Semantics of Probabilistic Programming

Xin Zhang

Peking University

Most of the content is from "Semantics of Probabilistic Programming: A Gentle Introduction" by Fredrik Dahlqvist, Alexandra Silva, and Dexter Kozen

Recap: Problem and Motivation

- Evaluate P(Z|X) and related expectations

- Problem with exact methods
  - Curse of dimensionality
  - P(Z|X) has a complex form making expectations analytically intractable

Recap: Variational Inference

- Functional: a function that maps a function to a value

$$\mathrm{H}[p] = \int p(x) \ln p(x) \, \mathrm{d}x$$

- Variational method: find an input function that maximizes the functional

- Variational inference: find a distribution q(z) to approximate p(Z|X) so a functional is maximized

Recap: Variational Inference

$$\ln p(\mathbf{X}) = \mathcal{L}(q) + \mathrm{KL}(q\|p)$$

Between p(Z|X) and q(Z)

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$

$$\mathrm{KL}(q\|p) = -\int q(\mathbf{Z}) \ln \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} \mathrm{d}\mathbf{Z}$$

If q can be any distribution, then variational inference is precise.
But in practice, it cannot

Is the following statement right?

- Probability p(Z,X) is usually easier to evaluate compared to P(Z|X).

Recap: Sampling Methods

- Stochastic methods
- Also called Monte Carlo methods

$$\mathbb{E}[f] = \int f(\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z} \quad \Rightarrow \quad \hat{f} = \frac{1}{L}\sum_{l=1}^{L} f(\mathbf{z}^{(l)})$$

$\mathbf{z}_1, \ldots, \mathbf{z}_l$ are samples from p

Recap: Sampling Methods

- Transformation method: CDF^{-1}(uniform(0,1))

- Rejection sampling
  - A proposal distribution q(z)
  - Choose k, such that k*q(z) >= p(z), for any x
  - Sampling process:
    - Sample z_0 from q(z)
    - Sample h from uniform(0, k*q(z_0))
    - If h > p(z_0), discard it; otherwise, keep it

Is the following statement correct?

- All primitive distributions can be constructed using the inverse CDF.

Is the following statement right?

- In rejection sampling, given k, the probability whether a sample is accepted does not depend on the proposal distribution

Is the following statement correct?

- The efficiency of rejection sampling depends on the choice of the proposal distribution

Recap: Sampling Methods

- Importance sampling
  - Used to evaluate f(z) where z is from p(z)

$$E(f) = \int f(z)p(z)dz = \int f(z)\frac{p(z)}{q(z)}q(z)dz \approx \frac{1}{L}\sum_{l=1}^{L}\frac{p(z^l)}{q(z^l)}f(z^l)$$

  - How to get real samples: create a new discrete distribution using the above samples and set their probabilities using the importance weights

Recap: Sampling Methods

- Markov Chain Monte Carlo
  - A sampling method that works with a large family of distributions and high dimensions

- Workflow
  - Start with some sample z_0
  - Suppose the current sample is z^{τ} . Draw next sample z^* from $q(z \mid z^{\tau})$
  - Decide whether to accept z^*as the next state based some criteria. If accepted, $z^{\tau+1} = z^*$. Otherwise, $z^{\tau+1} = z^{\tau}$
  - Samples form a Markov chain

Recap: Sampling Methods

	Metropolis	Metropolis-Hasting
Constraints on the proposal distribution	Symmetric	None
Accepting probability	$\min(1, \frac{p(z')}{p(z)})$	$\min(1, \frac{p(z')q(z'\|z)}{p(z)q(z\|z')})$

Recap: Why MCMC works?

- Markov chain: $$p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(m)}) = p(\mathbf{z}^{(m+1)}|\mathbf{z}^{(m)}).$$

- Stationary distribution of a Markov chain: each step in the chain does not change the distribution.

- Detailed balance: $$p^\star(\mathbf{z})T(\mathbf{z}, \mathbf{z}') = p^\star(\mathbf{z}')T(\mathbf{z}', \mathbf{z})$$
  - $p^*(\boldsymbol{z})$ is a stationary distribution

- A *ergodic* Markov chain converges to the same distribution regardless the initial distribution
  - The system does not return to the same state at fixed intervals
  - The expected number of steps for returning to the same state is finite

Is the following statement right?

- The samples drawn using MCMC are independent

Is the following statement right?

- A Markov chain can have more than one stationary distribution

Use MCMC to solve the problem below

- Super optimization
  - There is a straight-line program
  - A set of test cases are given
  - The program can be modified by deleting a statement, inserting a statement from the initial program at a given place
  - Optimize the program by using the above operations

Motivations

- In order to reason about properties of a program, we need formal tools

- Example questions
  - Is the postcondition satisfied?
  - Does this program halt on all inputs?
  - Does it always halt in polynomial time?

Motivations

- In order to reason about properties of a program, we need formal tools

- Example questions
  - What is the probability that the postcondition is satisfied?
  - What is the probability that this program halts on all inputs?
  - What is the probability that it halts in polynomial time?

Motivations

- When designing a language, rigorous semantics is needed to guarantee its correctness

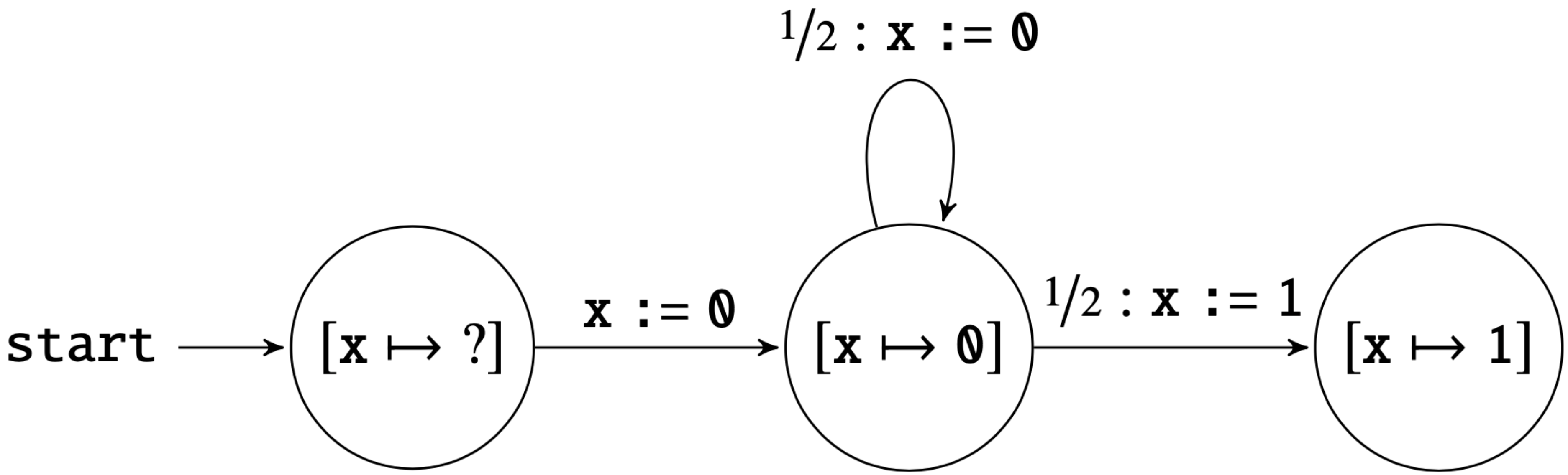
- An example that didn't have rigorous semantics: Javascript
  - https://javascriptwtf.com

Examples

We can decompose the semantics of a program into semantics of statements

```
x :=0
while x == 0 do
        x:=coin()
```

What is the probability that It runs through n iterations?
What is the expected number of iterations?
What is the probability that the program halts?



Examples

```
main{
        u:=0;
        v:=0;
        step(u,v);
        while u!=0 || v!=0 do
                step(u,v)
}

step(u,v){
        x:=coin();
        y:=coin();
        u:=u+(x-y);
        v:=v+(x+y-1)
}
```

What is the probability that the program halts?

The program is a two-dimensional random walk. According to probability theory, the probability that it returns to the origin is 1.

By relating to concepts in probabilities, we can simplify the reasoning

Examples

```
i:=0;
n:=0;
while i<1e9 do
    x:=rand();
    y:=rand();
    if (x*x+y*y) < 1 then n:=n+1;
    i:=i+1
i:=4*n/1e9;
```

What does this program compute?

How to reason about it?

Measure Theory
The mathematical foundation of probabilities and integration

Uniform(0,1) is called a *Lebesgue measure*

This Class

- Syntax of a simple imperative probabilistic language
- Operational semantics
- Measure theory & denotational semantics (brief)

A Simple Imperative Language

- Highly simplified version

- Enough to explain the core concepts

Syntax

- Deterministic terms (expressions)
- Terms (Deterministic + Probabilistic)
- Tests (expression that evaluate to Booleans)
- Programs

Syntax – Deterministic Terms

(i) Deterministic terms:

$$
\begin{aligned}
d ::= {} & a && a \in \mathbb{R},\ \text{constants} \\
\mid {} & x && x \in \text{Var},\ \text{a countable set of variables} \\
\mid {} & d \ \mathbf{op} \ d && \mathbf{op} \in \{+, -, *, \div\}
\end{aligned}
$$

Syntax - Terms

(ii) Terms:

$t ::= d$	d a deterministic term
$\mid$ `coin()` $\mid$ `rand()`	sample in $\{0, 1\}$ and $[0, 1]$, respectively
$\mid t$ op t	op $\in \{+, -, *, \div\}$

Syntax - Tests

(iii) Tests:

$b ::= \texttt{true} \mid \texttt{false}$	
$\mid d == d \mid d < d \mid d > d$	comparison of deterministic terms
$\mid b \mathbin{\&\&} b \mid b \mathbin{\texttt{\|\|}} b \mid \texttt{!}b$	Boolean combinations of tests

Syntax - Program

(iv) Programs:

$e ::= \texttt{skip}$	
$\mid x \texttt{ := } t$	assignment
$\mid e \texttt{ ; } e$	sequential composition
$\mid \texttt{if } b \texttt{ then } e \texttt{ else } e$	conditional
$\mid \texttt{while } b \texttt{ do } e$	`while` loop

Syntax - Example Program

```
if coin() == 1 then
        x := rand() * 5
else
        x := 6
if x > 4.5 then
        y := coin() + 2
else
        y := 100
```

Operational Semantics

- Model the step-by-step executions of a program on a machine

- Tracks the memory-state
  - Values assigned to each variable
  - Values of each random number generator
  - A stack of instructions

Random Number Generators

- Modeled as infinite streams of numbers:
  - coin(): $m_0 m_1$ … are i.i.d from Bernoulli(0.5)
  - rand(): $p_0 p_1$ … are i.i.d from uniform(0, 1)
- When invoking the generator, a number is taken from the stream
  - Pseudo-random generators

Operational Semantics: Machine States

- A memory-state is a triple (s, m, p)
  - A store $s: n \to R$, where there are n variables in the program
  - $m \in \{0,1\}^{\omega}$ is the current stream of available random Boolean values
  - $p \in [0,1]^{\omega}$ is the current stream of available random real values
- A machine-state is a 4-tuple (e, s, m, p)
  - e corresponds to a stack of instructions
  - (s, m, p) is a memory-state

Machine States: Example

$(\text{e}, \{x \rightarrow \perp\}, 1001011\ldots, 0.2\ 0.5\ 0.9\ 0.21\ldots)$

if coin() == 1 then

$(\text{x := rand() * 5}, \{x \rightarrow \perp\}, 001011\ldots, 0.2\ 0.5\ 0.9\ 0.21\ldots)$

x := rand() * 5

$(\text{skip}, \{x \rightarrow 1\}, 001011\ldots, 0.5\ 0.9\ 0.21\ldots)$

else

x := 6

Operational Semantics: Introduction

- We now talk about how a program modifies the machine state

- Type of the operational semantics

$$(e, s, m, p) \to (e', s', m', p')$$

- Before talking about the reduction, we need to define semantics of terms and tests

Semantics of Terms

$$[\![t]\!] : \quad \boldsymbol{R}^n \times \boldsymbol{N}^\omega \times \boldsymbol{R}^\omega \to \boldsymbol{R} \times \boldsymbol{N}^\omega \times \boldsymbol{R}^\omega$$

$$[\![r]\!] : (s, m, p) \mapsto (r, m, p)$$

$$[\![x_i]\!] : (s, m, p) \mapsto (s(i), m, p)$$

$$[\![\texttt{coin()}]\!] : (s, m, p) \mapsto (\mathsf{hd}\, m, \mathsf{tl}\, m, p)$$

$$[\![\texttt{rand()}]\!] : (s, m, p) \mapsto (\mathsf{hd}\, p, m, \mathsf{tl}\, p)$$

$$[\![t_1 \ \mathsf{op}\ t_2]\!] : (s, m, p) \mapsto \begin{array}{l} \texttt{let}\ (a_1, m', p') = [\![t_1]\!](s, m, p)\ \texttt{in} \\ \texttt{let}\ (a_2, m'', p'') = [\![t_2]\!](s, m', p')\ \texttt{in} \\ (a_1\ \mathsf{op}\ a_2, m'', p'') \end{array}$$

$opn \in \{+, 0, *, \div\}$ $hd(m_1 m_2, \ldots) = m_1$

Semantics of Tests

$$[\![b]\!]: \quad \boldsymbol{R}^n \times \boldsymbol{N}^\omega \times \boldsymbol{R}^\omega \to \{\boldsymbol{true}, \boldsymbol{false}\}$$

$$[\![t_1 == t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) = [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![t_1 < t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) < [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![t_1 > t_2]\!] : (s,m,p) \mapsto \begin{cases} \texttt{true} & \text{if } [\![t_1]\!](s,m,p) > [\![t_2]\!](s,m,p) \\ \texttt{false} & \text{otherwise} \end{cases}$$

$$[\![b_1 \,\&\&\, b_2]\!] : (s,m,p) \mapsto [\![b_1]\!](s,m,p) \land [\![b_2]\!](s,m,p)$$

$$[\![b_1 \;||\; b_2]\!] : (s,m,p) \mapsto [\![b_1]\!](s,m,p) \lor [\![b_2]\!](s,m,p)$$

$$[\![\,!\,b]\!] : (s,m,p) \mapsto \neg[\![b]\!](s,m,p)$$

Operational Semantics: Reduction

Assignment:

$$\frac{[\![t]\!](s,m,p) = (a,m',p')}{(x_i \;\texttt{:=}\; t,s,m,p) \longrightarrow (\texttt{skip}, s[i \mapsto a],m',p')}$$

Sequential composition:

$$\frac{(e_1,s,m,p) \longrightarrow (e_1',s',m',p')}{(e_1 \;\texttt{;}\; e_2,s,m,p) \longrightarrow (e_1' \;\texttt{;}\; e_2,s',m',p')}$$

$$\frac{}{(\texttt{skip} \;\texttt{;}\; e,s,m,p) \longrightarrow (e,s,m,p)}$$

Operational Semantics: Reduction

Conditional:

$$\frac{[\![b]\!](s,m,p) = \texttt{true}}{(\texttt{if}\ b\ \texttt{then}\ e_1\ \texttt{else}\ e_2, s, m, p) \longrightarrow (e_1, s, m, p)}$$

$$\frac{[\![b]\!](s,m,p) = \texttt{false}}{(\texttt{if}\ b\ \texttt{then}\ e_1\ \texttt{else}\ e_2, s, m, p) \longrightarrow (e_2, s, m, p)}$$

`while` *loops:*

$$\frac{}{(\texttt{while}\ b\ \texttt{do}\ e, s, m, p) \longrightarrow (\texttt{if}\ b\ \texttt{then}\ (e\ ;\ \texttt{while}\ b\ \texttt{do}\ e)\ \texttt{else}\ \texttt{skip}, s, m, p)}$$

Operational Semantics: Reduction

Reflexive-transitive closure:

$$\frac{}{(e, s, m, p) \xrightarrow{*} (e, s, m, p)}$$

$$\frac{(e_1, s_1, m_1, p_1) \longrightarrow (e_2, s_2, m_2, p_2)}{(e_1, s_1, m_1, p_1) \xrightarrow{*} (e_2, s_2, m_2, p_2)}$$

$$\frac{(e_1, s_1, m_1, p_1) \xrightarrow{*} (e_2, s_2, m_2, p_2) \qquad (e_2, s_2, m_2, p_2) \xrightarrow{*} (e_3, s_3, m_3, p_3)}{(e_1, s_1, m_1, p_1) \xrightarrow{*} (e_3, s_3, m_3, p_3)}$$

Operational Semantics: Termination

- A program e terminates from (s, m, p) if

$$(e, s, m, p) \xrightarrow{*} (\texttt{skip}, s', m', p').$$

- We say e diverges from (s, m, p) if it does not terminate

Operational Semantics: Examples

x :=0
while x == 0 do
    x:=coin()

What is the probability that the program halts?

$$\dfrac{\dfrac{\dfrac{(x := 0, s, m, p) \longrightarrow (\mathbf{skip}, s[x \mapsto 0], m, p)}{(x := 0 ; e, s, m, p) \longrightarrow (\mathbf{skip} ; e, s[x \mapsto 0], m, p)}}{(x := 0 ; e, s, m, p) \stackrel{*}{\longrightarrow} (\mathbf{skip} ; e, s[x \mapsto 0], m, p)} \quad \dfrac{(\mathbf{skip} ; e, s[x \mapsto 0], m, p) \longrightarrow (e, s[x \mapsto 0], m, p)}{(\mathbf{skip} ; e, s[x \mapsto 0], m, p) \stackrel{*}{\longrightarrow} (e, s[x \mapsto 0], m, p)}}{(x := 0 ; e, s, m, p) \stackrel{*}{\longrightarrow} (e, s[x \mapsto 0], m, p)}$$

Operational Semantics: Examples

```
x :=0
while x == 0 do
      x:=coin()
```

What is the probability that the program halts?

$$(x := 0 \; ; \; e, s, m, p) \xrightarrow{*} (e, s[x \mapsto 0], m, p)$$

$$(e, s[x \mapsto 0], m, p) \xrightarrow{*} (x := \texttt{coin()} \; ; \; e, s[x \mapsto 0], m, p)$$

$$\frac{}{(\texttt{while}\; b \;\texttt{do}\; e, s, m, p) \longrightarrow (\texttt{if}\; b \;\texttt{then}\; (e \; ; \; \texttt{while}\; b \;\texttt{do}\; e) \;\texttt{else skip}, s, m, p)}$$

$$\frac{[\![b]\!](s, m, p) = \texttt{true}}{(\texttt{if}\; b \;\texttt{then}\; e_1 \;\texttt{else}\; e_2, s, m, p) \longrightarrow (e_1, s, m, p)}$$

Operational Semantics: Examples

```
x :=0
while x == 0 do
        x:=coin()
```

What is the probability that the program halts?

$$(x := 0 \,;\, e, s, m, p) \xrightarrow{*} (e, s[x \mapsto 0], m, p)$$

$$(e, s[x \mapsto 0], m, p) \xrightarrow{*} (x := \texttt{coin()} \,;\, e, s[x \mapsto 0], m, p)$$

$$(x := \texttt{coin()} \,;\, e, s[x \mapsto 0], m, p) \xrightarrow{*} (e, [s \mapsto \mathsf{hd}\, m], \mathsf{tl}\, m, p).$$

$$hd(m_1 m_2 \ldots) = m_1$$
$$tl(m_1 m_2 \ldots) = m_2 \ldots$$

The loop continues until it reaches m inf the form of $1m'$

$$(e, s[x \mapsto 1], m', p) \xrightarrow{*} (\texttt{skip}, s[x \mapsto 1], m', p)$$

$$(x := 0 \,;\, e, s, m, p) \xrightarrow{*} (\texttt{skip}, s[x \mapsto 1], m', p)$$

Operational Semantics: Examples

$$\mathbb{P}\left[\exists m' \, (x \texttt{ := } 0 \texttt{ ; } e, s, m, p) \xrightarrow{*} (\texttt{skip}, s[x \mapsto 1], m', p)\right]$$

$$= \mathbb{P}\left[\exists k \geq 0 \; \exists m' \; m = 0^k 1 m'\right]$$

$$= \sum_{k=1}^{\infty} 2^{-k} = 1$$

Operational Semantics: Examples

```
main{
        u:=0;
        v:=0;
        step(u,v);
        while u!=0 || v!=0 do
                step(u,v)
}

step(u,v){
        x:=coin();
        y:=coin();
        u:=u+(x-y);
        v:=v+(x+y-1)
}
```

What is the probability that the program halts?

$$(\texttt{step}, s, 00m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,-1), (\texttt{x},\texttt{y}) \mapsto (0,0)], m, p)$$

$$(\texttt{step}, s, 01m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (-1,0), (\texttt{x},\texttt{y}) \mapsto (0,1)], m, p)$$

$$(\texttt{step}, s, 10m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (1,0), (\texttt{x},\texttt{y}) \mapsto (1,0)], m, p)$$

$$(\texttt{step}, s, 11m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,1), (\texttt{x},\texttt{y}) \mapsto (1,1)], m, p)$$

Operational Semantics: Examples

```
main{
        u:=0;
        v:=0;
        step(u,v);
        while u!=0 || v!=0 do
                step(u,v)
}

step(u,v){
        x:=coin();
        y:=coin();
        u:=u+(x-y);
        v:=v+(x+y-1)
}
```

What is the probability that the program halts?

We define i.i.d variables X_1, X_2 ... on Z^2 such that

$$X_i \in \{(0,1),(0,-1),(1,0),(-1,0)\}$$

$$S_n = \sum_{i=1}^{n} X_i$$

$$(\texttt{main}, s, m, p) \xrightarrow{*}$$

$$(\texttt{while !(u == 0) || !(v == 0) do step(u,v)}, s[(\texttt{u},\texttt{v}) \mapsto (i,j)], \mathsf{tl}^4(m), p)$$

Operational Semantics: Examples

```
main{
        u:=0;
        v:=0;
        step(u,v);
        while u!=0 || v!=0 do
                step(u,v)
}

step(u,v){
        x:=coin();
        y:=coin();
        u:=u+(x-y);
        v:=v+(x+y-1)
}
```

What is the probability that the program halts?

The program halts if $\exists n. S_{2n} = (0,0)$

$$(\texttt{main}, s, m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,0)], \mathsf{tl}^{4n}(m), p).$$

$$\mathbb{P}\left[\exists n\, (\texttt{main}, s, m, p) \xrightarrow{*} (\texttt{skip}, s[(\texttt{u},\texttt{v}) \mapsto (0,0)], \mathsf{tl}^{4n}(m), p)\right]$$

$$= \mathbb{P}\left[\bigvee_{n=0}^{\infty} S_{2n} = (0,0)\right]$$

Operational Semantics: Examples

```
main{
        u:=0;
        v:=0;
        step(u,v);
        while u!=0 || v!=0 do
                step(u,v)
}

step(u,v){
        x:=coin();
        y:=coin();
        u:=u+(x-y);
        v:=v+(x+y-1)
}
```

What is the probability that the program halts?

$$\mathbb{P}[S_{2n} = (0,0)] = 4^{-2n} \sum_{m=0}^{n} \frac{(2n)!}{m!m!(n-m)!(n-m)!}$$

$$= 4^{-2n} \binom{2n}{n} \sum_{m=0}^{n} \binom{n}{m}^2$$

$$= 4^{-2n} \binom{2n}{n}^2.$$

Operational Semantics: Examples

```
i:=0;
n:=0;
while i<1e9 do
        x:=rand();
        y:=rand();
        if (x*x+y*y) < 1
                then n:=n+1;
        i:=i+1
i:=4*n/1e9;
```

Given $\epsilon > 0$, what is $P(|\mathtt{i} - \pi| \leq \epsilon)$?

$$(\mathbf{prog}, s, m, p) \xrightarrow{*} (\mathbf{skip}, s[\mathbf{i} \mapsto 4n/N, \mathbf{n} \mapsto n, \ldots], m, \mathsf{tl}^{2N}(p))$$

n/N is the expectation of

$$Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$$

Operational Semantics: Examples

```
i:=0;
n:=0;
while i<1e9 do
        x:=rand();
        y:=rand();
        if (x*x+y*y) < 1
                then n:=n+1;
        i:=i+1
i:=4*n/1e9;
```

Given $\epsilon > 0$, what is $\mathrm{P}(|\mathrm{i} - \pi| \leq \epsilon)$?

n/N is the expectation of $Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$

$$\mathbb{P}\left[X^2 \leq t\right] = \mathbb{P}\left[X \leq \sqrt{t}\right] = \int_0^{\sqrt{t}} \mathbb{1}_{[0,1]}(x)\, dx = \sqrt{t}$$

$$f(t) = \frac{\partial \mathbb{P}\left[X^2 \leq t\right]}{\partial t} = \frac{1}{2\sqrt{t}} \mathbb{1}_{[0,1]}(t)$$

Operational Semantics: Examples

```
i:=0;
n:=0;
while i<1e9 do
        x:=rand();
        y:=rand();
        if (x*x+y*y) < 1
                then n:=n+1;
        i:=i+1
i:=4*n/1e9;
```

Given $\epsilon > 0$, what is $\mathrm{P}(|\mathrm{i} - \pi| \leq \epsilon)$?

n/N is the expectation of $Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$

The density of $X^2 + Y^2$ is

$$(f * f)(t) = \int_{-\infty}^{\infty} \frac{1}{2\sqrt{x}} \mathbb{1}_{[0,1]}(x) \frac{1}{2\sqrt{t-x}} \mathbb{1}_{[0,1]}(t-x)\, dx$$

$$= \begin{cases} \int_0^t \frac{1}{4\sqrt{x}\sqrt{t-x}}\, dx & \text{if } 0 \leq t \leq 1 \\ \int_{t-1}^1 \frac{1}{4\sqrt{x}\sqrt{t-x}}\, dx & \text{if } 1 < t \leq 2 \end{cases}$$

Operational Semantics: Examples

```
i:=0;
n:=0;
while i<1e9 do
        x:=rand();
        y:=rand();
        if (x*x+y*y) < 1
                then n:=n+1;
        i:=i+1
i:=4*n/1e9;
```

Given $\epsilon > 0$, what is $\mathrm{P}(|\mathrm{i} - \pi| \le \epsilon)$?

n/N is the expectation of $Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$

$\exp(Z)$ is

$$\int_0^t \frac{1}{4\sqrt{x}\sqrt{t-x}}\,dx = \int_0^1 \frac{1}{2\sqrt{1-u^2}}\,du = \frac{1}{2}(\sin^{-1}(1) - \sin^{-1}(0)) = \frac{\pi}{4}.$$

$$\mathbb{P}\left[X^2 + Y^2 \le 1\right] = \int_0^1 (f * f)(t)\,dt = \int_0^1 \frac{\pi}{4}\,dt = \frac{\pi}{4}.$$

Operational Semantics: Examples

```
i:=0;
n:=0;
while i<1e9 do
        x:=rand();
        y:=rand();
        if (x*x+y*y) < 1
                then n:=n+1;
        i:=i+1
i:=4*n/1e9;
```

Given $\epsilon > 0$, what is $P(|i - \pi| \le \epsilon)$?

n/N is the expectation of $Z = \begin{cases} 1 & \text{if } X^2 + Y^2 < 1 \\ 0 & \text{else} \end{cases}$

$$\mathbb{P}\left[X^2 + Y^2 \le 1\right] = \int_0^1 (f * f)(t)\, dt = \int_0^1 \frac{\pi}{4}\, dt = \frac{\pi}{4}.$$

$$\mathbb{P}\left[\left|\frac{n}{N} - \frac{\pi}{4}\right| > \varepsilon\right] \le \frac{\sigma^2}{N\varepsilon^2}.$$ Where $\sigma^2 = \frac{\pi}{4} - \left(\frac{\pi}{4}\right)^2$

Chebyshev's inequality

This Class

- Syntax of a simple imperative probabilistic language
- Operational semantics
- **Measure theory & denotational semantics (brief)**

Denotational vs. Operational Semantics

- Consider x := coin(), in operational semantics:

$$(\mathbf{x} := \mathtt{coin()}, s, m, p) \longrightarrow (\mathbf{skip}, s[\mathbf{x} \mapsto \mathtt{0}], \mathsf{tl}\ m, p)$$

$$(\mathbf{x} := \mathtt{coin()}, s, m, p) \longrightarrow (\mathbf{skip}, s[\mathbf{x} \mapsto \mathtt{1}], \mathsf{tl}\ m, p)$$

- Denotational semantics:
  - Model all possible executions together
  - States: probability distribution over memory states
  - $\frac{1}{2}s[x \mapsto 0] + \frac{1}{2}s[x \mapsto 1]$

Denotational Semantics: Introduction

- State s can be identified with the Dirac measure σ_s, then the semantics of x:=coin() can be viewed as $\sigma_s \to \frac{1}{2} s[x \mapsto 0] + \frac{1}{2} s[x \mapsto 1]$

- In general, a program is interpreted as an operator mapping probability distributions to (sub)probability distributions

Denotational Semantics: Definition

- Assume there are n real variables, then a state is a distribution on R^n

- A program $e: MR^n \rightarrow MR^n$
  - An operator called a state transformer

Measure Theory

- Measures: generalization of concepts like length, area, or volume

Measure Example: Length

- What subsets of R can meaningfully be assigned a length?

- What properties should the length function l satisfy?

Measure Example: Length

$$\ell([a_1,b_1] \cup [a_2,b_2]) = \ell([a_1,b_1]) + \ell([a_2,b_2]) = (b_1 - a_1) + (b_2 - a_2).$$ $b_1 < a_2$

$$\ell\left(\bigcup_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} \ell(A_i).$$ $A_i\ and\ A_j\ are\ disjoined\ .l\ is\ called\ additive$

$$\ell\left(\bigcup_{i=0}^{\infty} A_i\right) = \sum_{i=0}^{\infty} \ell(A_i).$$ $A_i\ and\ A_j\ are\ disjoined\ .The\ set\ is\ countable.$
$l\ is\ called\ countably\ additive\ or\ \sigma - additive$

$l(R) = \infty$, but we are only going to talk about finite measures

$$\ell(B \setminus A) = \ell(B) - \ell(A)$$ Domain should be closed under complementation

Measure Example: Length

- Can we extend the domain of length l to all subsets of R?

- No. Counterexample: Vitali sets
  - $V \subseteq [0,1]$, such that for each real number r, there exists exactly one number $v \in V$ such that $v - r$ is rational
  - Let $q_1, q_2, \ldots$ be the rational numbers in $[-1,1]$, construct sets $V_k = V + q_k$
  - $[0,1] \subseteq \bigcup_k V_k \subseteq [-1,2]$
  - $l(V_k) = l(V)$, and there are infinitely many V_k

- l is called the *Lebesgue measure* on real numbers

Measurable Spaces and Measures

- (**S**, **B**) is a measurable space
  - **S** is a set
  - **B** is a σ-algebra on **S**, which is a collection of subsets of **S**
    - It contains $\emptyset$
    - Closed under complementation in **S**
    - Closed under countable union
  - The elements of **B** are called measurable sets

- If **F** is a collection of subsets of **S**, $\boldsymbol{\sigma(F)}$ is the smallest $\boldsymbol{\sigma}$-algebra containing **F**, or $\sigma(\mathcal{F}) \triangleq \bigcap\{\mathcal{A} \mid \mathcal{F} \subseteq \mathcal{A} \text{ and } \mathcal{A} \text{ is a } \sigma\text{-algebra}\}$. We say (S, $\boldsymbol{\sigma(F)}$) is generated by **F**.

Measurable Functions

- $(\boldsymbol{S}, \boldsymbol{B_S})$ and $(\boldsymbol{T}, \boldsymbol{B_T})$ are measurable spaces. A function $f: \boldsymbol{S} \to \boldsymbol{T}$ is measurable if $f^{-1}(\boldsymbol{B}) = \{x \in S | f(x) \in B\}$ for every $\boldsymbol{B} \in \boldsymbol{B_T}$ is a measurable subset of S

Example:

$$\chi_B(s) = \begin{cases} 1, & s \in B, \\ 0, & s \notin B. \end{cases}$$

Measures: Definitions

- A signed (finite) measure on $(\boldsymbol{S}, \boldsymbol{B})$ is a countably additive map $\mu: \boldsymbol{B} \to \boldsymbol{R}$ such that $\mu(\emptyset) = 0$
- Positive signed measure: $\mu(A) \geq 0$ for all $A \in \boldsymbol{B}$
- A positive measure is a probability measure if $\mu(S) = 1$
- …is a subprobability measure if $\mu(S) \leq 1$

Measures: Definitions

- If $\mu(B) = 0$, then B is a μ-nullset

- A property is said to hold μ-almost surely (everywhere) if the sets of points on which it does not hold is contained in nullset

- In probability theory, measures are sometimes called distributions

Measures: Discrete Measures

- For $s \in S$, the Dirac measure, or Dirac delta, or point mass on s:

$$\delta_s(B) = \begin{cases} 1, & s \in B, \\ 0, & s \notin B. \end{cases}$$

- A measure is discrete if it is a countable weighted sum of Dirac measures
  - If the weights add up to one, then it is a discrete probability measure

- Continuous measure: $\mu(\{s\}) = 0$ for all singleton sets {s} in $\boldsymbol{B}$ of $(\boldsymbol{S}, \boldsymbol{B})$

Measures: Pushforward Measure and Lebesgue Integration

- Given $f: (\boldsymbol{S}, \boldsymbol{B_S}) \rightarrow (\boldsymbol{T}, \boldsymbol{B_T})$ measurable, an a measure μ on $\boldsymbol{B_S}$, the **pushfoward measure** $\mu(f^{-1}(B))$ on $\boldsymbol{B_T}$ is defined as

$$f_*(\mu)(B) = \mu(f^{-1}(B)), \quad B \in \mathcal{B}_T .$$

- **Lebesgue integration**: given $(\boldsymbol{S}, \boldsymbol{B})$, $\mu: \boldsymbol{B} \rightarrow \boldsymbol{R}$, $f: \boldsymbol{S} \rightarrow \boldsymbol{R}$, where $\mathrm{m} < f < M$

$$\int f \, d\mu = \lim_{n \to max} \sum_{i=0}^{n} f(s_i) \mu(B_i)$$

where $B_0, \ldots, B_n$ is a measurable partition of $\boldsymbol{S}$, and the value of f does not vary more than $(M - m)/n$ in any B_i and $s_i \in B_i$

Markov Kernels

- Given $(\boldsymbol{S}, \boldsymbol{B_S})$ and $(\boldsymbol{T}, \boldsymbol{B_T})$, $P: \boldsymbol{S} \times \boldsymbol{B_T} \to \boldsymbol{R}$ is called a Markov kernel if
  - For fixed $A \in \boldsymbol{B_T}$, the map $\lambda s. P(s, A) \to \boldsymbol{R}$ is a measurable function on $(\boldsymbol{S}, \boldsymbol{B_S})$
  - For fixed $s \in \boldsymbol{S}$, the map $\lambda A. P(s, A) \to \boldsymbol{R}$ is a probability measure on $(\boldsymbol{T}, \boldsymbol{B_T})$

- Composition of two Markov kernels
  - Given $P: S \to T$, $Q: T \to U$ $(P\ ;\ Q)(s, A) = \int_{t \in T} P(s, dt) \cdot Q(t, A).$

- Given μ on $\boldsymbol{B_S}$, its push forward under the Markov Kernel P is

$$P_*(\mu)(B) = \int_{s \in S} P(s, B)\ \mu(ds).$$

More on Markov Kernels

- $(\boldsymbol{S}, \boldsymbol{B_S})$: x = … (x>0)
- $(\boldsymbol{T}, \boldsymbol{B_T})$: y = uniform(0,x)
- Markov kernel $P(x, \bigcup_{i=1}^{i=M}[a_i, b_i]) = \sum_{i=1}^{i=M} length([a_i, b_i] \cap [0, x])/x$

More on Markov Kernels

- $(\boldsymbol{S}, \boldsymbol{B_S})$: x = … (x>0)
- $(\boldsymbol{O}, \boldsymbol{B_O})$: y = uniform(0,x)
- $(\boldsymbol{T}, \boldsymbol{B_T})$: z = uniform(0,y)
- Composition: $(P;Q)(x,[0,z]) = \int_{y\in[0,\infty]} P(x,dy) * Q(y,[0,z])$

z < x

$$= \int_{y\in[0,x]} \frac{dy}{x} * \frac{length([0,z]\cap[0,y])}{y}$$

$$= \int_{y\in[0,z]} \frac{dy}{x} * \frac{y}{y} + \int_{y\in[z,x]} \frac{dy}{x} * \frac{z}{y} = \frac{z}{x} + \frac{z}{x}(lnx - lnz)$$

More on Markov Kernels

- $(\boldsymbol{S}, \boldsymbol{B_S})$: x = uniform(0.1, 1.1) $\mu([a, b]) = \mathrm{length}([\mathrm{a}, \mathrm{b}] \cap [0.1, 1.1])$
- $(\boldsymbol{T}, \boldsymbol{B_T})$: y = uniform(0,x)
- Markov kernel $P(x, \cup_{i=1}^{i=M}[a_i, b_i]) = \sum_{i=1}^{i=M} length([a_i, b_i] \cap [0, x])/x$
- μ's pushforward under P is

$$P_*(\mu)(B_T) = \int_{x \in [0.1, 1.1]} B_T \cap [0, x] * \mu(dx)$$

More on Markov Kernels

- We can use Markov kernels to define the meanings of statements

- A term can be seen as a Markov kernel that links the input variables (can be a distribution) with the output distribution

Summary

- To reason about properties and correctness of probabilistic programs, we need semantics

- To define semantics, we can
  - Decompose it into semantics of program structures
  - Link it with mathematical concepts