

# CS7480: Topics in Programming Languages: Probabilistic Programming

## Lecture 5: Exact Inference

**Instructor:** Steven Holtzen

**Place:** Northeastern University

**Term:** Fall 2021

**Course webpage:**

<https://www.khoury.northeastern.edu/home/sholtzen/CS7480Fall21/>



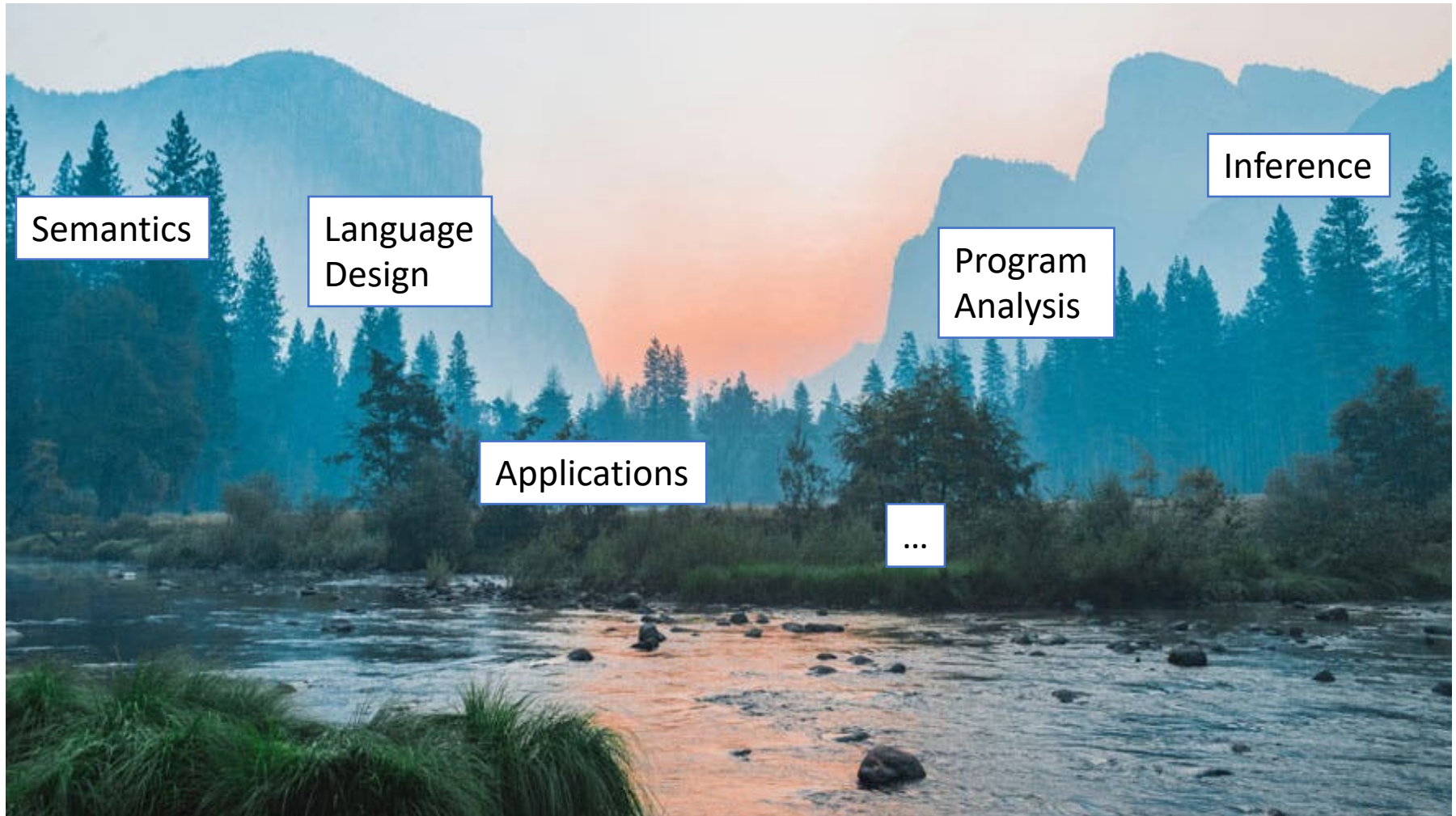
# Course Update

- Soon we will be switching to a paper-reading style course
- Check course webpage for updated calendar
- I will be picking the papers initially and we will likely spend a week or so on them to continue ramping up

# Project Update

- A couple of minor bugs with the spec already found! Thanks John and Luke
  1. Draw 5000 *total* samples for the sampling problems; spec was ambiguous
  2. A problem with latex formatting for the parser
- I'll upload a fixed spec tomorrow, let you know on teams

# The PPL Research Landscape



# Inference Overview

- Automated probabilistic inference was one of the very first applications of computers

THE JOURNAL OF CHEMICAL PHYSICS

VOLUME 21, NUMBER 6

JUNE, 1953

## Equation of State Calculations by Fast Computing Machines

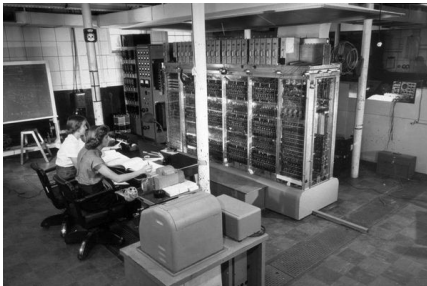
NICHOLAS METROPOLIS, ARIANNA W. ROSENBLUTH, MARSHALL N. ROSENBLUTH, AND AUGUSTA H. TELLER,  
*Los Alamos Scientific Laboratory, Los Alamos, New Mexico*

AND

EDWARD TELLER,\* *Department of Physics, University of Chicago, Chicago, Illinois*

(Received March 6, 1953)

A general method, suitable for fast computing machines, for investigating such properties as equations of state for substances consisting of interacting individual molecules is described. The method consists of a modified Monte Carlo integration over configuration space. Results for the two-dimensional rigid-sphere system have been obtained on the Los Alamos MANIAC and are presented here. These results are compared to the free volume equation of state and to a four-term virial coefficient expansion.



MANIAC I  
Los Alamos, 1953  
© Corbis Images

# Inference Overview

← → ↻ [cs.gmu.edu/~henryh/483/top-10.html](https://cs.gmu.edu/~henryh/483/top-10.html)

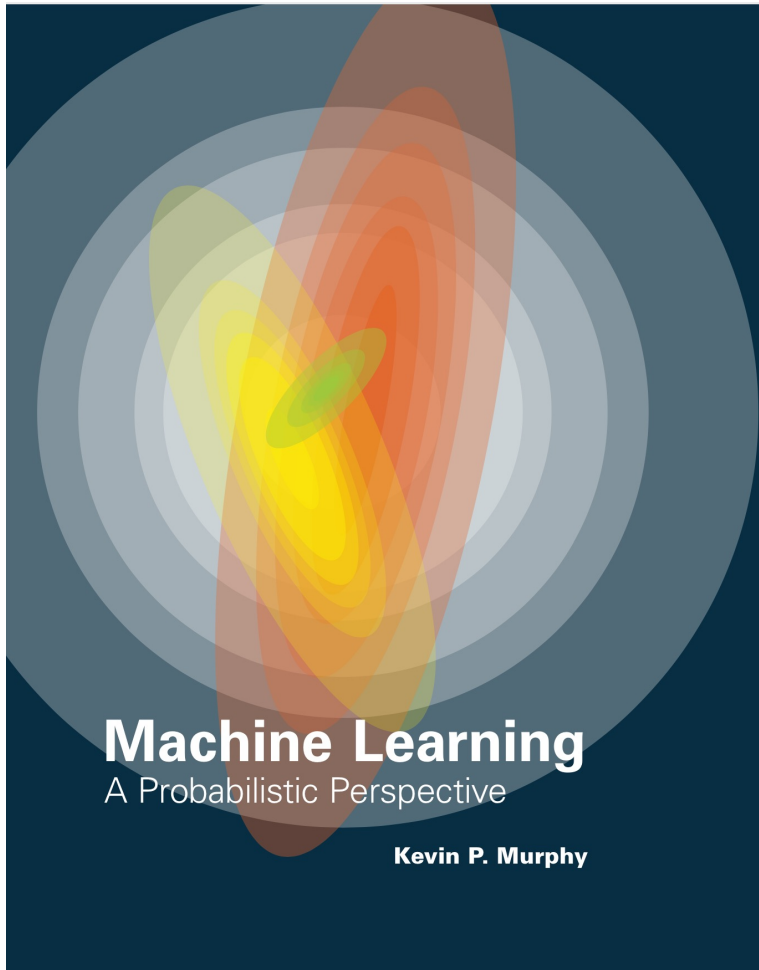
## Top 10 Algorithms

The 10 Algorithms with the Greatest Influence on the Development and Practice of Science and Engineering in the 20th Century

- Computing in Science & Engineering

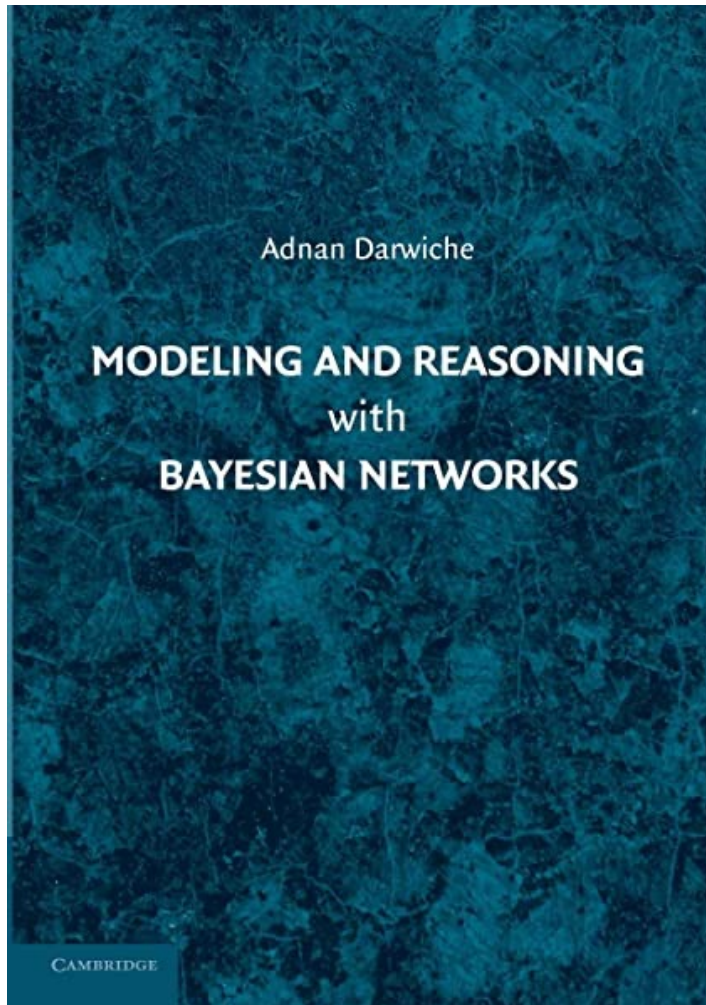
- Metropolis Algorithm for Monte Carlo
- Simplex Method for Linear Programming
- Krylov Subspace Iteration Methods
- The Decompositional Approach to Matrix Computations
- The Fortran Optimizing Compiler
- QR Algorithm for Computing Eigenvalues
- Quicksort Algorithm for Sorting
- Fast Fourier Transform
- Integer Relation Detection
- Fast Multipole Method

# Resources on Inference



- Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.
- Chapters 20, 21, 22, 23, 24

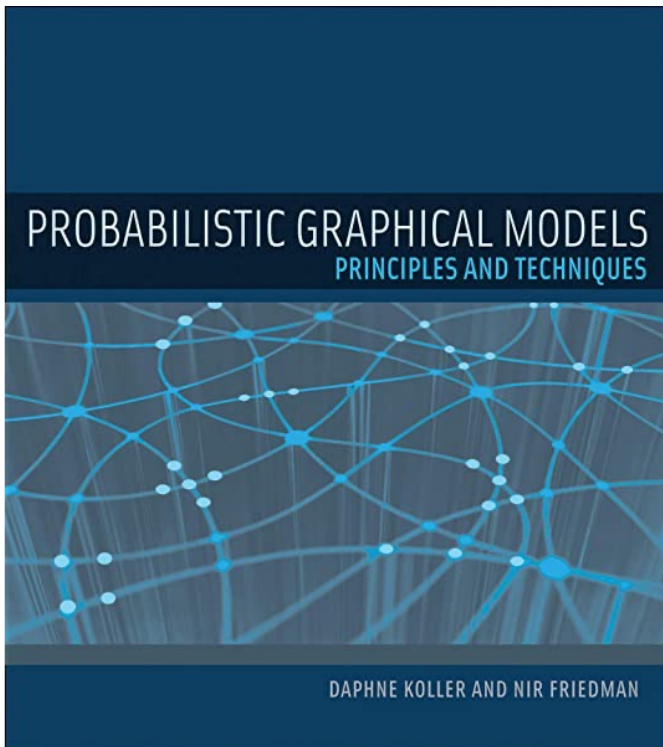
# Resources on Inference



- Darwiche, Adnan. *Modeling and reasoning with Bayesian networks*. Cambridge university press, 2009.
- Majority of the book is on inference
- Especially good reference for discrete inference

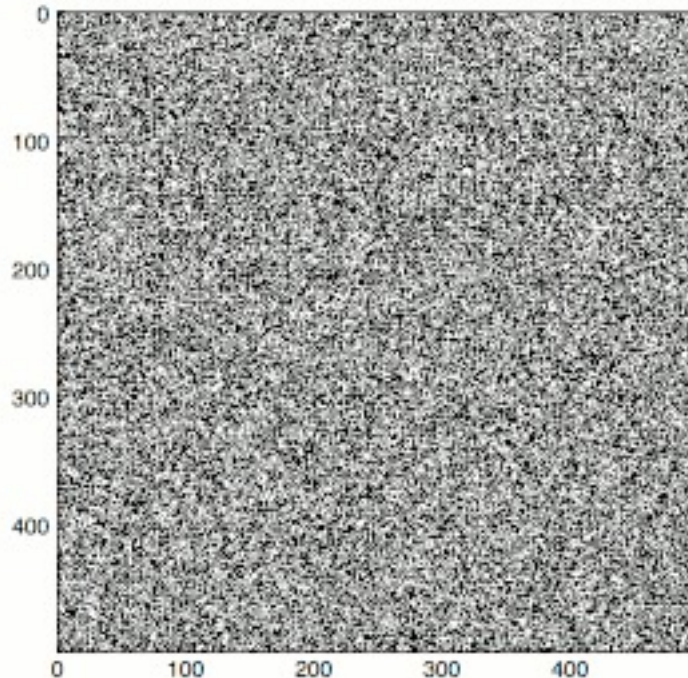


# Resources on Inference



- Koller, Daphne, and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- Basically the whole book

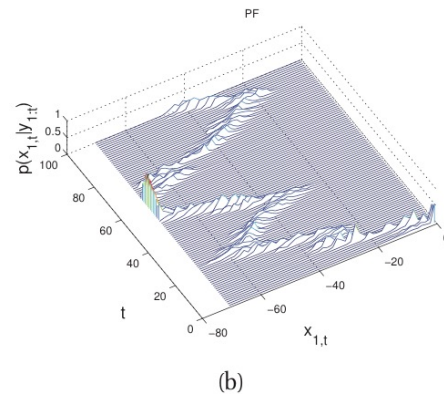
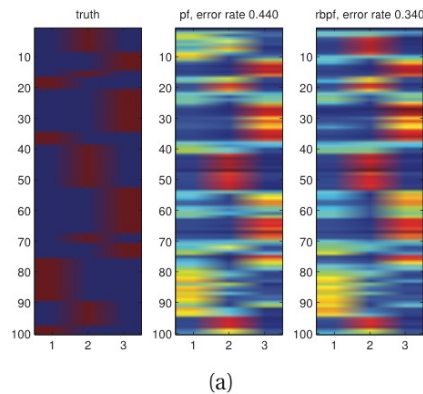
# Application: Physics



Ising model

Source: [https://upload.wikimedia.org/wikipedia/commons/e/e6/Ising\\_quench\\_b10.gif](https://upload.wikimedia.org/wikipedia/commons/e/e6/Ising_quench_b10.gif)

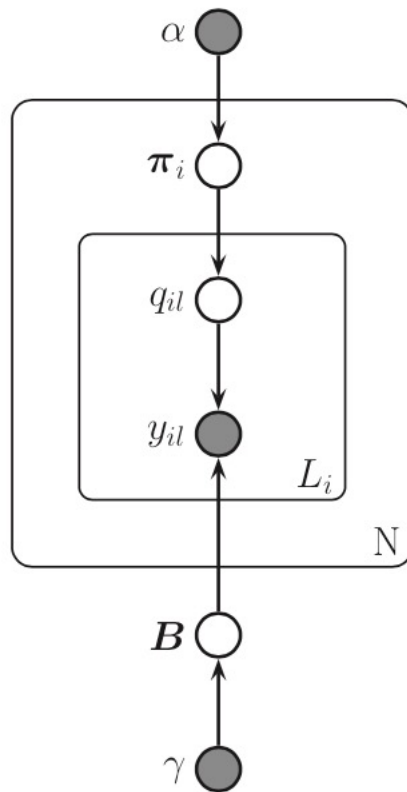
# Application: State Localization



[Murphy 2021, pg. 835]

**Figure 23.8** Belief states corresponding to Figure 23.7. (a) Discrete state. The system starts in state 2 (red x in Figure 23.7), then moves to state 3 (black \* in Figure 23.7), returns briefly to state 2, then switches to state 1 (blue circle in Figure 23.7), etc. (b) Horizontal location (PF estimate). Figure generated by `rbpfManeuverDemo`, based on code by Nando de Freitas.

# Application: Linguistics



Latent Dirichlet Allocation  
[Murphy 2012, pg. 981]

# Application: Biology

## **communications** biology

---


[Explore content](#) ▾ [About the journal](#) ▾ [Publish with us](#) ▾

---

[nature](#) > [communications biology](#) > [articles](#) > [article](#)

Article | [Open Access](#) | [Published: 24 February 2021](#)

## **Universal probabilistic programming offers a powerful approach to statistical phylogenetics**

[Fredrik Ronquist](#) , [Jan Kudlicka](#), [Viktor Senderov](#), [Johannes Borgström](#), [Nicolas Lartillot](#), [Daniel Lundén](#), [Lawrence Murray](#), [Thomas B. Schön](#) & [David Broman](#)

*Communications Biology* **4**, Article number: 244 (2021) | [Cite this article](#)

# Application: Player Matchmaking

microsoft.com/en-us/research/project/trueskill-ranking-system/

Microsoft | Research Our research ▾ Programs & events ▾ Blogs & podcasts ▾ About ▾ Sign up: Research Newsletter All Microso

## TrueSkill™ Ranking System

Established: November 18, 2005

Overview Publications Groups



The TrueSkill ranking system is a skill based ranking system for [Xbox Live](#) developed at [Microsoft Research](#). The purpose of a ranking system is to both identify and track the skills of gamers in a game (mode) in order to be able to match them into competitive matches. TrueSkill has been used to rank and match players in many different games, from Halo 3 to [Forza Motorsport 7](#).

An improved version of the TrueSkill ranking system, named [TrueSkill 2](#), launched with [Gears of War 4](#) and was later incorporated into [Halo 5](#).

The classic TrueSkill ranking system only uses the final standings of all teams in a match in order to update the skill estimates (ranks) of all players in the match. The TrueSkill 2 ranking system also uses the individual scores of players in order to weight the contribution of each player to each team. As a result, TrueSkill 2 is much faster at figuring out the skill of a new player.

# Inference and PPLs

- In many (successful!) cases, *inference came before the associated probabilistic programming language*



***Stan***

<https://mc-stan.org/>

# Bayesian Parameter Estimation

```
isBiasedCoin ~ flip 0.1;
if isBiasedCoin {
  flip1 ~ flip 0.01;
  flip2 ~ flip 0.01;
  flip3 ~ flip 0.01;
} else {
  flip1 ~ flip 0.5;
  flip2 ~ flip 0.5;
  flip3 ~ flip 0.5;
}
observe (&& flip1 (&& flip2
flip3));
return isBiasedCoin
```

- Have a *prior* probability that the coin is biased
- Given three observations, compute the *posterior* probability that the coin is biased
- Called *Bayesian learning* (or *parameter estimation*)



# Inference and PPLs

- We saw how the PL community discovered PPLs
  - Semantics, verifying randomized algorithms
- The AI community discovered PPLs from the perspective of *making inference accessible and general*
  - “Separate modeling from reasoning”
  - Same motivation as graphical models

# Inference Landscape

- **Exact inference**
  - Poly-tree algorithm
  - Variable elimination
  - Join-tree algorithm
  - Inference via compilation
- Approximate Inference (next time)
  - Direct sampling
  - Importance sampling
  - Markov-Chain Monte Carlo (MCMC)
  - Inference via Optimization

# Exact Inference

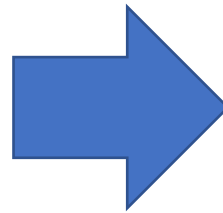
# Conciseness & Tractability

- The cost of inference is often in tension with the ***conciseness*** of the language
- We say a PPL  $A$  is *more concise* than PPL  $B$  if there is a polynomial space reduction from any program written in  $B$  to a program written in  $A$ 
  - Pretty informal definition, relies on a notion of *equality between probabilistic programs in different languages* that we yet to define

# Conciseness Reduction

- Reduction from Tabular to SimPLL statements

A	B	Pr
1	1	0.2
1	0	0.3
0	1	0.4
0	0	0.1



$s$  {  
  B ~ flip 0.6;  
  if B {  
    A ~ flip  $\frac{1}{3}$   
  } else {  
    A ~ flip 0.75  
  }  
}

- These programs are equal if they have the same semantics

$$\llbracket s \rrbracket (\{A \mapsto T, B \mapsto F\}) = 0.4 \times 0.75 = 0.3$$

# Tables are Less Concise than **SimPPL** Programs

- Consider the follow *family of SimPPL programs*

A ~ flip 0.1;

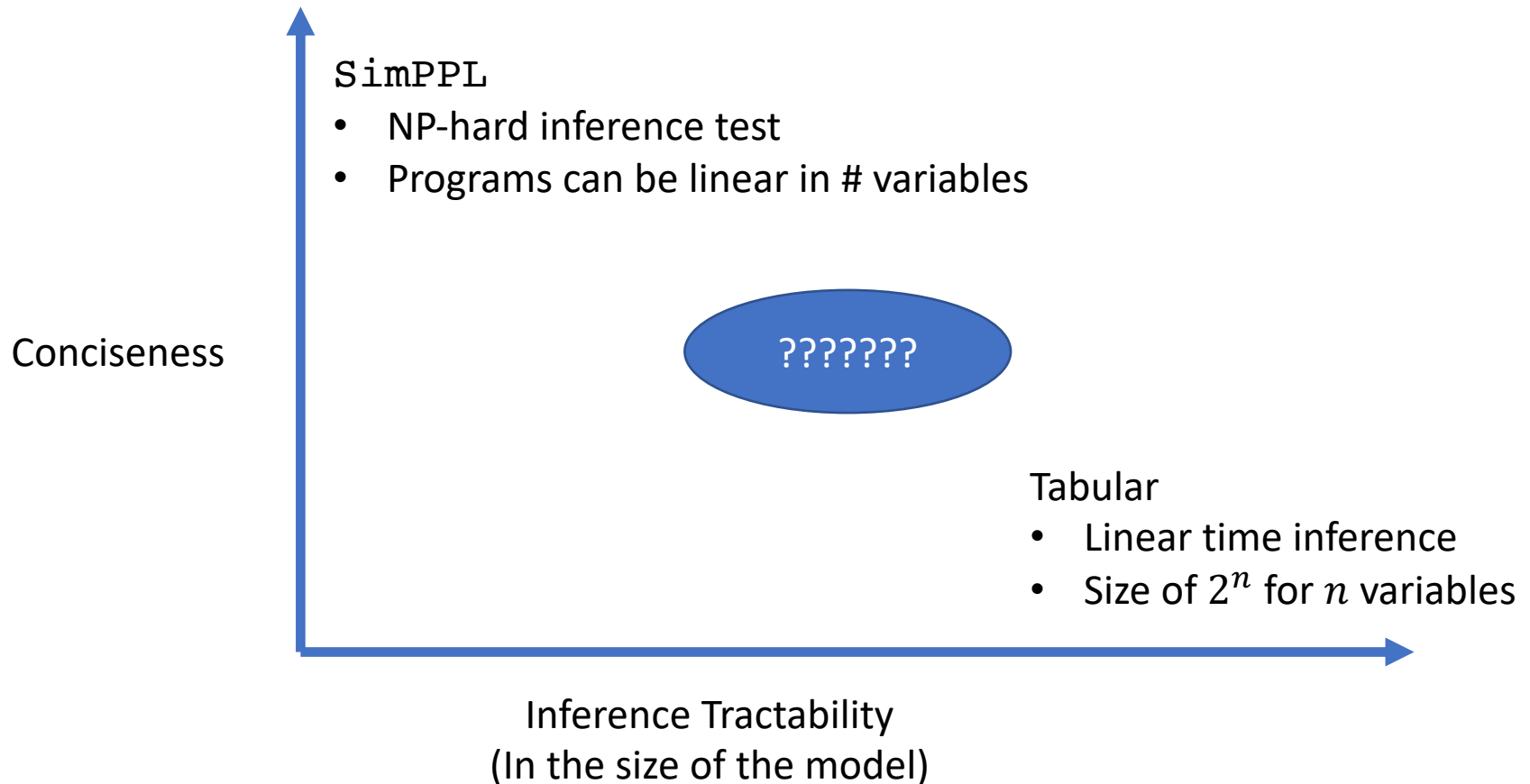
B ~ flip 0.2;

C ~ flip 0.3;

...

- Call this a family of programs  $\text{Indep}_n$
- Claim: the size of a tabular representation of  $\text{Indep}_n$  grows exponentially in  $n$
- $\Rightarrow$  SimPPL is *more concise* than tables

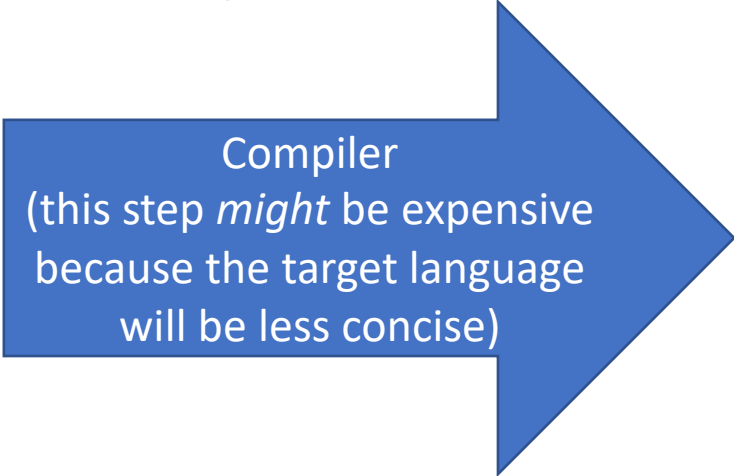
# Tractability–Conciseness Tradeoff



# Inference via Compilation

- Can we find an interesting probabilistic model somewhere in between `SimPPL` and Tabular that strikes a different tradeoff?
- Why is this this useful? We can potentially compile `SimPPL` into this representation for inference

`SimPPL`  
Slow inference :(  
Very concise 😊



Compiler  
(this step *might* be expensive  
because the target language  
will be less concise)

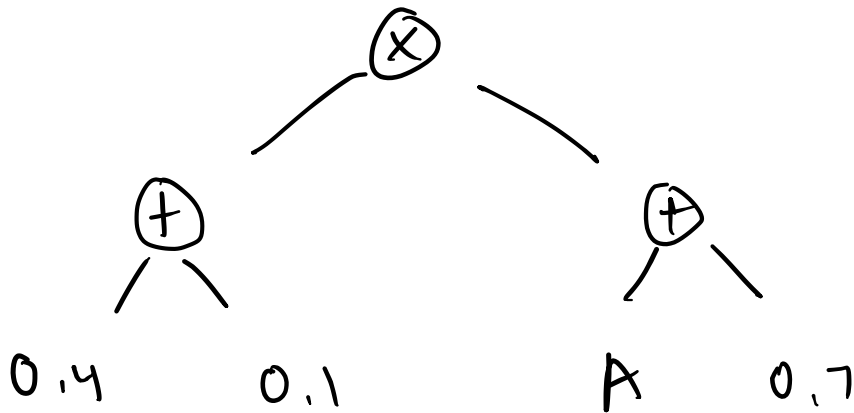
Magic Language

- Fast inference 😊
- More concise than tables



# Circuits

- An expression tree that describes a sequence of arithmetic computations
  - A well-known primitive in complexity theory

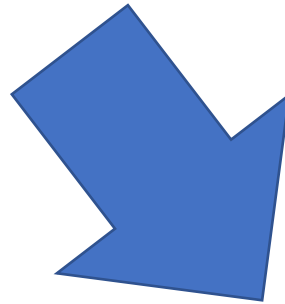


- Syntax:
  - Internal nodes: Add (+) and multiply (x)
  - Leaf nodes: Real values or variables
- Semantics:
  - Compute bottom-up, substituting variables for values

# Encoding Tables to Polynomials

- We can write a probability distribution as a *multi-linear polynomial*

A	B	Pr
1	1	0.2
1	0	0.3
0	1	0.4
0	0	0.1

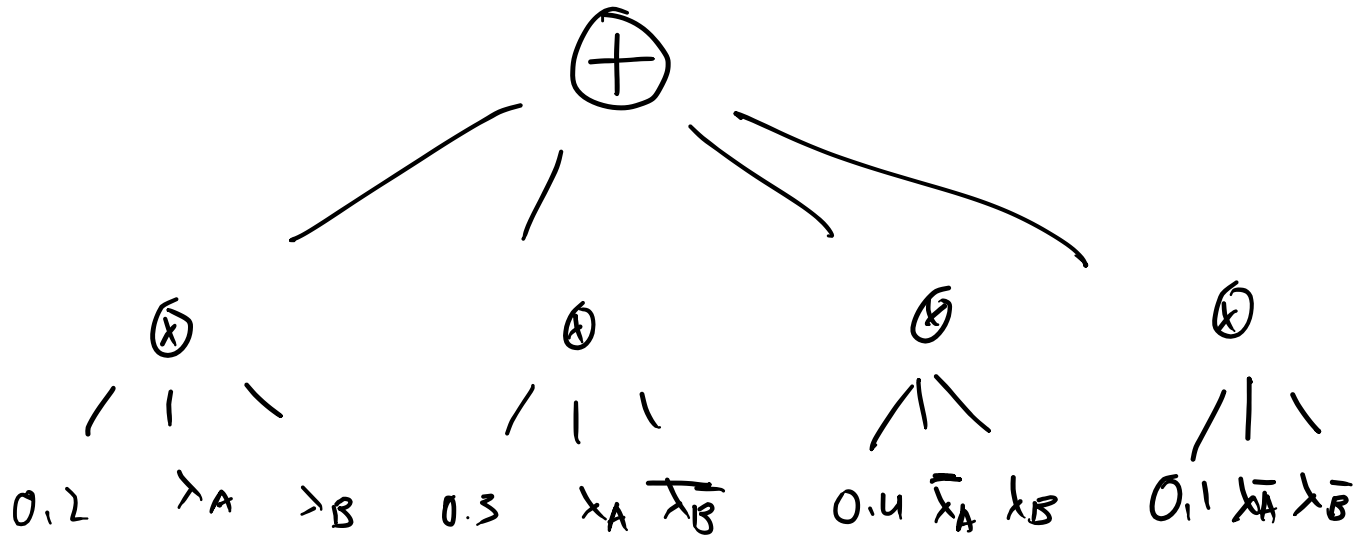


$$f(\lambda_A, \lambda_B, \bar{\lambda}_A, \bar{\lambda}_B) = 0.2\lambda_A\lambda_B + 0.3\lambda_A\bar{\lambda}_B + 0.4\bar{\lambda}_A\lambda_B + 0.1\bar{\lambda}_A\bar{\lambda}_B$$

- *Semantically equivalent*: to look up a row  $A = 1, B = 0$ , set  $\lambda_A = 1, \bar{\lambda}_A = 0, \lambda_B = 0, \bar{\lambda}_B = 1$

# Representing Polynomials as Circuits

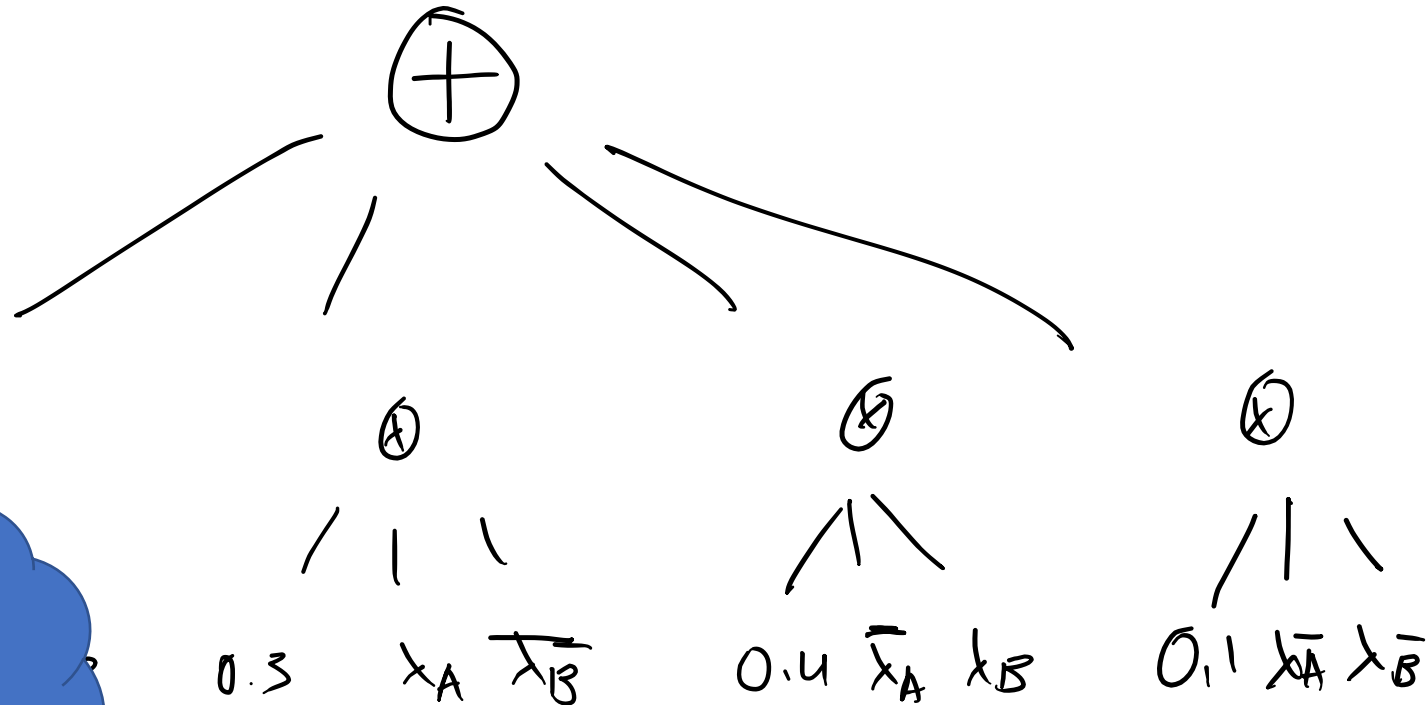
$$f(\lambda_A, \lambda_B, \bar{\lambda}_A, \bar{\lambda}_B) = 0.2\lambda_A\lambda_B + 0.3\lambda_A\bar{\lambda}_B + 0.4\bar{\lambda}_A\lambda_B + 0.1\bar{\lambda}_A\bar{\lambda}_B$$



- This circuit is called the *enumeration circuit*

# Representing Polynomials as Circuits

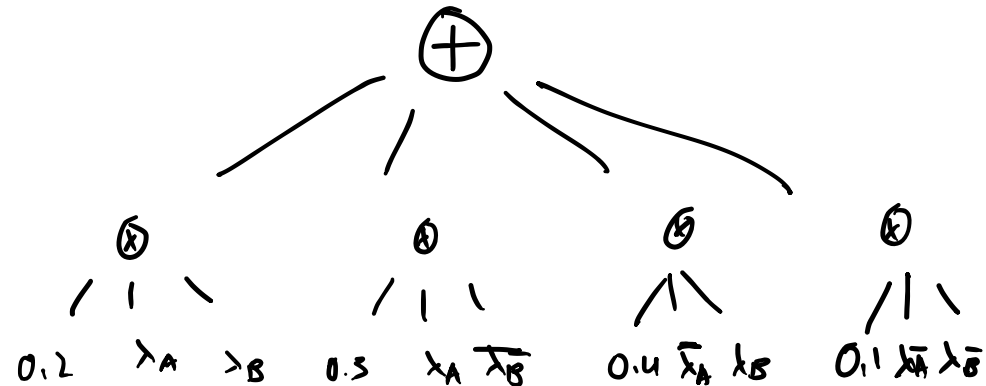
$$f(\lambda_A, \lambda_B, \bar{\lambda}_A, \bar{\lambda}_B) = 0.2\lambda_A\lambda_B + 0.3\lambda_A\bar{\lambda}_B + 0.4\bar{\lambda}_A\lambda_B + 0.1\bar{\lambda}_A\bar{\lambda}_B$$



This proves circuits are at least as concise as tables

# Circuits

- How hard is inference?



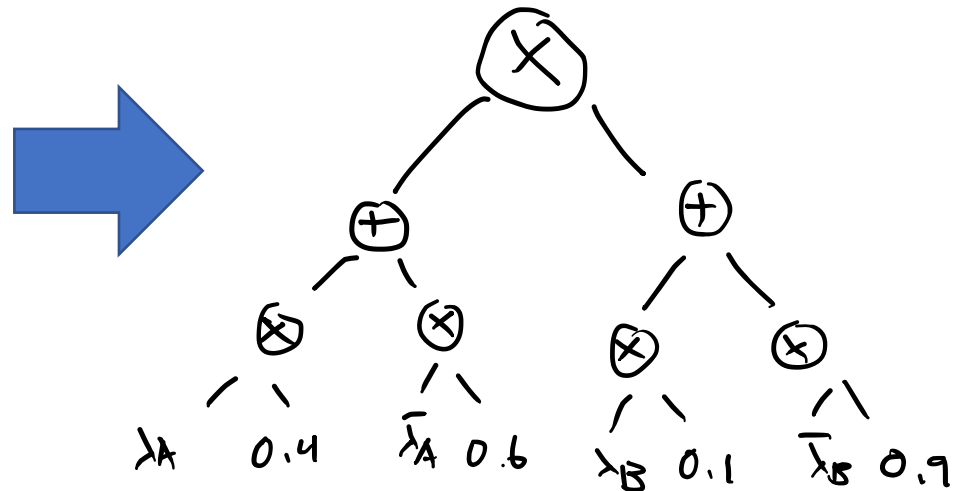
- For the above circuit, it's linear time in the size! Do a single pass bottom-up to compute any probability
  - Fast inference 😊
- However... this enumeration circuit is large! It's exponential in #variables, just like tables
  - Not concise 😞

# Circuits: Conciseness

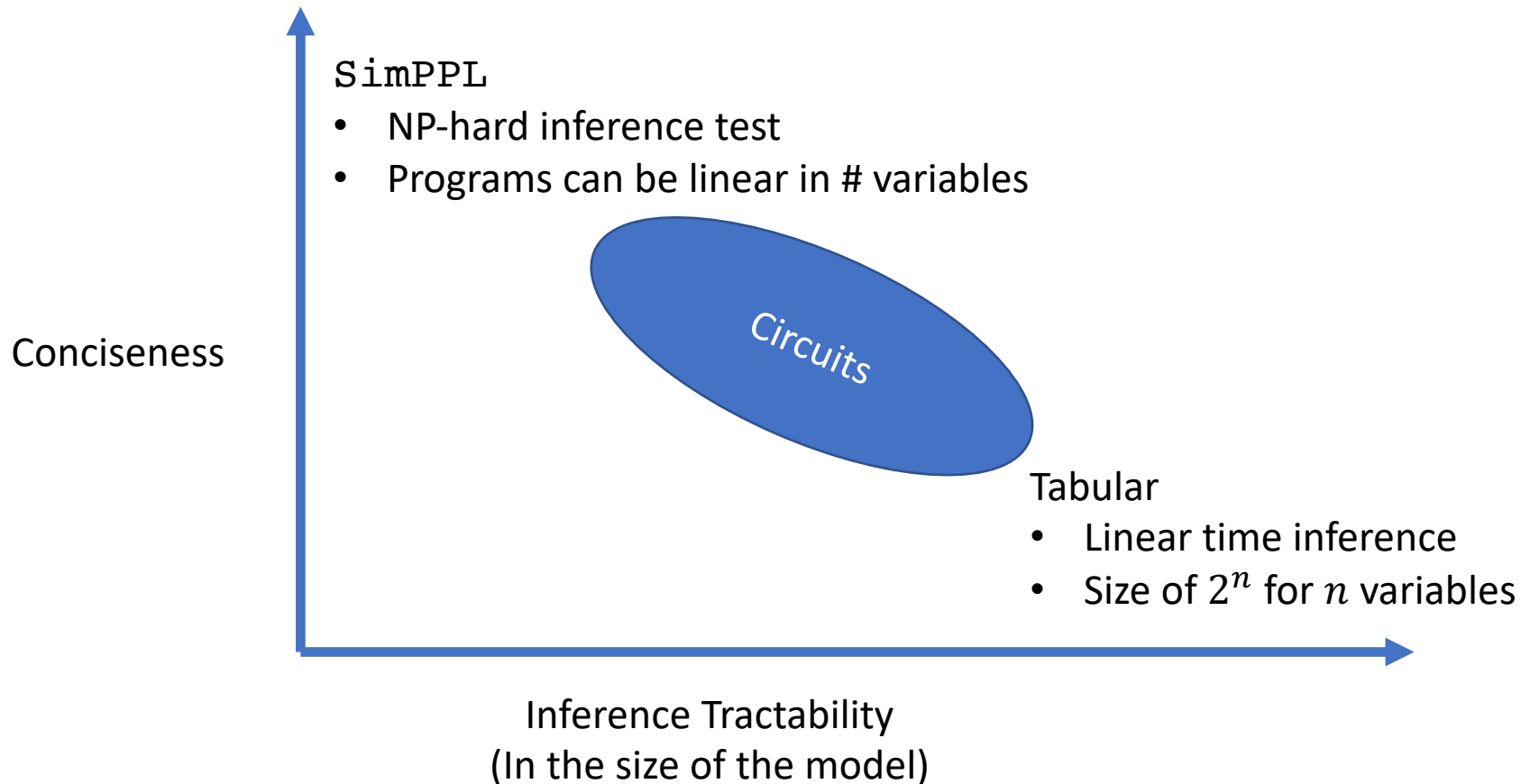
- Can circuits be *smaller than tables*? **Yes!**
  - But they have to be different kinds of circuits than enumeration circuits
- Suppose  $A$  and  $B$  are ***independent***
  - Then, we know  $\Pr(A, B) = \Pr(A) \times \Pr(B)$

A	B	Pr
1	1	0.04
1	0	0.06
0	1	0.36
0	0	0.54

Looks bigger, but scales *linearly* as number of variables grows

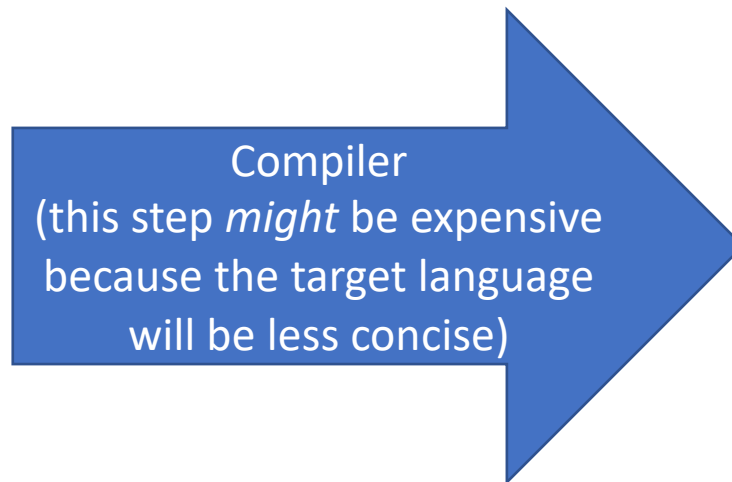


# Tractability–Conciseness Tradeoff



# Inference via Compilation

SimPPL  
Slow inference :(  
Very concise 😊



Circuit Language

- Fast inference 😊
- More concise than tables

- Next time:
  - More circuit languages
  - How to compile



# Inference via Compilation

- Been applied to do fast inference in a variety of domains
  - Discrete Bayesian networks
    - Chavira, Mark, and Adnan Darwiche. "Compiling Bayesian networks with local structure." *IJCAI*. Vol. 5. 2005.
    - Chavira, Mark, Adnan Darwiche, and Manfred Jaeger. "Compiling relational Bayesian networks for exact inference." *International Journal of Approximate Reasoning* 42.1-2 (2006): 4-20.
  - Probabilistic logic programs
    - Fierens, Daan, et al. "Inference in probabilistic logic programs using weighted CNF's." *arXiv preprint arXiv:1202.3719* (2012).
  - Discrete probabilistic programs
    - Holtzen, Steven, Guy Van den Broeck, and Todd Millstein. "Scaling exact inference for discrete probabilistic programs." *Proceedings of the ACM on Programming Languages* 4.OOPSLA (2020): 1-31.

# Conclusion

- One of the main jobs of a probabilistic programming language designer is to strike a useful balance between tractability and expressivity
- Research direction: exploring the space of languages with different tractability/conciseness tradeoffs

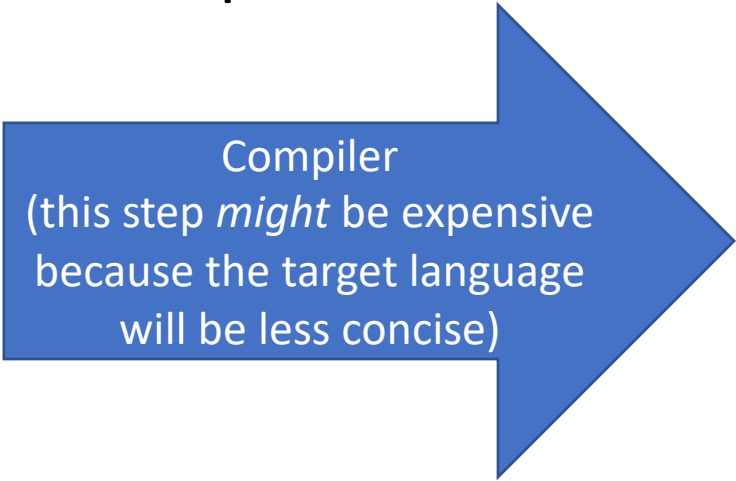
# More About TPMs

- A growing field
  - Not just for discrete models [

# Inference via Compilation

- Can we find an interesting probabilistic model somewhere in between `SimPPL` and Tabular that strikes a different tradeoff?
- Why is this this useful? We can potentially compile `SimPPL` into this representation for inference

`SimPPL`  
Slow inference :(  
Very concise 😊



Compiler  
(this step *might* be expensive  
because the target language  
will be less concise)

Magic Language

- Fast inference 😊
- More concise than tables