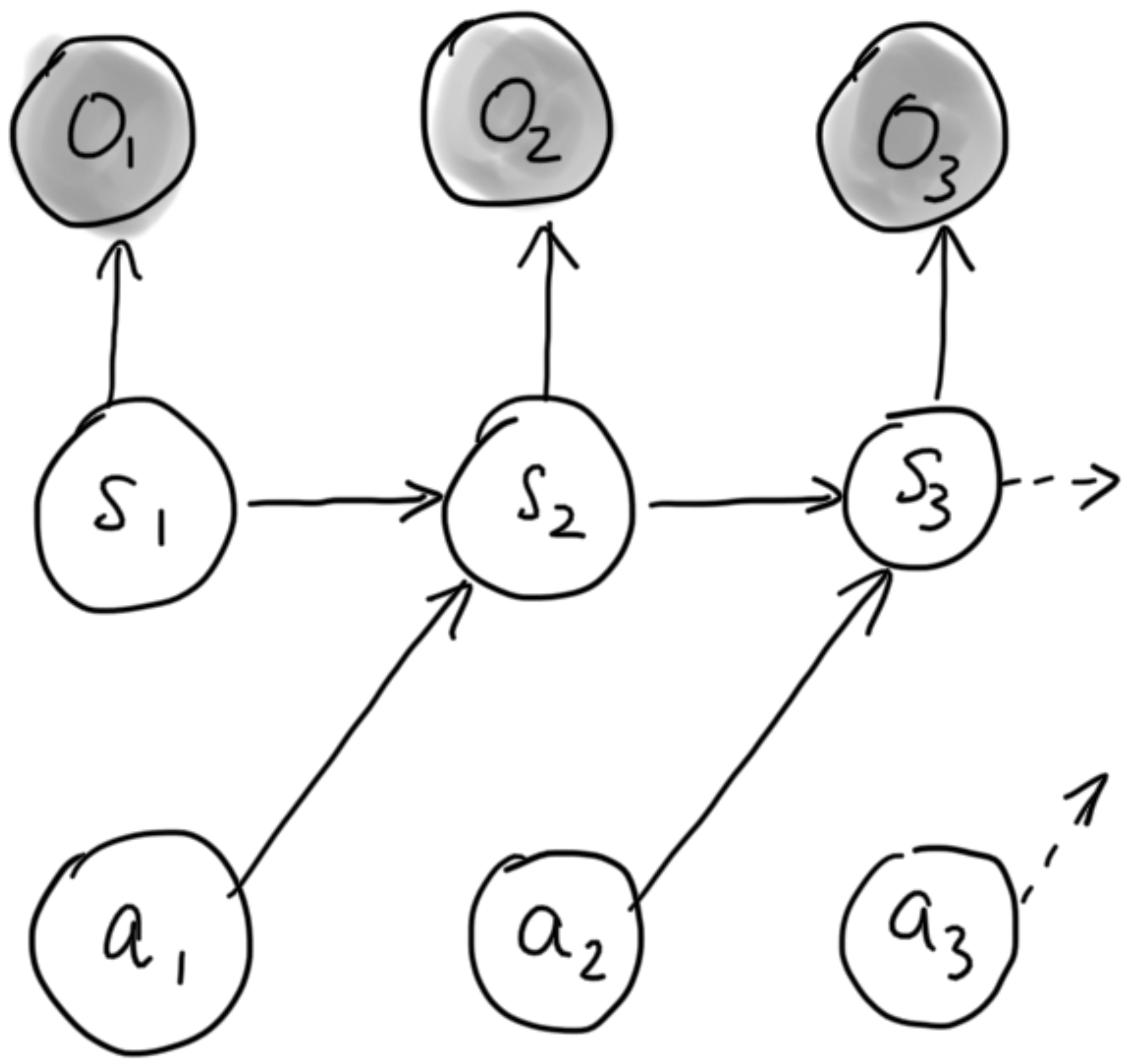
agent trajectory


optimality

optimality variables

states

actions



$$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$$

Goal of inference:

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

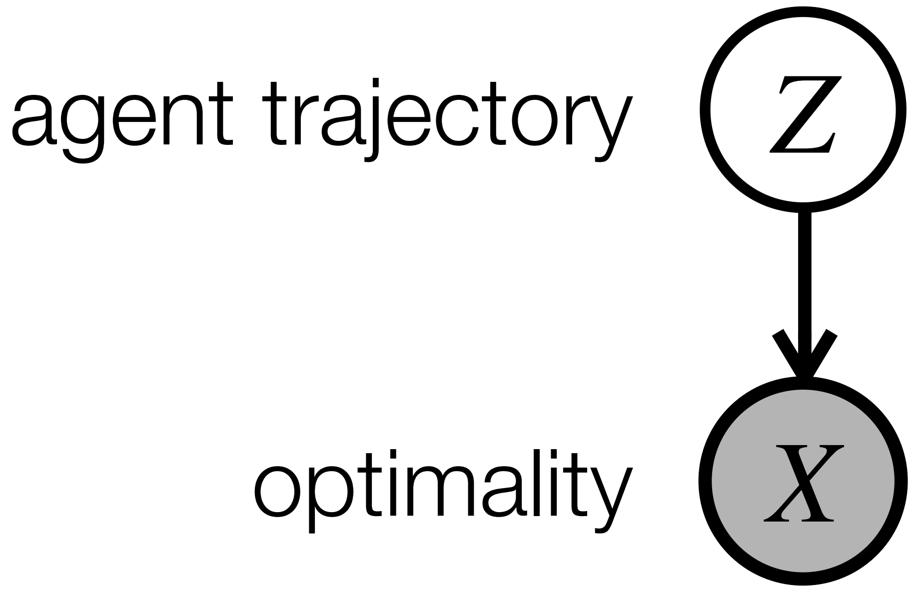
Environment

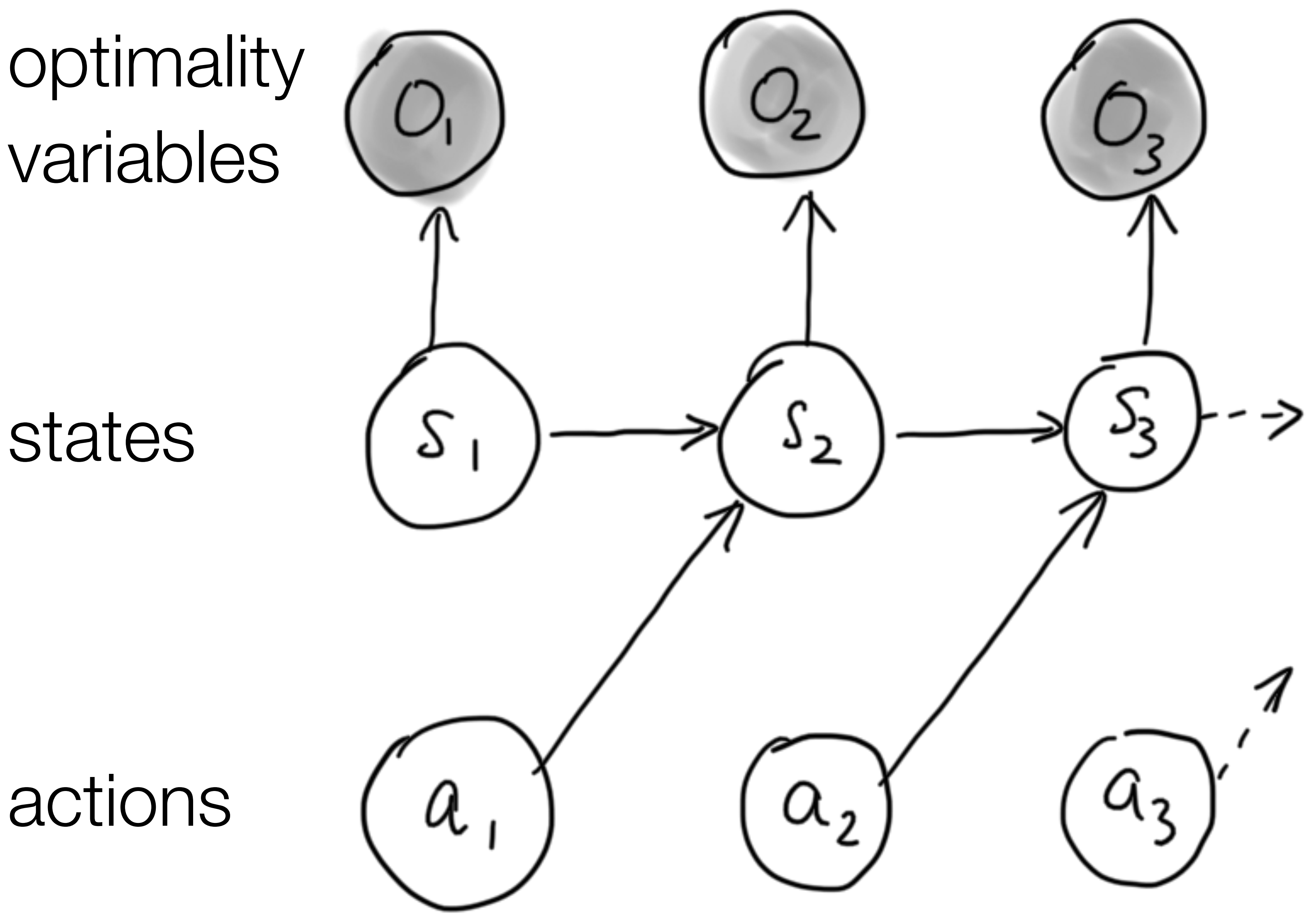
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

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

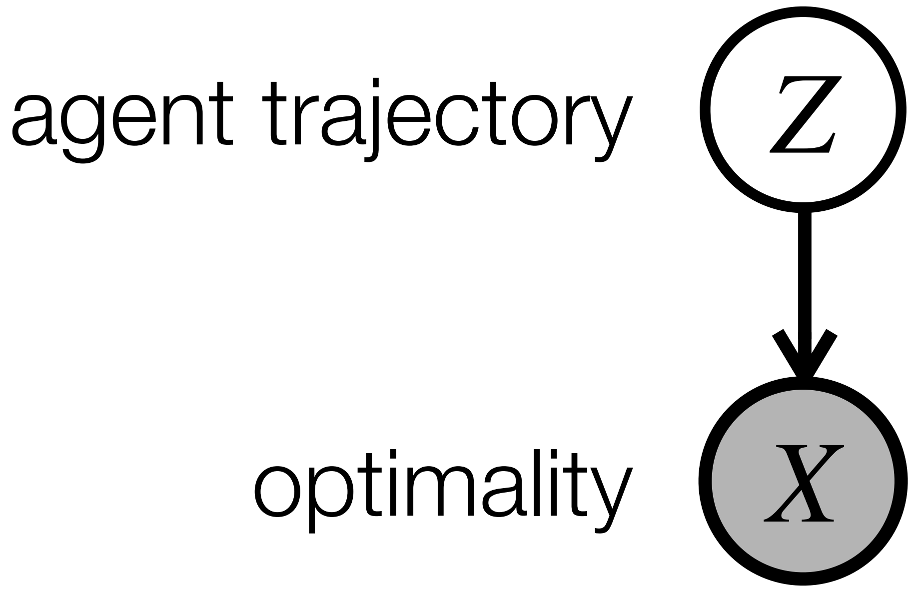
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

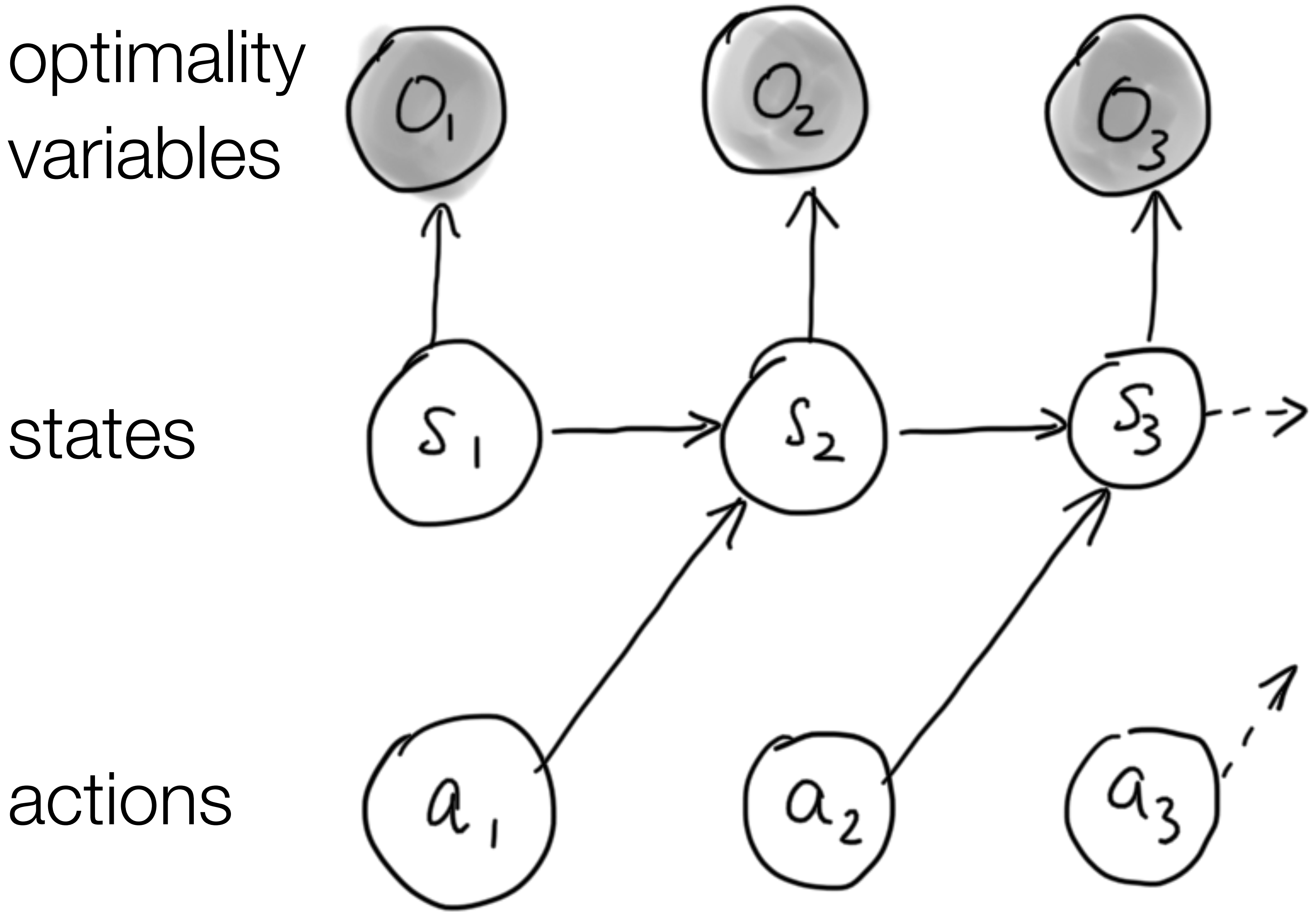
Agent

```
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
```

$$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$$

Example: Reinforcement Learning





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

Environment

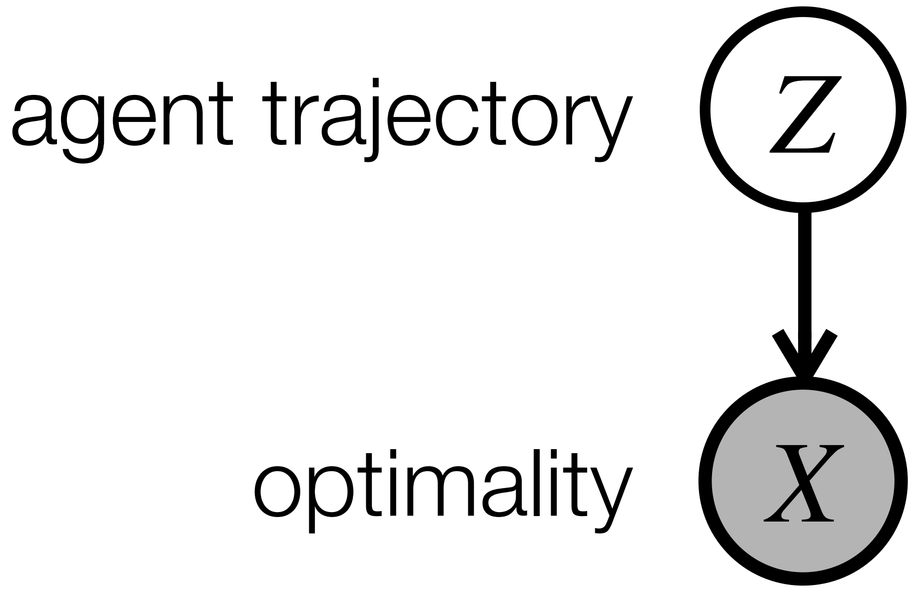
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

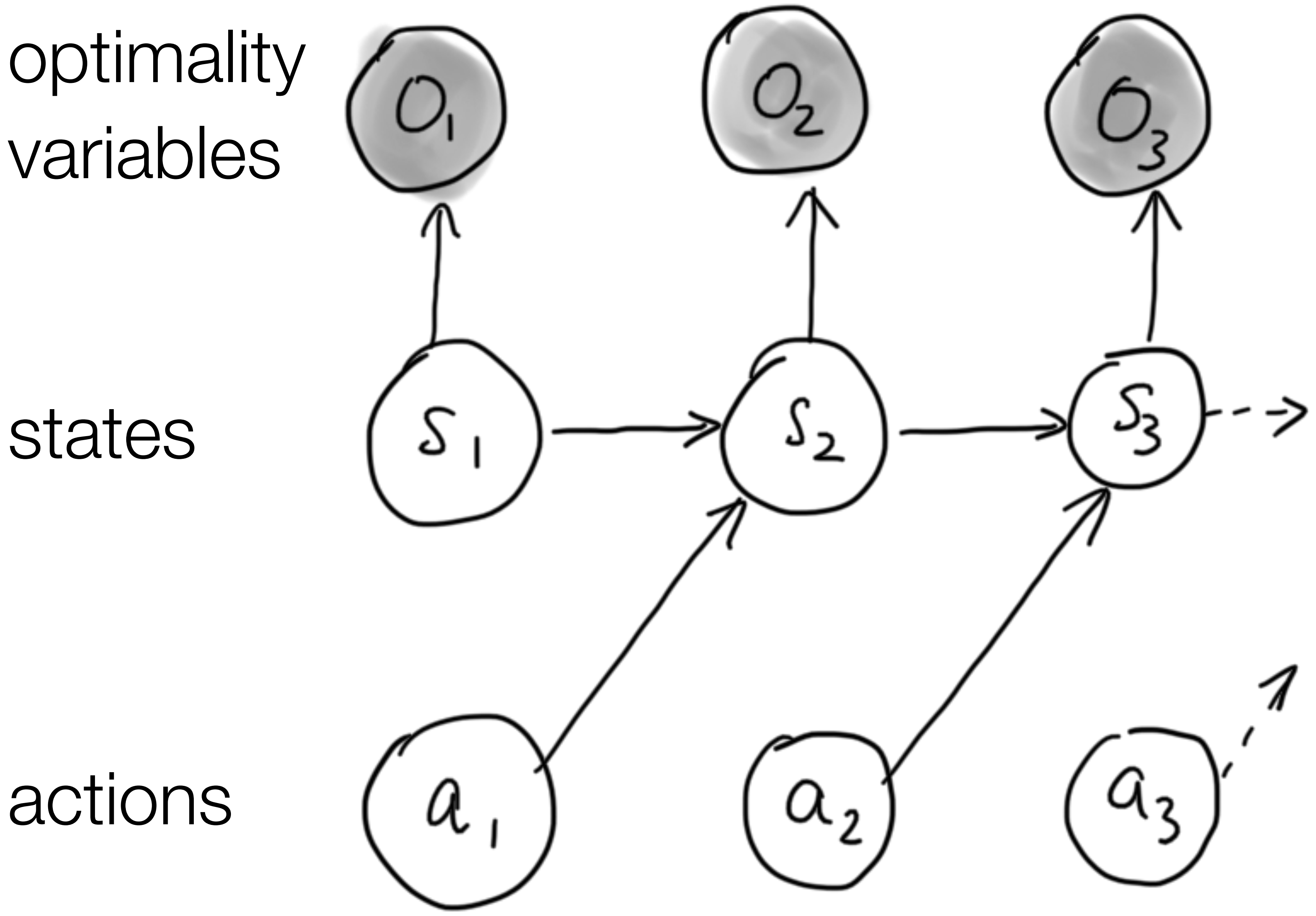
Agent

```
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
```

$$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$$

Example: Reinforcement Learning





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

Environment

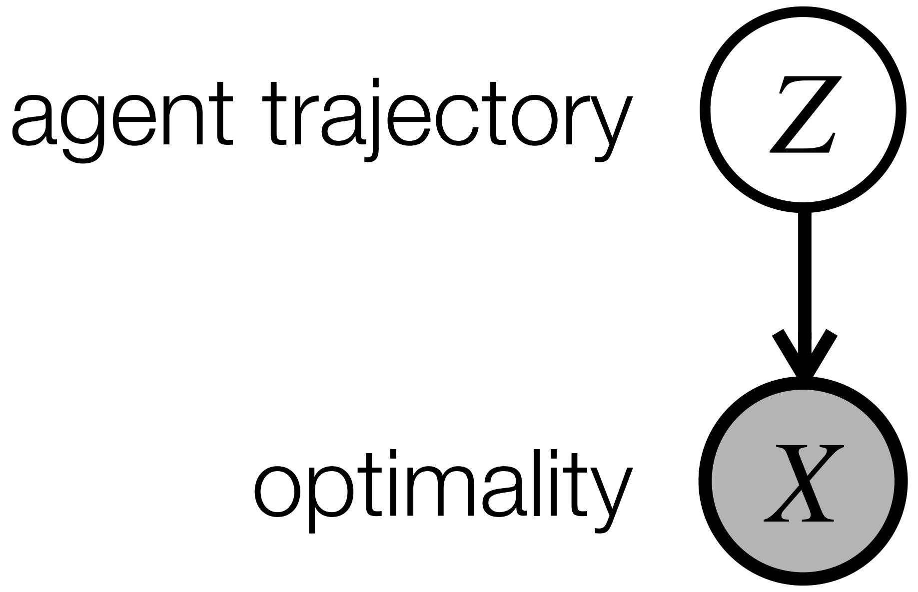
```
def reward(state) // immediate reward
def transition(state, action) // step the environment
```

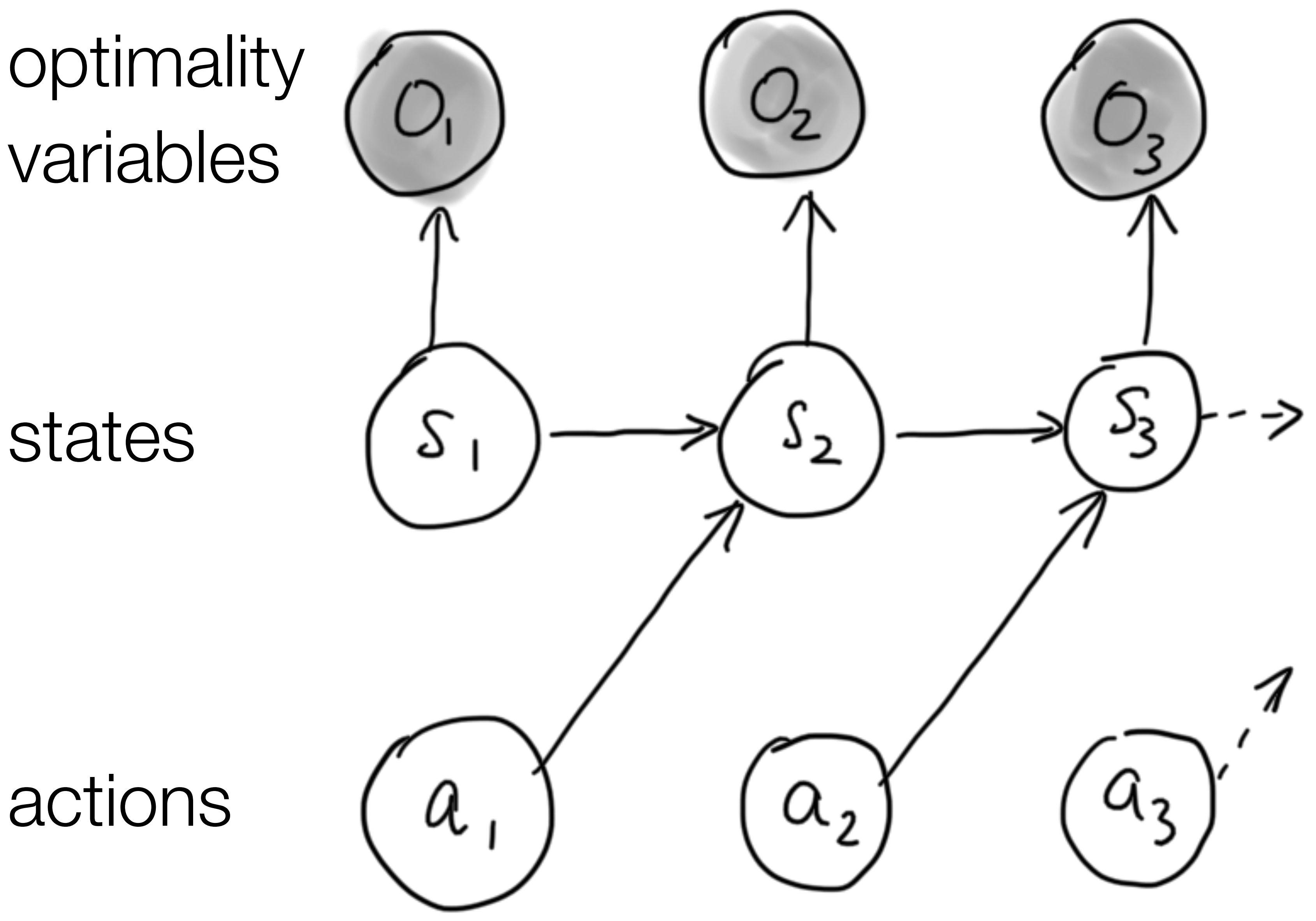
Agent

```
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
```

$$p(O_t = 1 \mid s_t) \overset{def}{=} \exp(r(s_t))$$

Example: Reinforcement Learning





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

a policy function $\pi : \mathrm{State} \to \mathrm{Dist[Action]}$ leading to optimal trajectory, rather than a trajectory per se

Environment

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

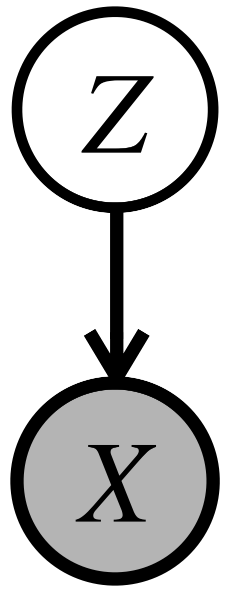
Agent

```
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
```

$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$

Example: Reinforcement Learning

agent trajectory

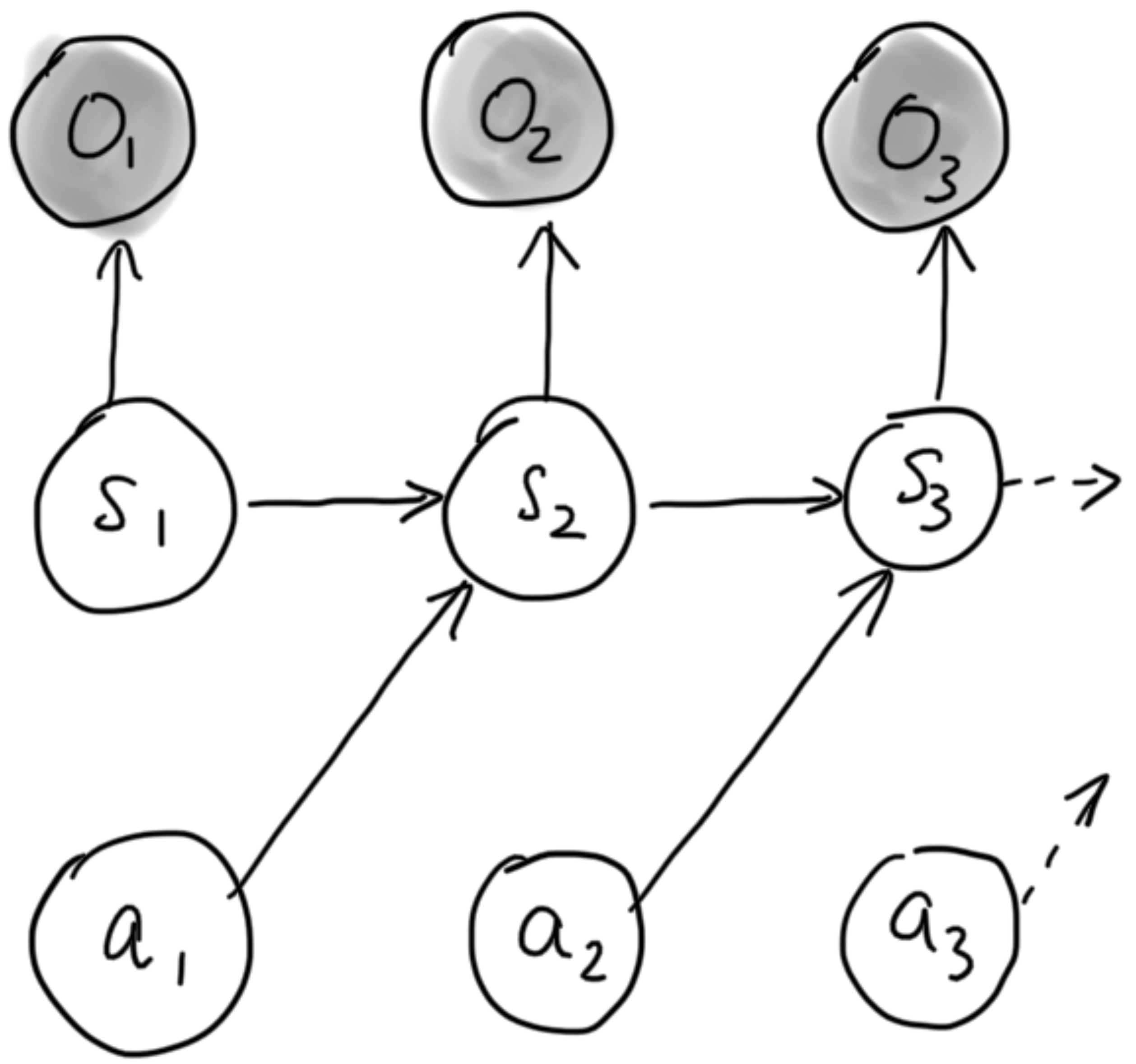


optimality

optimality variables

states

actions



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

a policy function $\pi : \mathrm{State} \to \mathrm{Dist[Action]}$ 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
```

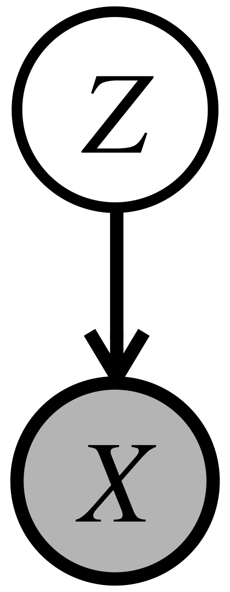
$p(a_t)$

$p(s_{t+1} \mid s_t, a_t)$

$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$

Example: Reinforcement Learning

agent trajectory

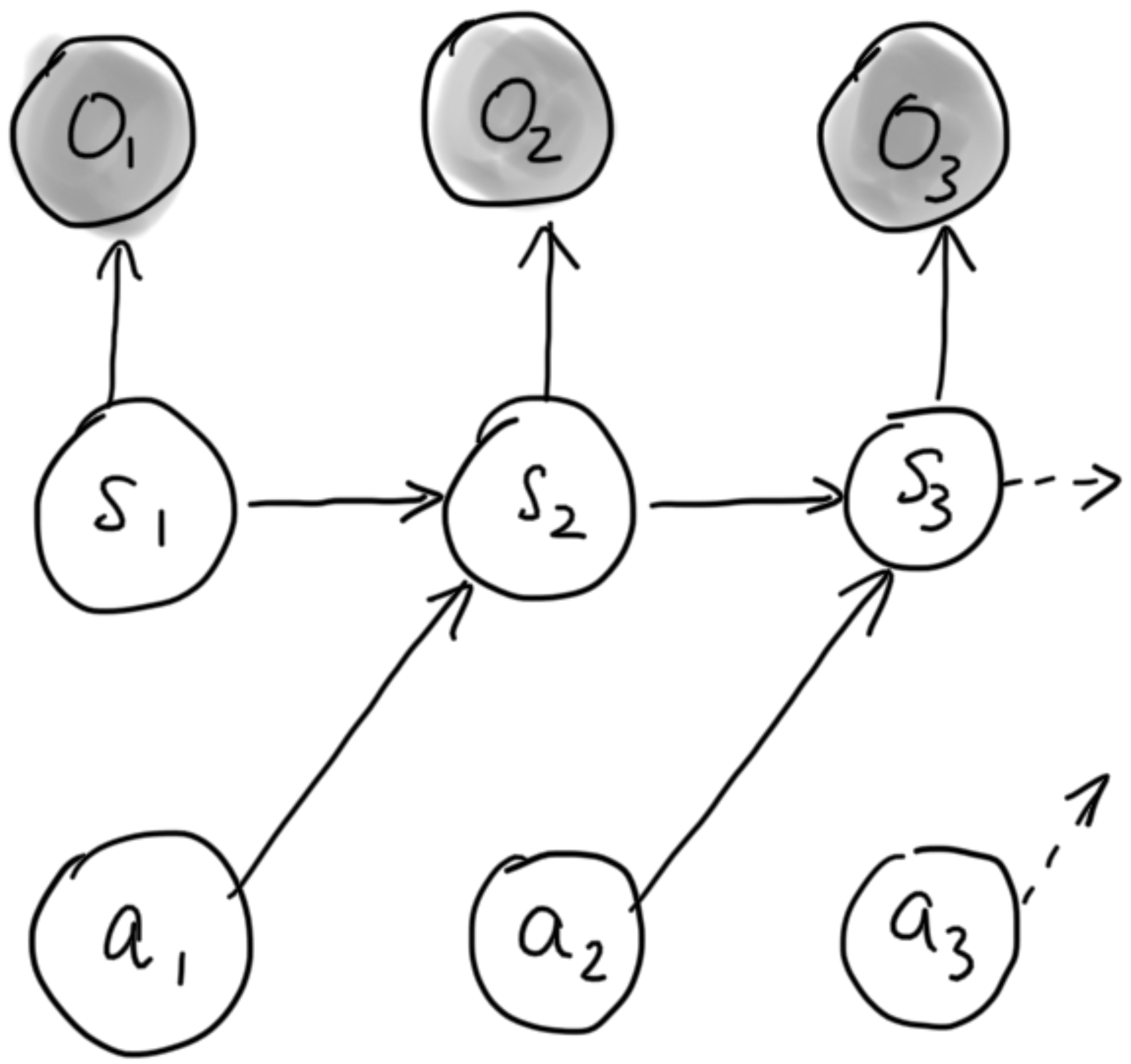


optimality

optimality variables

states

actions



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

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

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

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
```

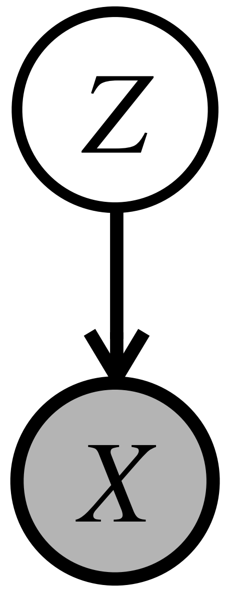
$p(a_t)$

$p(s_{t+1} \mid s_t, a_t)$

$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$

Example: Reinforcement Learning

agent trajectory

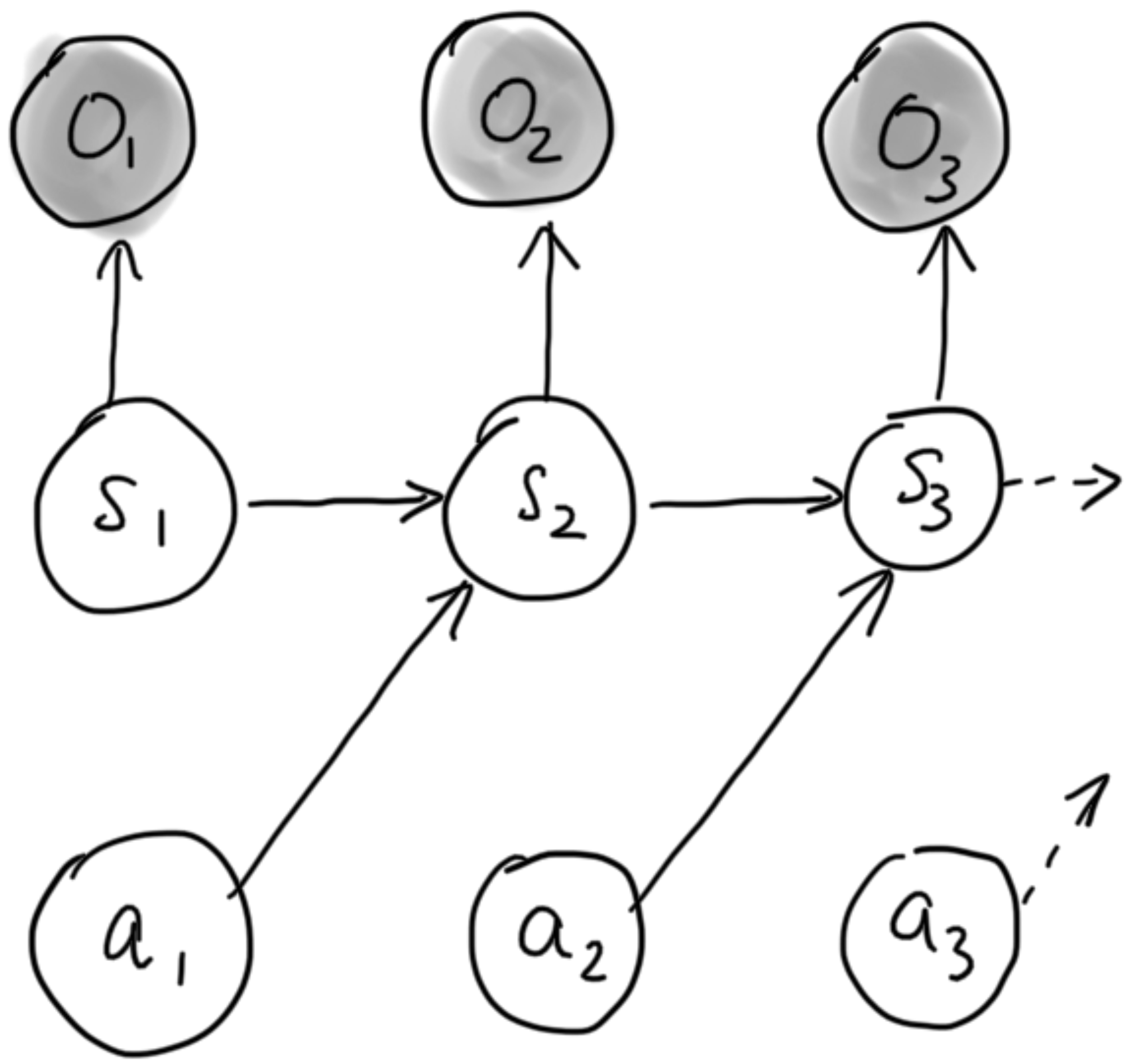


optimality

optimality variables

states

actions



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

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

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

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

Agent

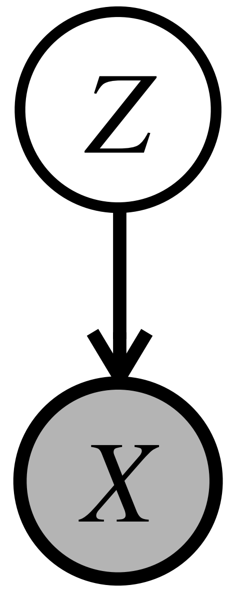
```
def MDP(state): // recursive MDP description
    if (terminal(state))
        return
    action = sample(…) // sample action from prior
    nextState = transition(state, action)
    factor(reward(nextState)) // condition on optimality
    MDP(nextState) // recurse
```

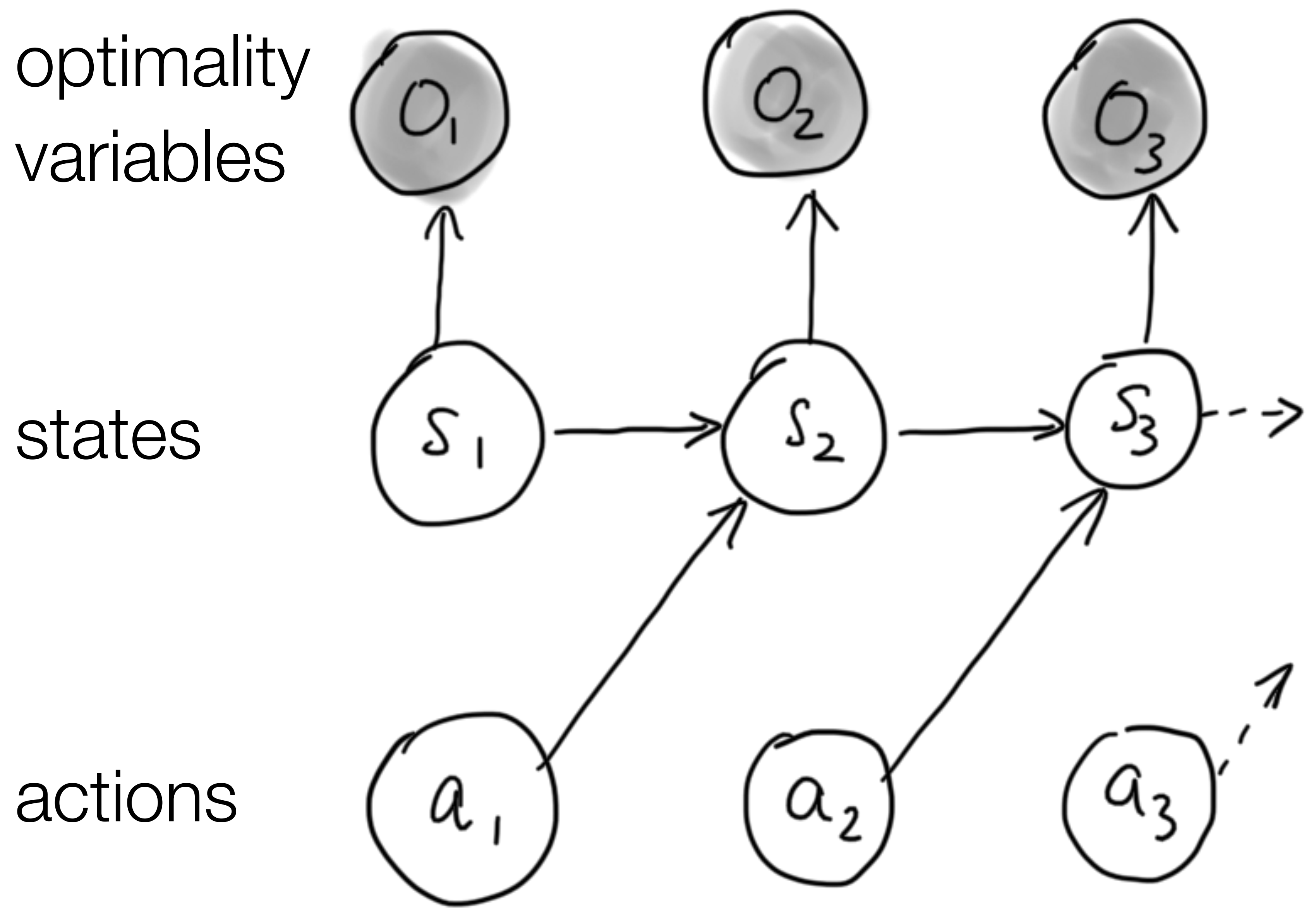
$p(a_t)$

$p(s_{t+1} \mid s_t, a_t)$

$p(O_t = 1 \mid s_t) \stackrel{def}{=} \exp(r(s_t))$

Example: Reinforcement Learning





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

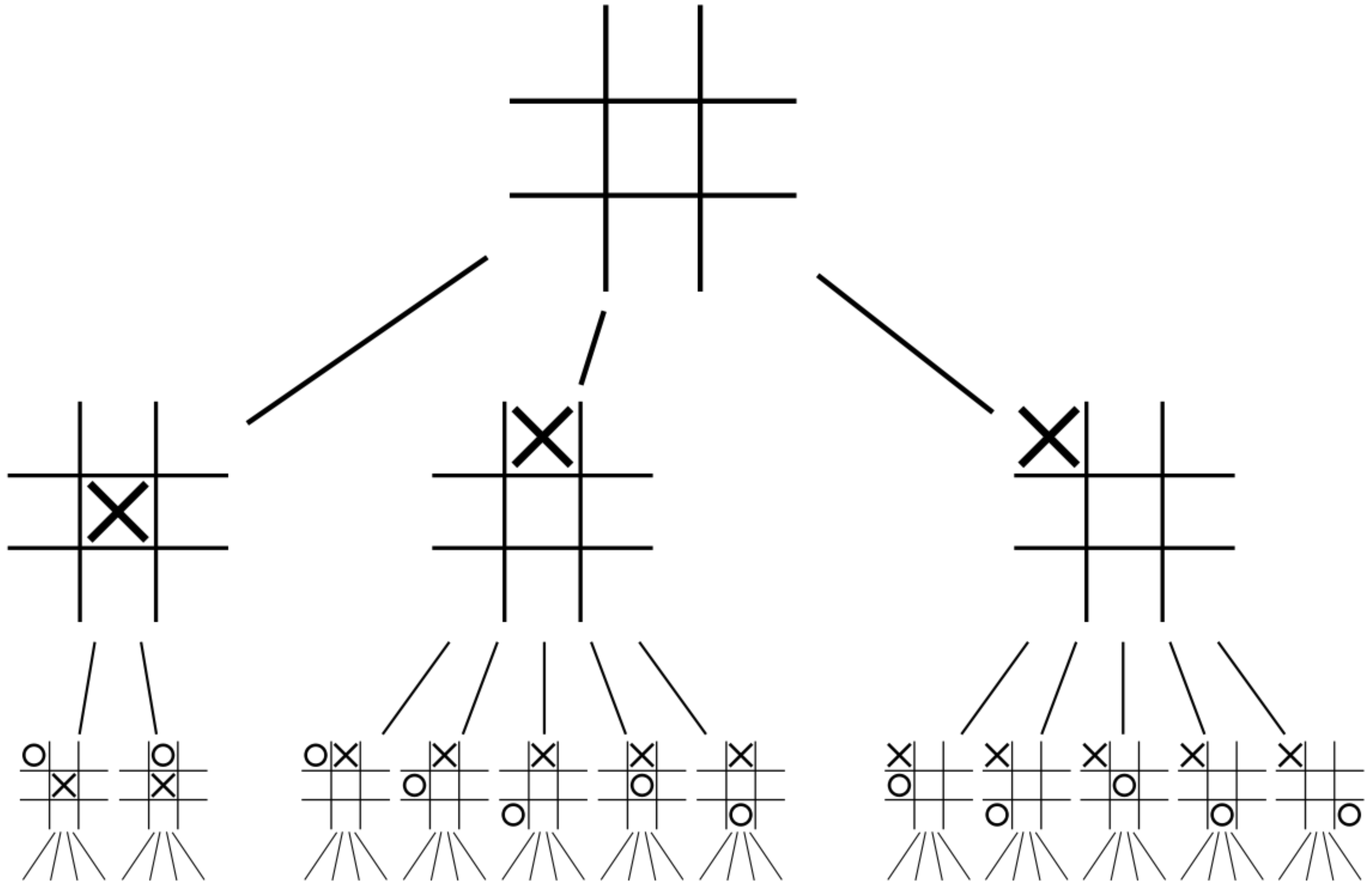
a policy function $\pi : \mathrm{State} \to \mathrm{Dist}[\mathrm{Action}]$ leading to optimal trajectory, rather than a trajectory per se

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

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

$$\min_{\phi} D_{\mathrm{KL}}\Big(q(s_1, a_1, \ldots, s_t, a_t; \phi) \,||\, p(s_1, a_1, \ldots \mid \text{optimality})\Big)$$

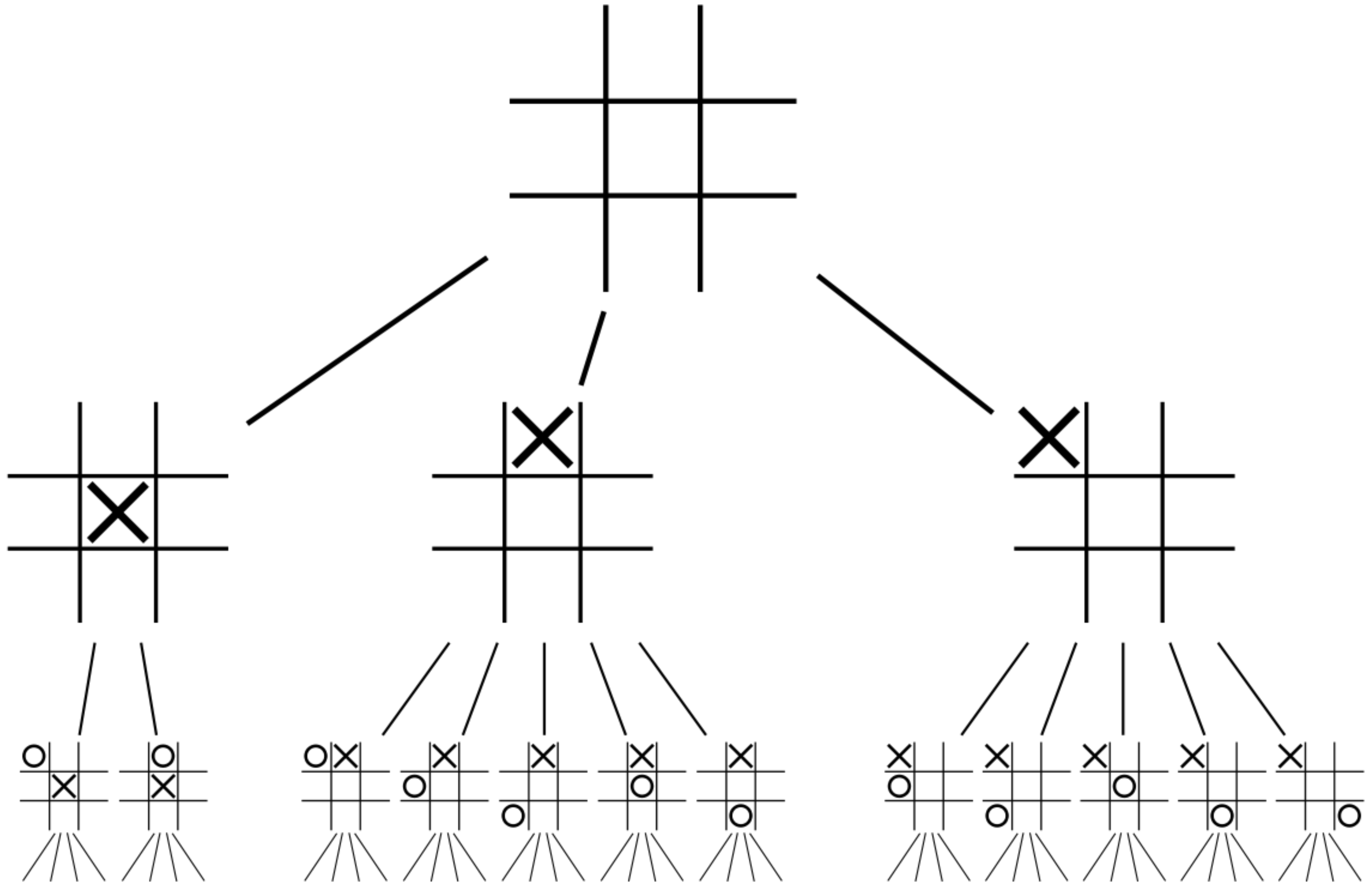
Example: Two-Player Game



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

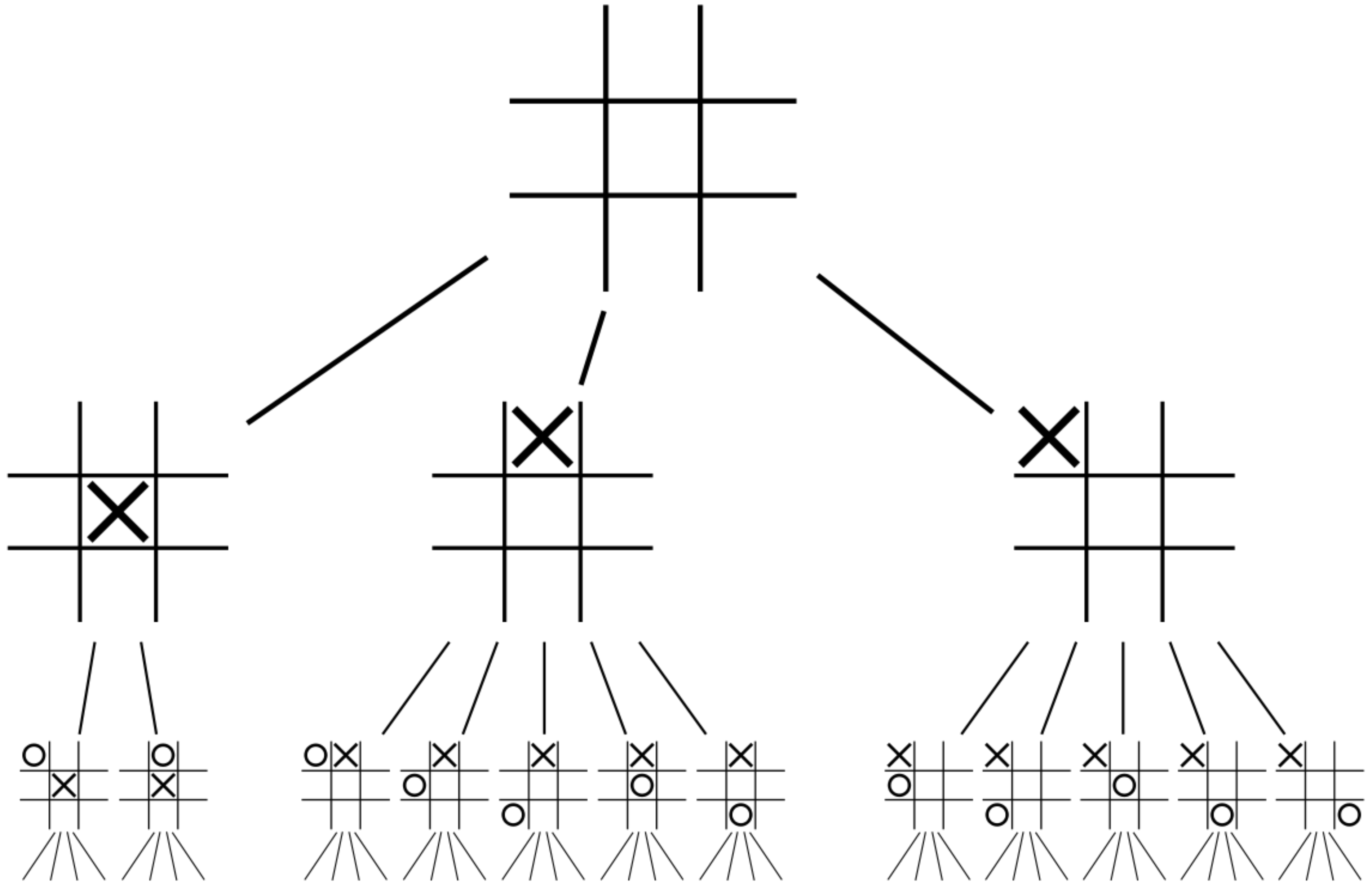
```
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};
```



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

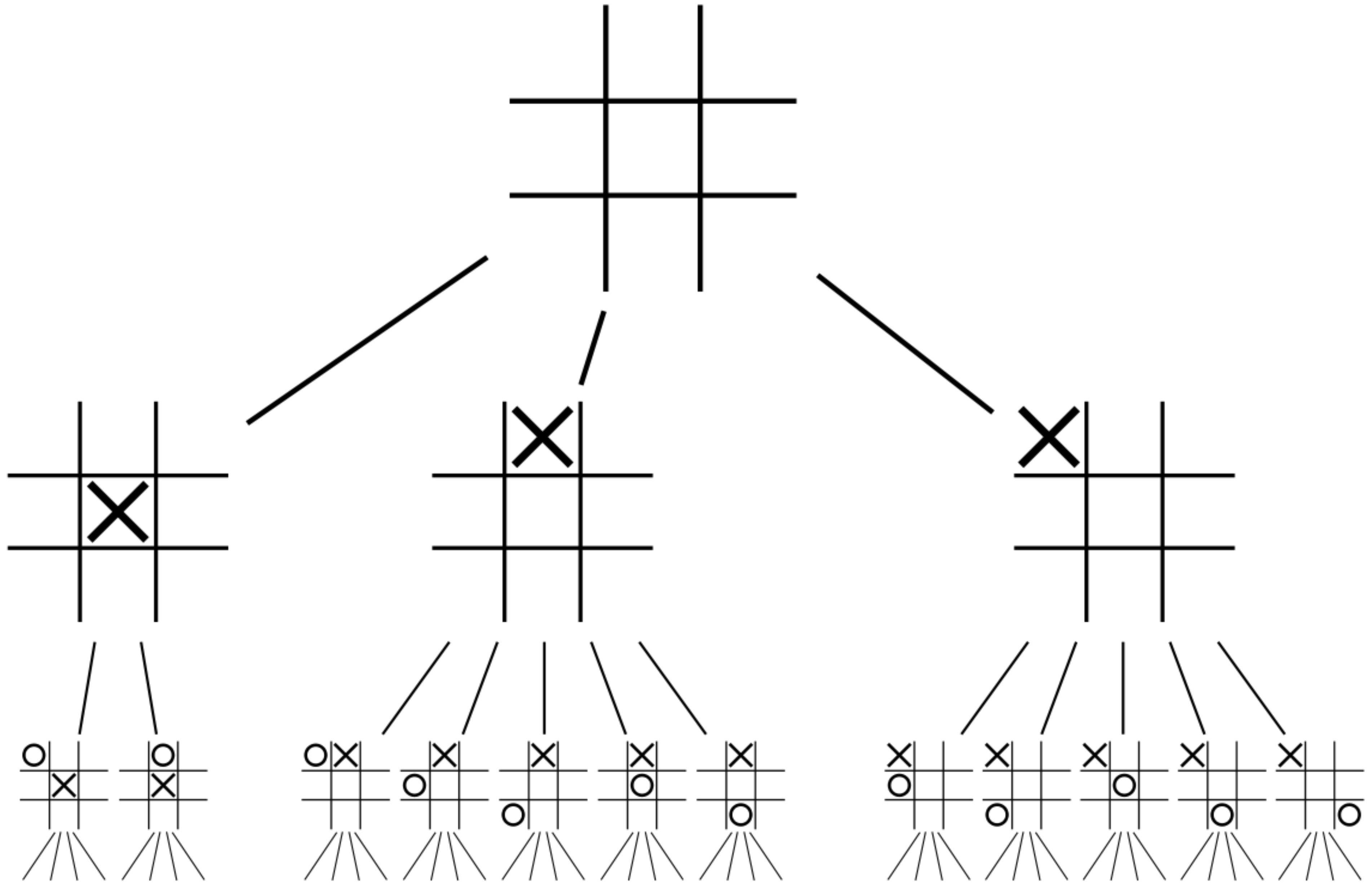
```
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

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; }
}
```



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

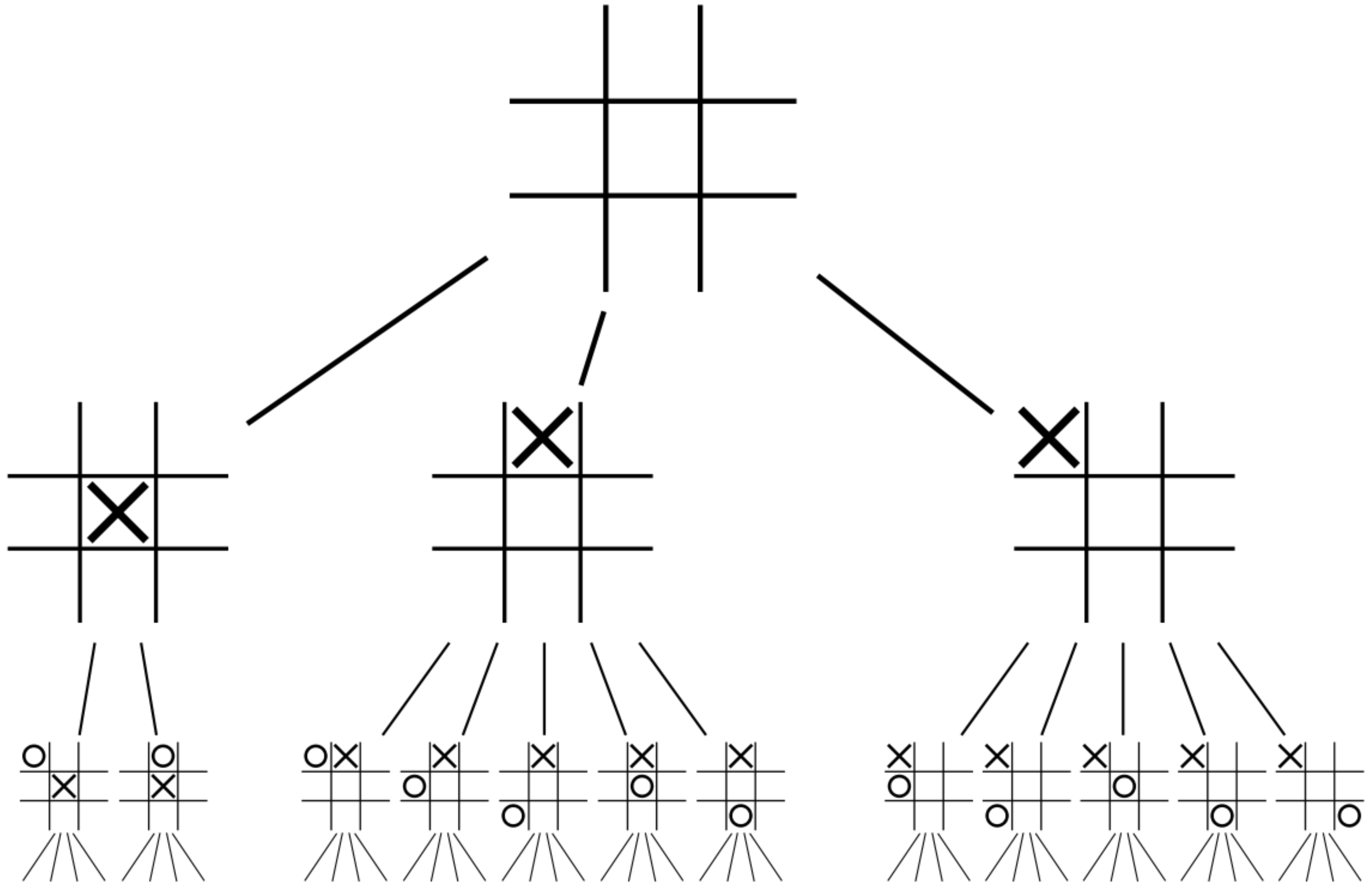
```
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

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);
        }}));
    }
};
```



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

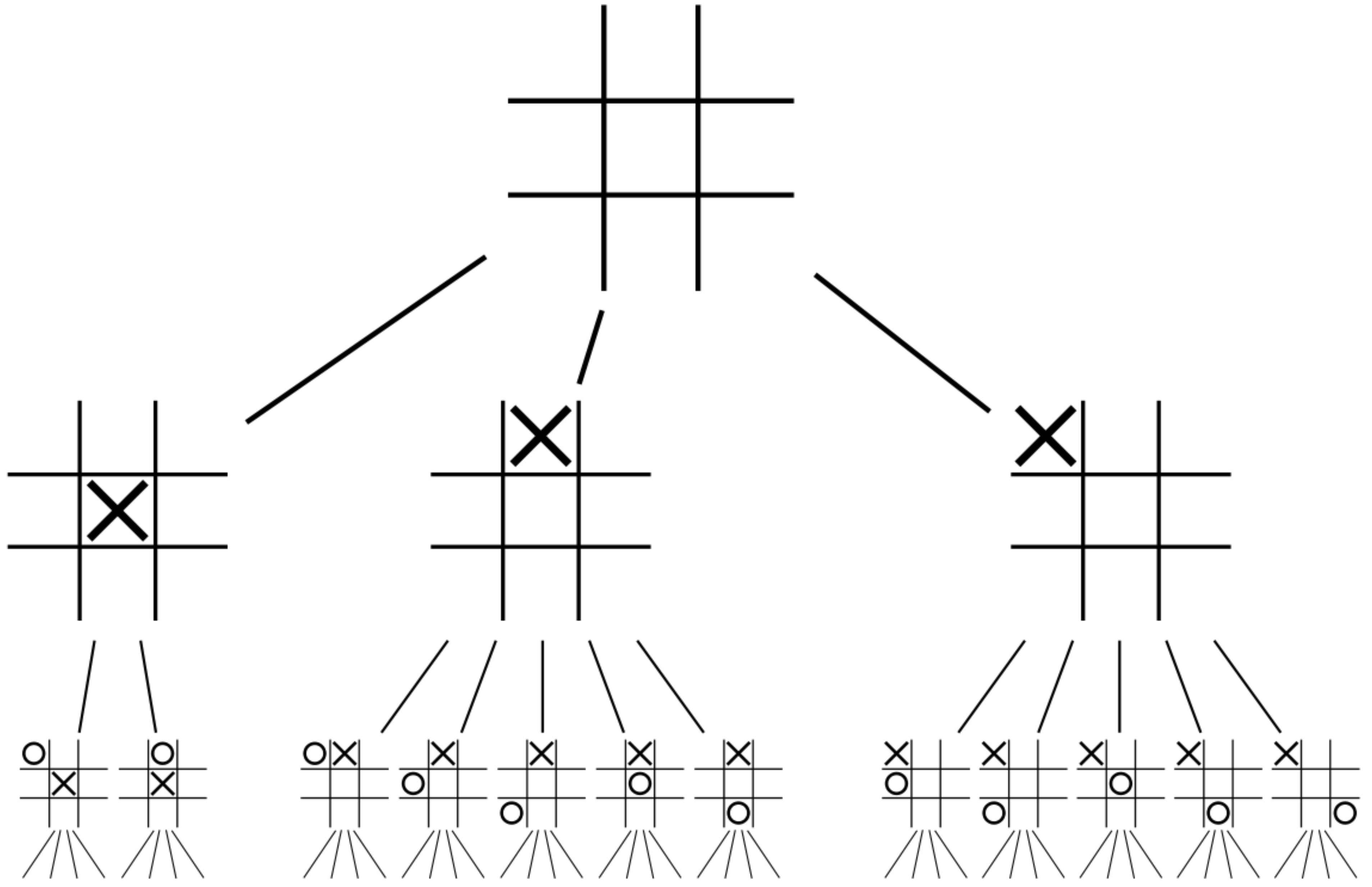
```
var act = function(state, player) {
    var move = sample(movePrior(state));
    var u = utility(state, move, player);
    factor(alpha * u);
    return move;
};

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);
        }}));
    }
};
```



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

```
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);
        }}));
    }
};
```



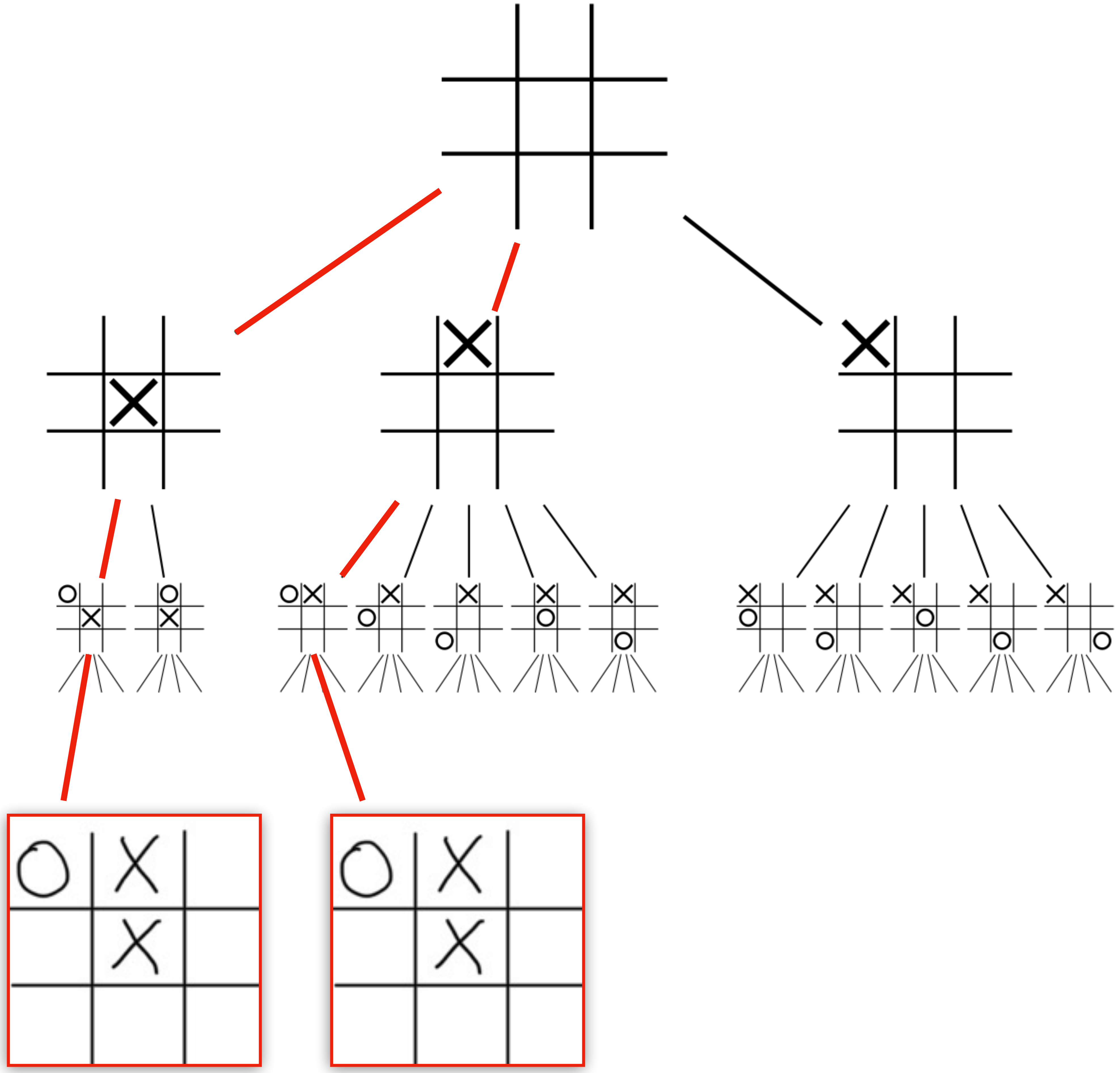
Inference inside inference => Thinking about thinking

Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

```
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);
        }}));
    }
};
```

Amortizing nested inference

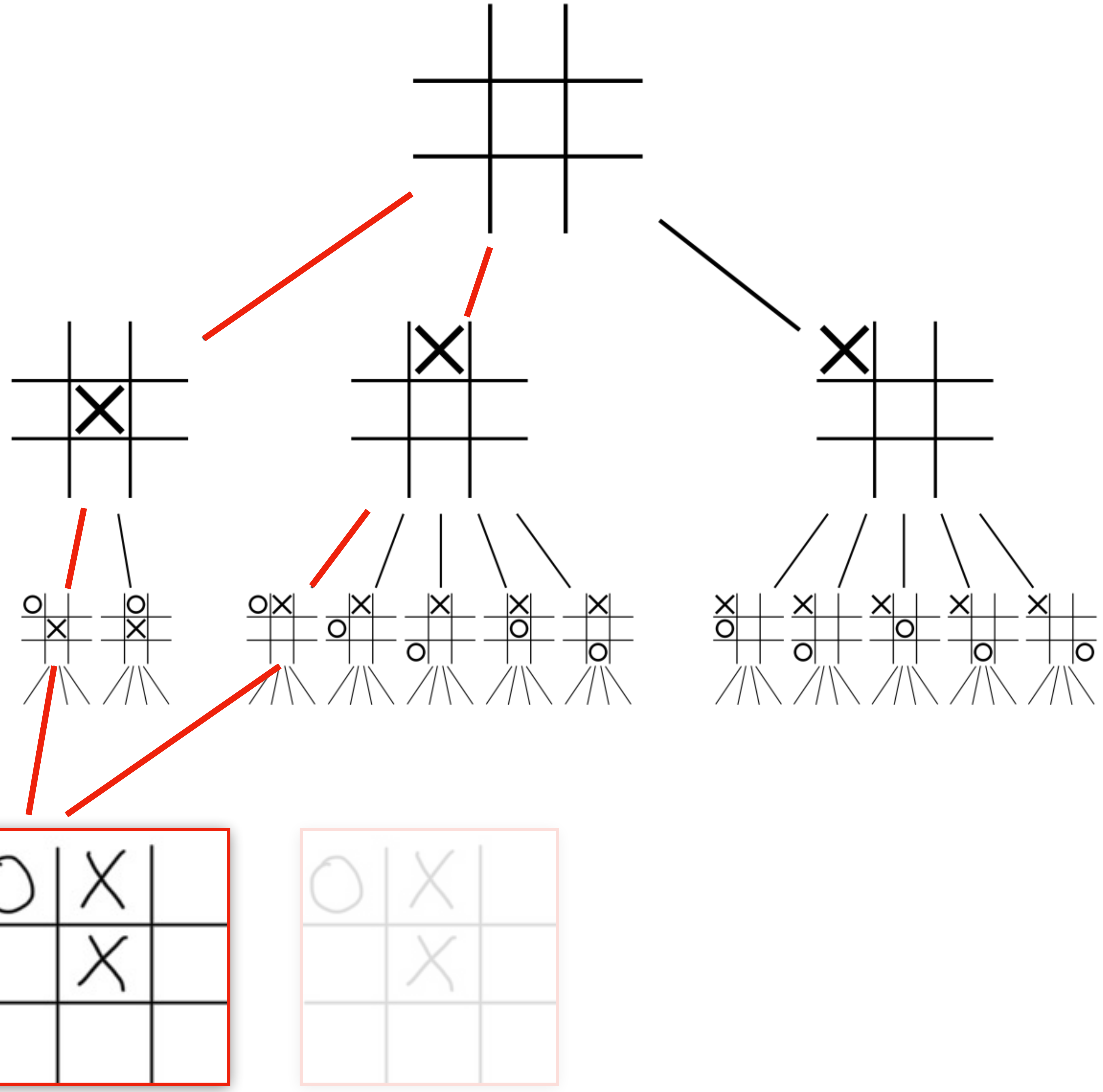


Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

Example: Two-Player Game

```
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);
        }}));
    }
};
```

Amortizing nested inference



Stuhlmüller and Goodman. Reasoning about reasoning by nested conditioning: Modeling theory of mind with probabilistic programs. CogSci. 2014.
https://agentmodels.org/chapters/7-multi-agent.html

A Glimpse into Formal Semantics

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

$$\rho_n(\ell, t, V) \stackrel{def}{=} \begin{cases} r & \langle \ell \mid t \rangle \longrightarrow_n \langle \epsilon \mid v \rangle \bullet r \text{ and } v \in V \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_n(t, V) \stackrel{def}{=} \int \rho_n(\ell, t, V)\, \mathrm{d}\ell$$

$$\mu(t, V) \stackrel{def}{=} \lim_{n \to \infty} \mu_n(t, V)$$

$$\boxed{t_1 \longrightarrow t_2}$$

[KTX] $$\frac{t_1 \longrightarrow t_2}{K[t_1] \longrightarrow K[t_2]}$$

[BETA] $$(\lambda x.\, t)\ v \longrightarrow t\,\{v/x\}$$

[OP] $$\frac{[\![\mathsf{op}_n]\!](r_1, \ldots, r_n) = r}{\mathsf{op}_n(r_1, \ldots, r_n) \longrightarrow r}$$

$$\boxed{\langle \ell_1 \mid t_1 \rangle \longrightarrow \langle \ell_2 \mid t_2 \rangle \bullet r}$$

[PURE] $$\frac{t_1 \longrightarrow t_2}{\langle \ell \mid t_1 \rangle \longrightarrow \langle \ell \mid t_2 \rangle \bullet 1}$$

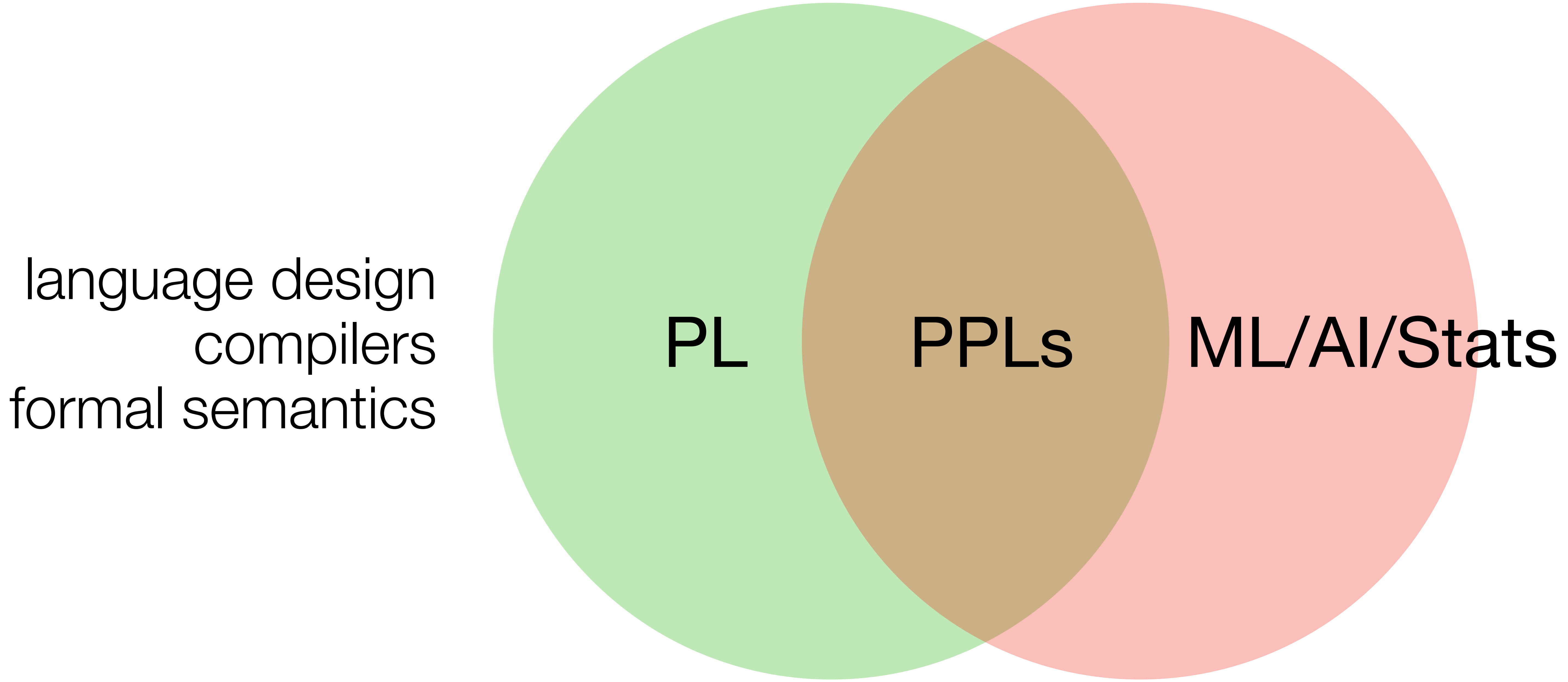
[SAMPLE] $$\frac{0 \leq r \leq 1}{\langle r, \ell \mid \textbf{sample} \rangle \longrightarrow \langle \ell \mid r \rangle \bullet 1}$$

[FACTOR] $$\frac{0 < r}{\langle \ell \mid \textbf{factor}\ r \rangle \longrightarrow \langle \ell \mid 0 \rangle \bullet r}$$

Takeaway messages

PPLs are powerful tools for probabilistic modeling and inference

Exciting area of ongoing research



CS 152 Programming Languages

Probabilistic Programming & Probabilistic Programming Languages

Yizhou Zhang
University of Waterloo