

**UCLA**

**Computer  
Science**



# Tractable Probabilistic Circuits

Guy Van den Broeck

Stanford Statistics Seminar - Oct 3 2023

# Outline

1. What are probabilistic circuits?  
*tractable deep generative models*
2. What are they useful for?  
*controlling generative AI*
3. What is the underlying theory?  
*probability generating polynomials*

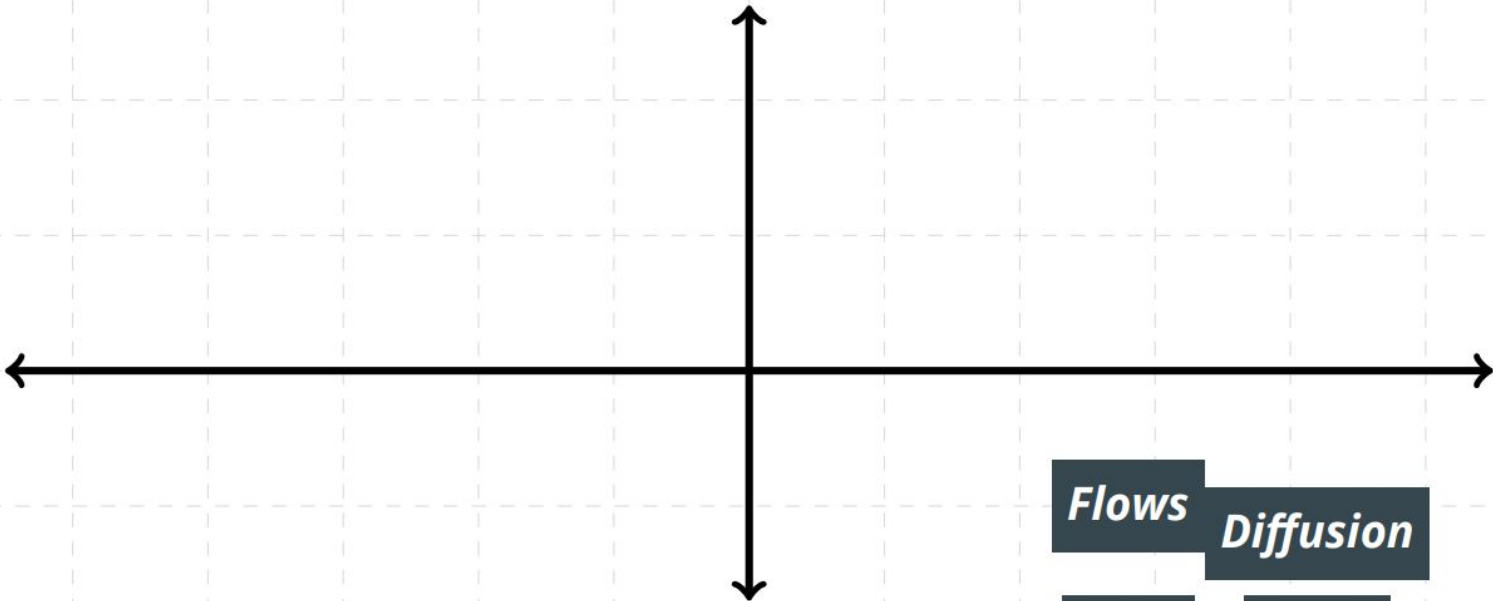
# Outline

1. **What are probabilistic circuits?**  
*tractable deep generative models*
2. What are they useful for?  
*controlling generative AI*
3. What is the underlying theory?  
*probability generating polynomials*

less expressive

more tractable

more expressive



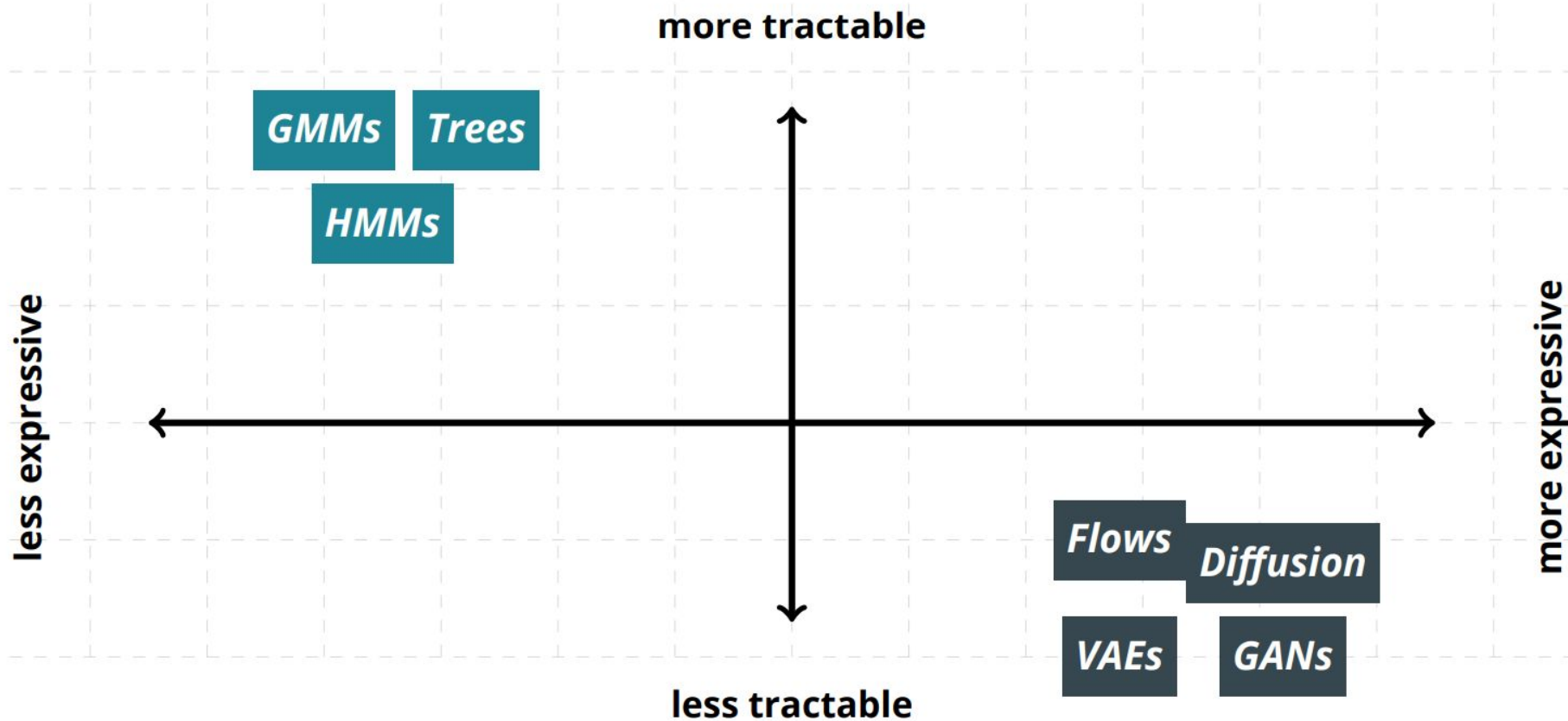
*Flows*

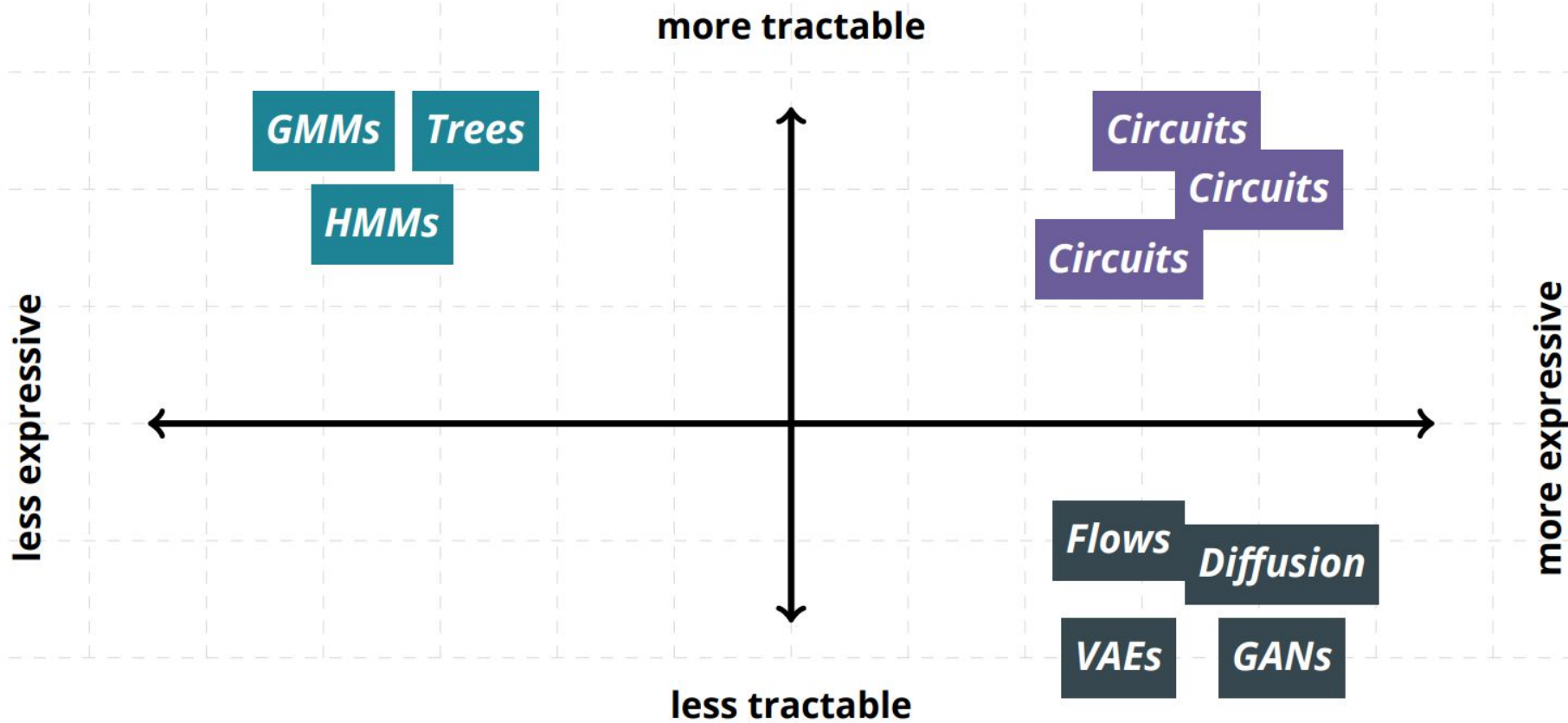
*Diffusion*

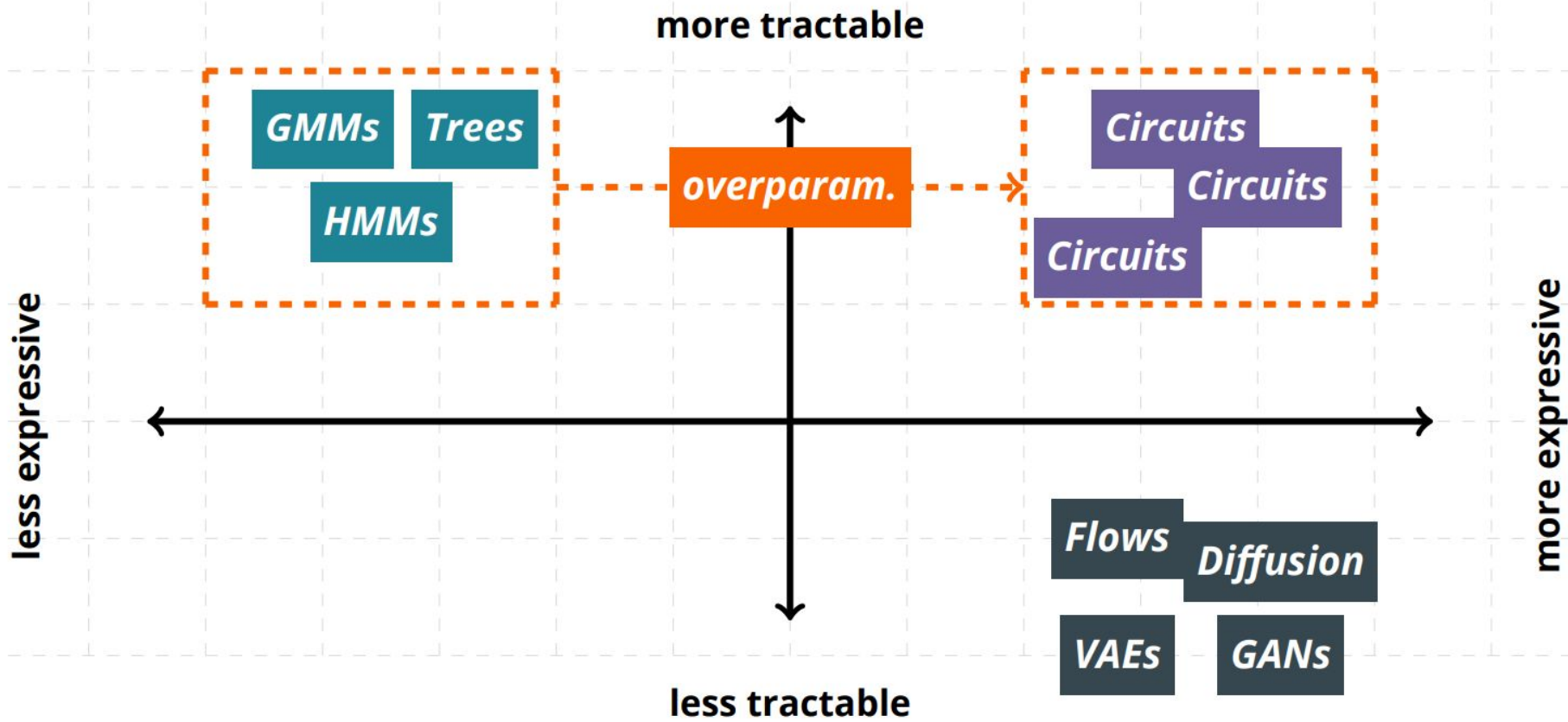
*VAEs*

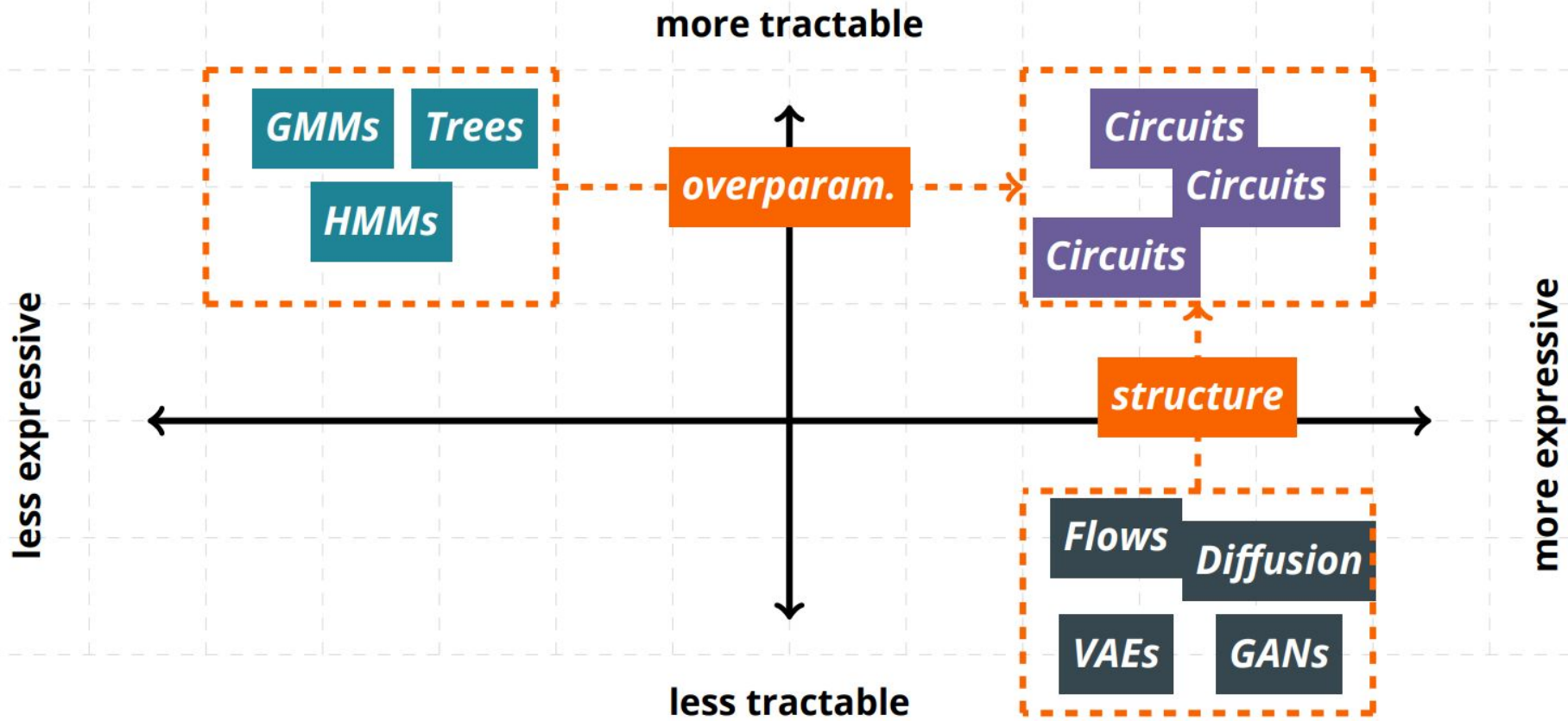
*GANs*

less tractable











# Probabilistic circuits

*computational graphs* that recursively define distributions



$\neg X$



$X_1$

# Probabilistic circuits

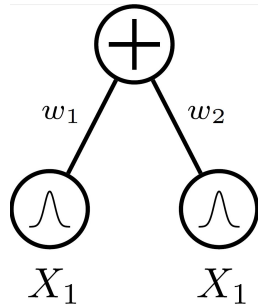
*computational graphs* that recursively define distributions



$\neg X$



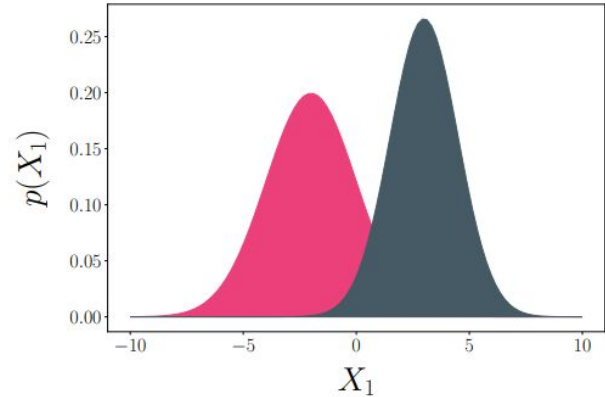
$X_1$



$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

$\Rightarrow$

*mixtures*



$$p(X) = p(Z = \mathbf{1}) \cdot p_1(X|Z = \mathbf{1}) \\ + p(Z = \mathbf{2}) \cdot p_2(X|Z = \mathbf{2})$$

# Probabilistic circuits

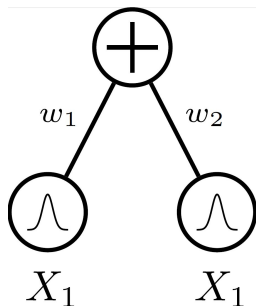
*computational graphs* that recursively define distributions



$\neg X$

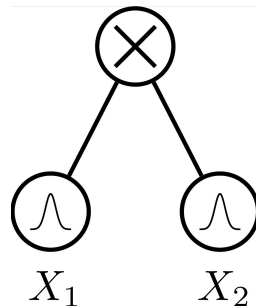


$X_1$



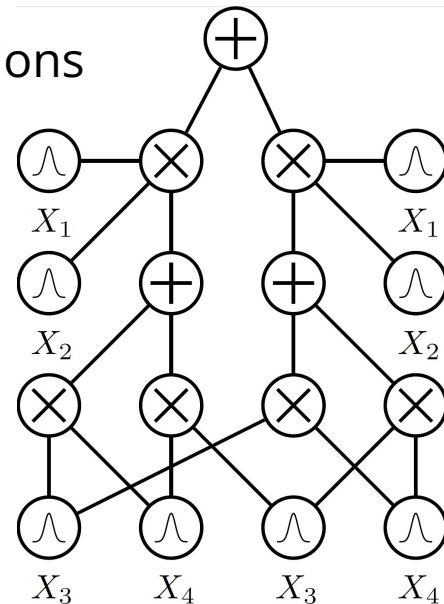
$$p(X_1) = w_1 p_1(X_1) + w_2 p_2(X_1)$$

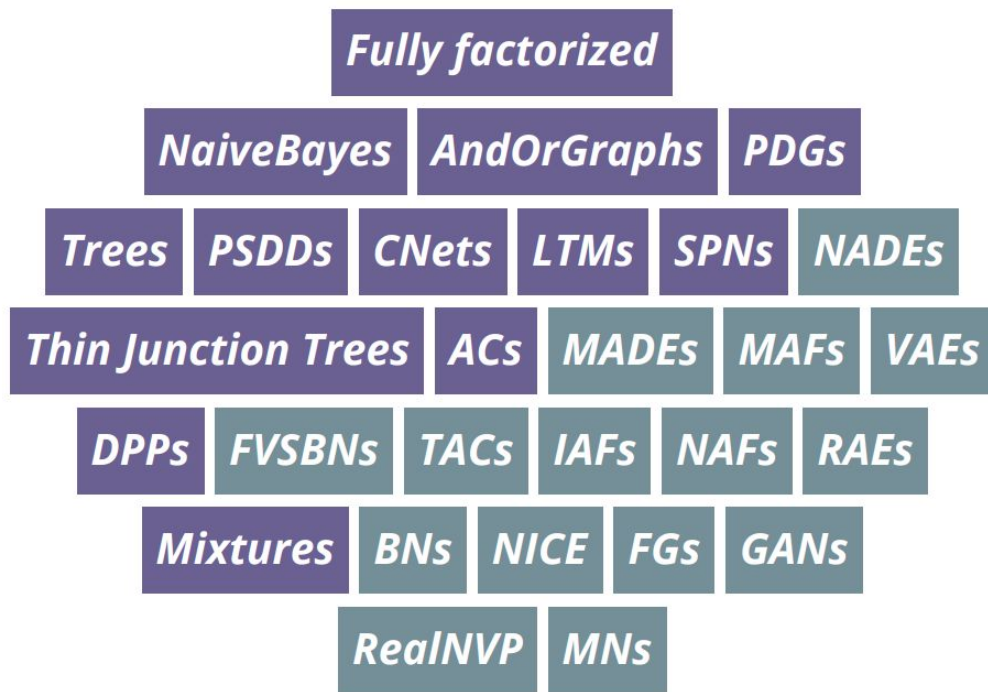
$\Rightarrow$   
*mixtures*



$$p(X_1, X_2) = p(X_1) \cdot p(X_2)$$

$\Rightarrow$   
*factorizations*

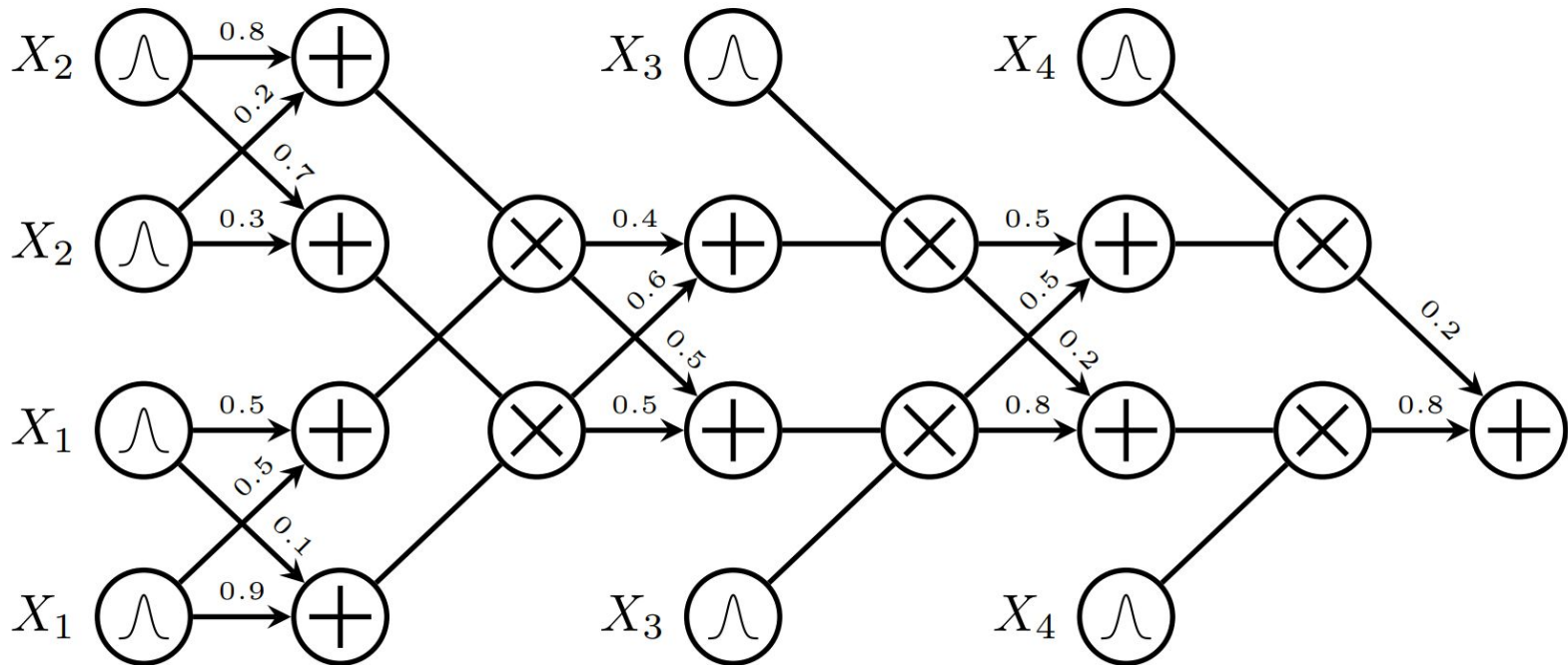




***a unifying framework* for tractable models**

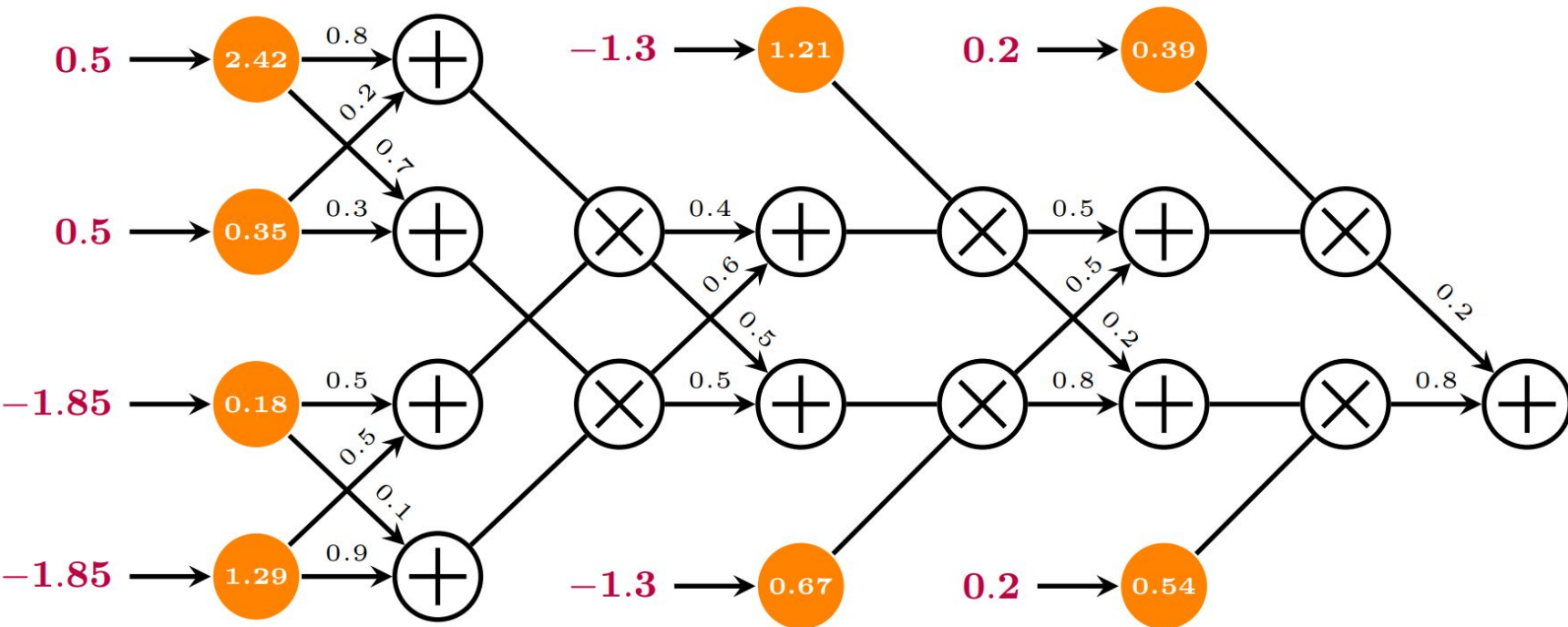
# Likelihood

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$



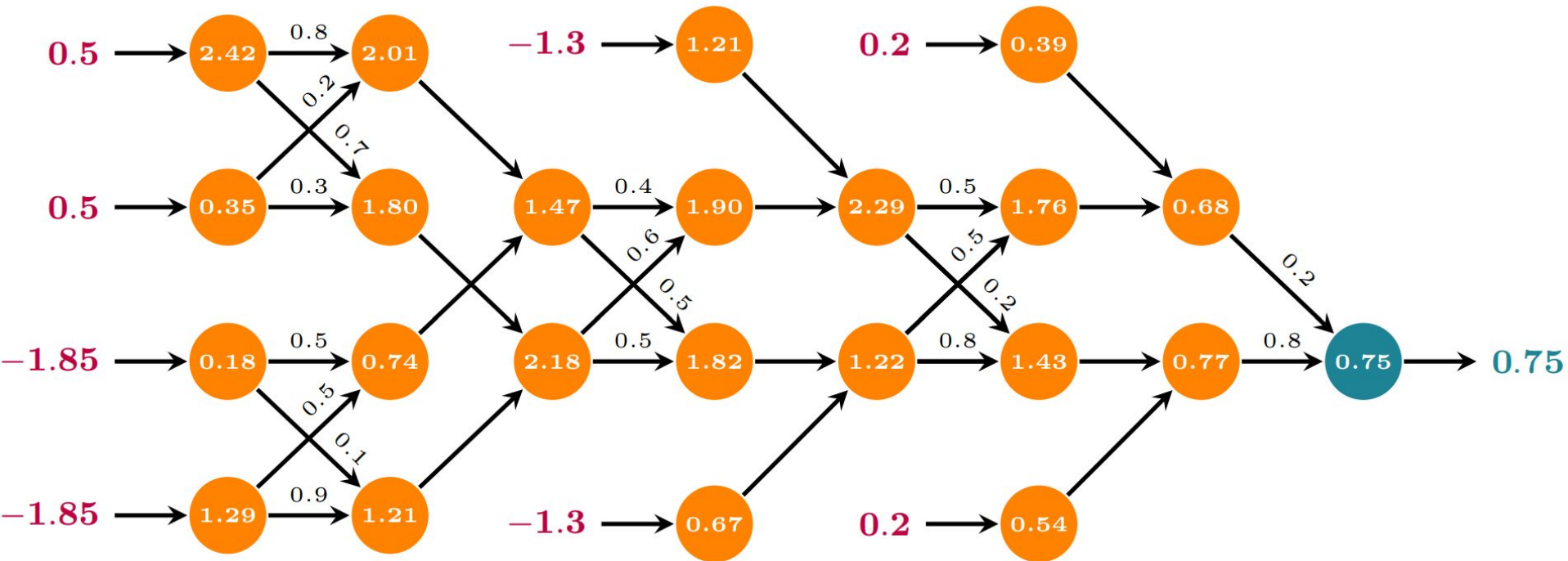
# Likelihood

$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$



# Likelihood

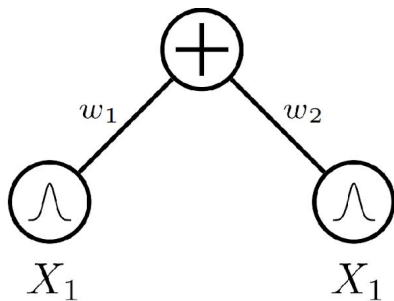
$$p(X_1 = -1.85, X_2 = 0.5, X_3 = -1.3, X_4 = 0.2)$$



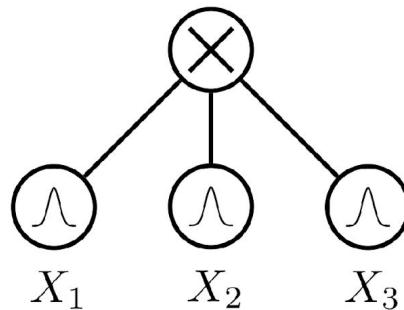
# Tractable marginals

A sum node is *smooth* if its children depend on the same set of variables.

A product node is *decomposable* if its children depend on disjoint sets of variables.



***smooth circuit***



***decomposable circuit***



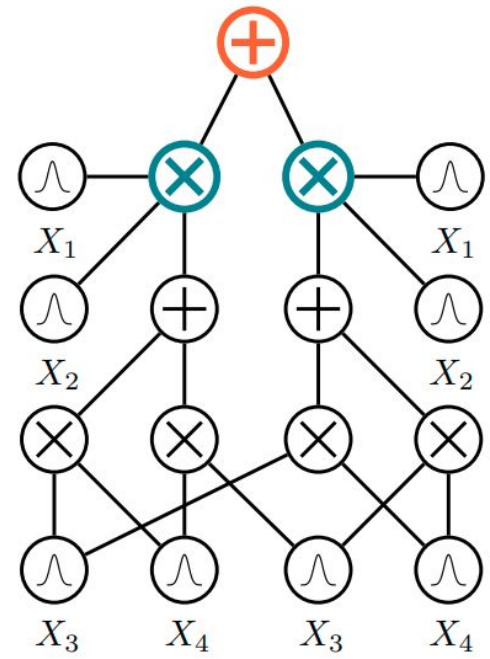
# Smoothness + decomposability = tractable MAR

If  $p(\mathbf{x}) = \sum_i w_i p_i(\mathbf{x})$ , (**smoothness**):

$$\int p(\mathbf{x}) d\mathbf{x} = \int \sum_i w_i p_i(\mathbf{x}) d\mathbf{x} =$$

$$= \sum_i w_i \int p_i(\mathbf{x}) d\mathbf{x}$$

$\Rightarrow$  integrals are "pushed down" to children

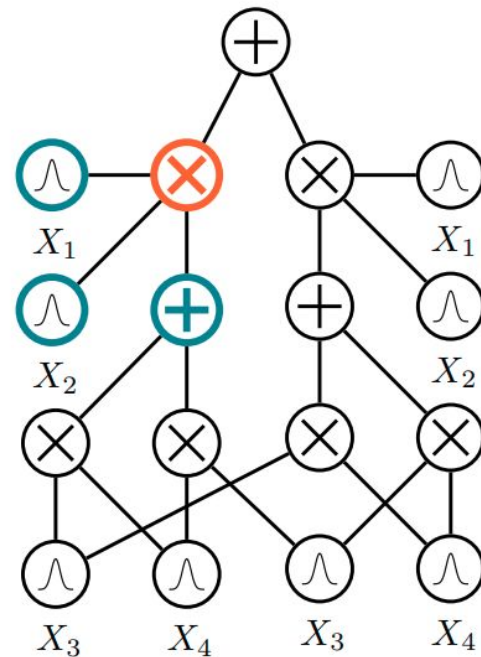


# Smoothness + decomposability = tractable MAR

If  $p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$ , (**decomposability**):

$$\begin{aligned} & \int \int \int p(\mathbf{x}, \mathbf{y}, \mathbf{z}) dx dy dz = \\ &= \int \int \int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z}) dx dy dz = \\ &= \int p(\mathbf{x}) dx \int p(\mathbf{y}) dy \int p(\mathbf{z}) dz \end{aligned}$$

$\Rightarrow$  integrals decompose into easier ones



# Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

$\Rightarrow$  linear in circuit size!

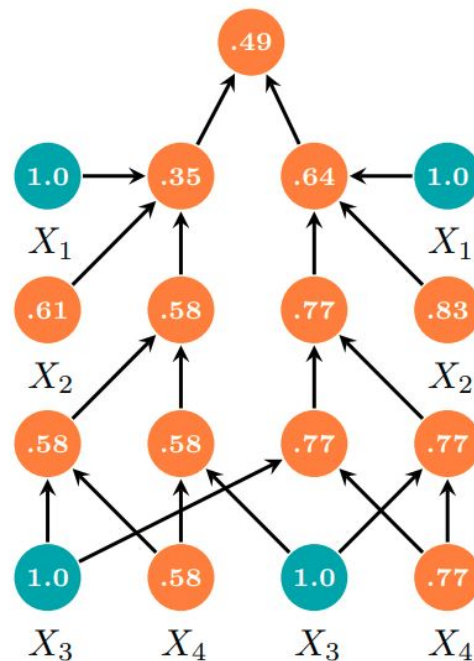
E.g. to compute  $p(x_2, x_4)$ :

■ leafs over  $X_1$  and  $X_3$  output  $Z_i = \int p(x_i) dx_i$

$\Rightarrow$  for normalized leaf distributions: **1.0**

■ leafs over  $X_2$  and  $X_4$  output **EVI**

■ feedforward evaluation (bottom-up)



# Outline

1. What are probabilistic circuits?

*tractable deep generative models*








2. **What are they useful for?**

***controlling generative AI***

3. What is the underlying theory?

*probability generating polynomials*

# *Cute, but these models cannot compete?*

bpd	2008-2020
Tabular	
MNIST	
F-MNIST	
EMNIST-L	
CIFAR	
Imagenet32	
Imagenet64	

# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021
Tabular	😐	😊
MNIST	😱	😱 > 1.67
F-MNIST	😱	😱 > 4.29
EMNIST-L	😱	😱 > 2.73
CIFAR	😱	😱
Imagenet32	😱	😱
Imagenet64	😱	😱

General-purpose architecture

# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021	ICLR 22
Tabular	😐	😊	🍰
MNIST	😱	😱 > 1.67	1.20
F-MNIST	😱	😱 > 4.29	3.34
EMNIST-L	😱	😱 > 2.73	1.80
CIFAR	😱	😱	😱 > 5.50
Imagenet32	😱	😱	😱
Imagenet64	😱	😱	😱

General-purpose architecture

Custom GPU kernels

# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular	😐	😊	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14
F-MNIST	😱	😱 > 4.29	3.34	3.27
EMNIST-L	😱	😱 > 2.73	1.80	1.58
CIFAR	😱	😱	😱 > 5.50	😱
Imagenet32	😱	😱	😱	😱
Imagenet64	😱	😱	😱	😱























General-purpose architecture

Custom GPU kernels

Pruning without losing likelihood



# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22
Tabular				
MNIST		 > 1.67	1.20	1.14
F-MNIST		 > 4.29	3.34	3.27
EMNIST-L		 > 2.73	1.80	1.58
CIFAR			 > 5.50	
Imagenet32				
Imagenet64				

	Discrete Flow	Hierarchical VAE	PixelVAE
MNIST	1.90	1.27	1.39
F-MNIST	3.47	3.28	3.66
EMNIST-L	1.95	1.84	2.26

# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23
Tabular	😐	😊	🍰	🍰	🍰
MNIST	😱	😱 > 1.67	1.20	1.14	🍰
F-MNIST	😱	😱 > 4.29	3.34	3.27	🍰
EMNIST-L	😱	😱 > 2.73	1.80	1.58	🍰
CIFAR	😱	😱	😱 > 5.50	😱	4.38
Imagenet32	😱	😱	😱	😱	4.39
Imagenet64	😱	😱	😱	😱	4.12































General-purpose architecture

Custom GPU kernels

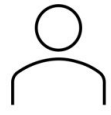
Pruning without losing likelihood

Latent Variable Distillation

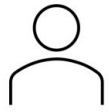
# *Cute, but these models cannot compete?*

bpd	2008-2020	2020-2021	ICLR 22	NeurIPS 22	ICLR 23	ICML 23
Tabular						
MNIST		 > 1.67	1.20	1.14		
F-MNIST		 > 4.29	3.34	3.27		
EMNIST-L		 > 2.73	1.80	1.58		
CIFAR			 > 5.50		4.38	3.87
Imagenet32					4.39	4.06
Imagenet64					4.12	3.80

	Flow	Hierarchical VAE	Diffusion
CIFAR	3.35	3.08	2.65
Imagenet32	4.09	3.96	3.72
Imagenet64	3.81	-	3.40



Generate a sentence using "frisbee", "caught" and "dog", following the given order.

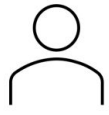


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

*ChatGPT*

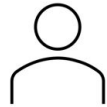


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

*ChatGPT*

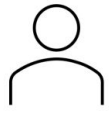


That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

*ChatGPT*

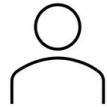


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

*ChatGPT*



That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

*ChatGPT*

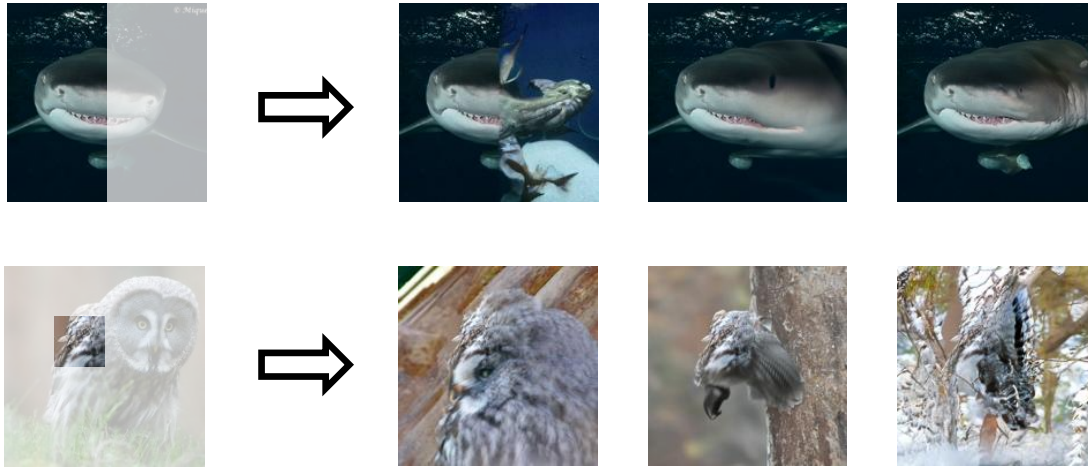


A frisbee is caught by a dog.

A pair of frisbee players are caught in a dog fight.

*GeLaTo*

# Inpainting/constrained generation is still challenging

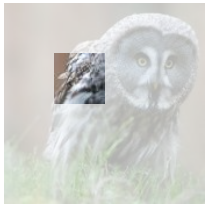
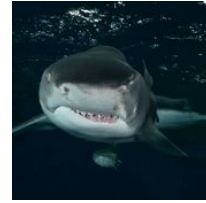


Diffusion models are good at fine-grained details, but not so good at global consistency of generated images.





# Inpainting/constrained generation is still challenging



**Tiramisu**



# What do we have?

Prefix: “The weather is”

Constraint  $\alpha$ : text contains “winter”

Model only does  $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

Train some  $q(.|\alpha)$  for a specific task distribution  $\alpha \sim p_{\text{task}}$   
(*amortized inference, encoder, masked model, seq2seq, prompt tuning,...*)

Train  $q(\text{next-token}|\text{prefix}, \alpha)$

# What do we need?

Prefix: “The weather is”

Constraint  $\alpha$ : text contains “winter”

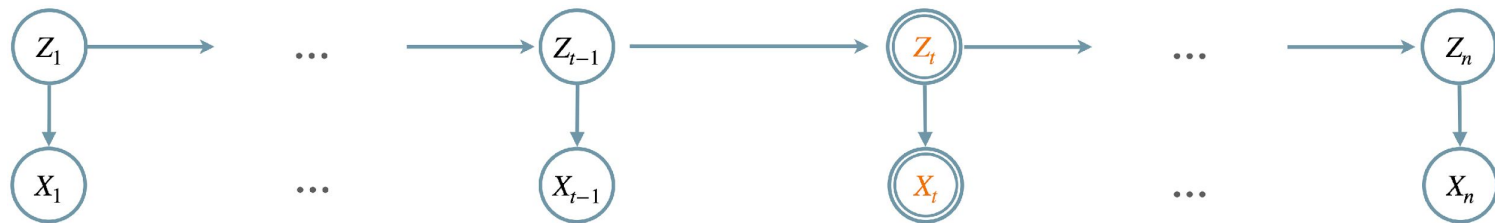
Generate from  $p(\text{next-token}|\text{prefix}, \alpha) =$

cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

***Marginalization!***

Distill an HMM  $p_{\text{hmm}}$  that approximates  $p_{\text{gpt}}$



1. HMM with 4096 hidden states and 50k emission tokens
2. Data sampled from GPT2-large (domain-adapted), minimizing  $\text{KL}(p_{\text{gpt}} // p_{\text{HMM}})$
3. Leverages latent variable distillation for training PCs at scale [ICLR 23].  
(Cluster embeddings of examples to estimate latent  $Z_i$ )

# Computing $p(\alpha \mid x_{1:t+1})$

For constraint  $\alpha$  in CNF:

$$(w_{1,1} \vee \dots \vee w_{1,d_1}) \wedge \dots \wedge (w_{m,1} \vee \dots \vee w_{m,d_m})$$

where each  $w_{ij}$  is a keyword (i.e. a string of tokens),  
representing that  $w_{ij}$  appears in the generated text.

e.g.,  $\alpha = (\text{"swims"} \vee \text{"like swimming"}) \wedge (\text{"lake"} \vee \text{"pool"})$

# Computing $p(\alpha \mid x_{1:t+1})$

For constraint  $\alpha$  in CNF:

$$(w_{1,1} \vee \dots \vee w_{1,d_1}) \wedge \dots \wedge (w_{m,1} \vee \dots \vee w_{m,d_m})$$

where each  $w_{ij}$  is a keyword (i.e. a string of tokens),  
representing that  $w_{ij}$  appears in the generated text.

e.g.,  $\alpha = (\text{"swims"} \vee \text{"like swimming"}) \wedge (\text{"lake"} \vee \text{"pool"})$

Efficient algorithm:

For  $m$  clauses and sequence length  $n$ , time-complexity for HMM generation is  $O(2^{|m|}n)$

Trick: dynamic programming with clever preprocessing and local belief updates

# CommonGen: a Challenging Benchmark

Given 3-5 concepts (keywords), goal is to generate a sentence using all keywords, in any order and any form of inflections. e.g.,

Input: snow drive car

Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

$$(w_{1,1} \vee \dots \vee w_{1,d_1}) \wedge \dots \wedge (w_{m,1} \vee \dots \vee w_{m,d_m})$$

Each clause represents the inflections for one keyword.

# GeLaTo Overview



**Lexical Constraint**  $\alpha$ : sentence contains keyword "winter"

**Constrained Generation:**  $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

**✗ intractable**

**✓ efficient**

Pre-trained  
Language Model

Tractable  
Probabilistic Model

Minimize KL-divergence

$x_{t+1}$	$\Pr_{LM}(x_{t+1}   x_{1:t})$
cold	0.05
warm	0.10

$x_{t+1}$	$\Pr_{TPM}(\alpha   x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01



# GeLaTo Overview



**Lexical Constraint**  $\alpha$ : sentence contains keyword "winter"

**Constrained Generation:**  $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

**✗ intractable**

**✓ efficient**

Pre-trained  
Language Model

Tractable  
Probabilistic Model

Minimize KL-divergence

$x_{t+1}$	$\Pr_{LM}(x_{t+1}   x_{1:t})$
cold	0.05
warm	0.10

$x_{t+1}$	$\Pr_{TPM}(\alpha   x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

$x_{t+1}$	$p(x_{t+1}   \alpha, x_{1:t})$
cold	0.025
warm	0.001

# Control $p_{gpt}$ via $p_{hmm}$

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

we view  $p_{HMM}(x_{t+1} | x_{1:t}, \alpha)$  and  $p_{gpt}(x_{t+1} | x_{1:t})$  as classifiers trained for the same task with different biases; thus we generate from their weighted geometric mean:

$$p(x_{t+1} | x_{1:t}, \alpha) \propto p_{hmm}(x_{t+1} | x_{1:t}, \alpha)^w \cdot p_{gpt}(x_{t+1} | x_{1:t})^{1-w}$$

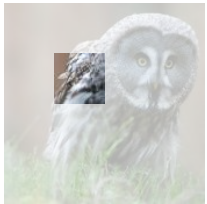
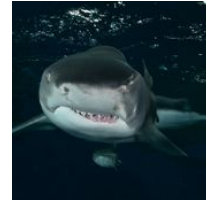
Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>
<i>Supervised</i>												
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	93.9 <sup>†</sup>
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	97.9 <sup>†</sup>
NADO (Meng et al., 2022)	44.4 <sup>†</sup>	-	30.8	-	16.1 <sup>†</sup>	-	<b>32.0<sup>†</sup></b>	-	97.1	-	88.8 <sup>†</sup>	-
GeLaTo	<b>46.0</b>	<b>45.6</b>	<b>34.1</b>	<b>32.9</b>	<b>16.7</b>	<b>16.8</b>	31.3	<b>31.9</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>	<b>100.0</b>

# Advantages of GeLaTo:

1. Constraint  $\alpha$  is guaranteed to be satisfied: for any next-token  $x_{t+1}$  that would make  $\alpha$  unsatisfiable,  $p(x_{t+1} | x_{1:t}, \alpha) = 0$  for both settings.
2. Training  $p_{\text{hmm}}$  does not depend on  $\alpha$ , which is only imposed at inference (generation) time. Once  $p_{\text{hmm}}$  is trained, we can impose whatever  $\alpha$ .
3. We can impose additional tractable constraints:
  - The keywords are generated following a particular order.
  - (Some) keywords must appear at a particular position.
  - (Some) keywords must not appear in the generated sentence.

Conclusion: you can control an intractable generative model using a tractable probabilistic circuit.

# Inpainting/constrained generation is still challenging

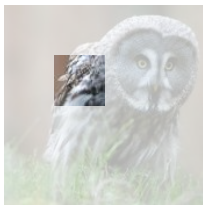


**Tiramisu**



# Constrained posterior in diffusion models

Unconstrained denoising step:  $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t) := \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1}|\tilde{\mathbf{x}}_0, \mathbf{x}_t) \cdot p_{\theta}(\tilde{\mathbf{x}}_0|\mathbf{x}_t)$



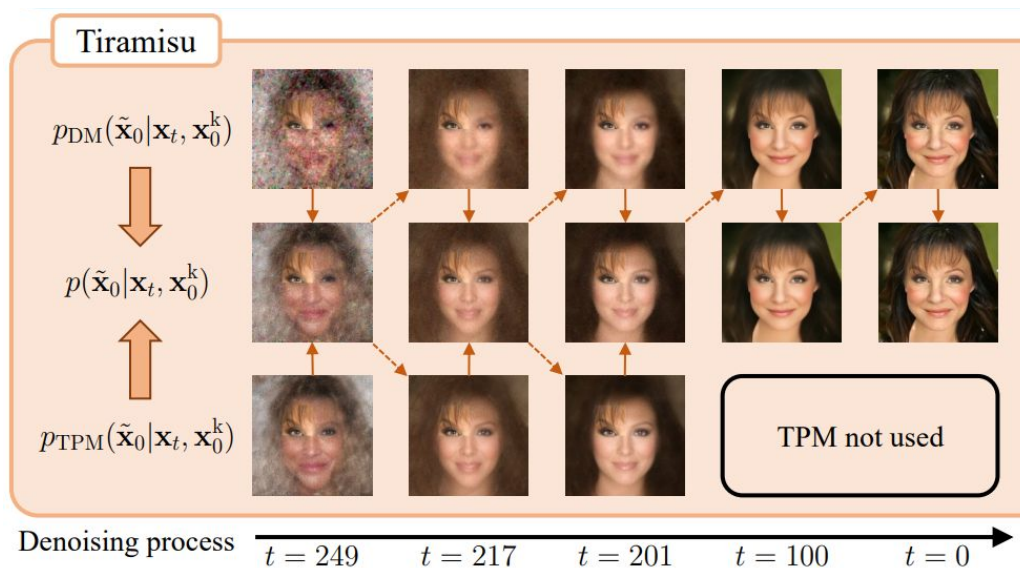
Constraint  $c$  on the generated image (e.g., inpainting)

Constrained denoising step:  $p_{\theta}(\mathbf{x}_{t-1}|\mathbf{x}_t, c) := \sum_{\tilde{\mathbf{x}}_0} q(\mathbf{x}_{t-1}|\tilde{\mathbf{x}}_0, \mathbf{x}_t) \cdot p_{\theta}(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$

Computing or sampling from the constrained posterior  $p_{\theta}(\tilde{\mathbf{x}}_0|\mathbf{x}_t, c)$  is **intractable** for diffusion models.



$$\text{Denoising } p(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k) \propto p_{\text{DM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)^\alpha \cdot p_{\text{TPM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)^{1-\alpha}$$



$$p_{\text{DM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, c)$$

From the diffusion model:  
Good at generating vivid details

$$p_{\text{TPM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, c)$$

From the probabilistic circuit:  
Exact samples – better global coherence

# Controlling the denoiser with a probabilistic circuit

CoPaint

$p_{\text{DM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)$



Tiramisu

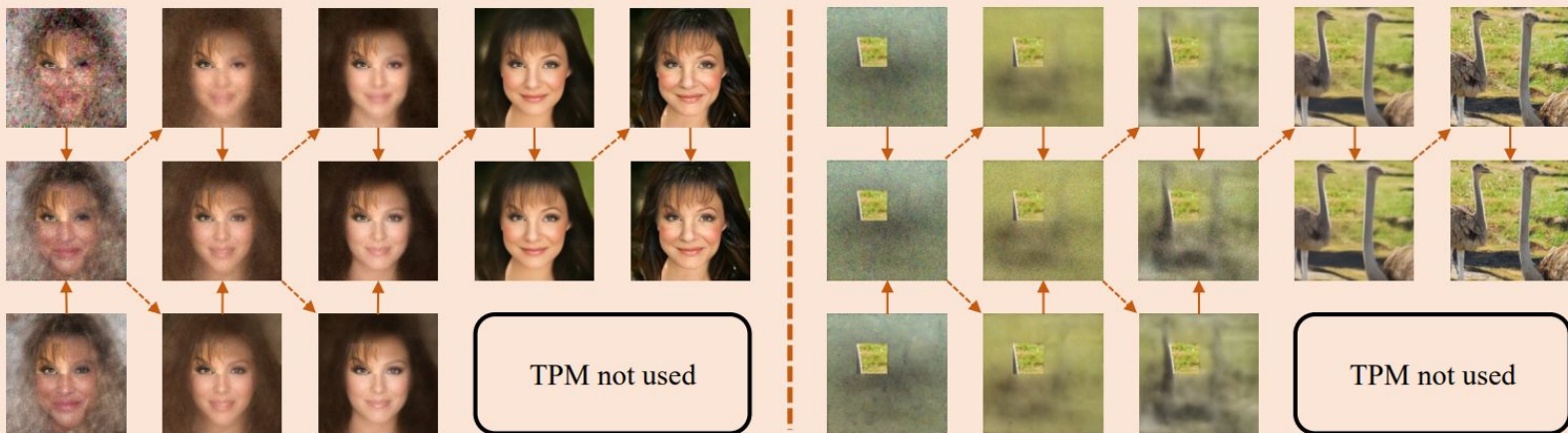
$p_{\text{DM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)$



$p(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)$



$p_{\text{TPM}}(\tilde{\mathbf{x}}_0 | \mathbf{x}_t, \mathbf{x}_0^k)$



Denoising process

$t = 249$

$t = 217$

$t = 201$

$t = 100$

$t = 0$

$t = 249$

$t = 217$

$t = 201$

$t = 100$

$t = 0$

# High-resolution image benchmarks

Tasks		Algorithms						
Dataset	Mask	Tiramisu (ours)	CoPaint	RePaint	DDNM	DDRM	DPS	Resampling
CelebA-HQ	Left	0.189	<b>0.185</b>	0.195	0.254	0.275	0.201	0.257
	Top	0.187	<b>0.182</b>	0.187	0.248	0.267	0.187	0.251
	Expand1	<b>0.454</b>	0.468	0.504	0.597	0.682	0.466	0.613
	Expand2	0.442	0.455	0.480	0.585	0.686	<b>0.434</b>	0.601
	V-strip	<b>0.487</b>	0.502	0.517	0.625	0.724	0.535	0.647
	H-strip	<b>0.484</b>	0.488	0.517	0.626	0.731	0.492	0.639
ImageNet	Left	<b>0.286</b>	0.289	0.296	0.410	0.369	0.327	0.369
	Top	<b>0.308</b>	0.312	0.336	0.427	0.373	0.343	0.368
	Expand1	<b>0.616</b>	0.623	0.691	0.786	0.726	0.621	0.711
	Expand2	<b>0.597</b>	0.607	0.692	0.799	0.724	0.618	0.721
	V-strip	0.646	0.654	0.741	0.851	0.761	<b>0.637</b>	0.759
	H-strip	0.657	0.660	0.744	0.851	0.753	<b>0.647</b>	0.774
LSUN-Bedroom	Left	<b>0.285</b>	0.287	0.314	0.345	0.366	0.314	0.367
	Top	<b>0.310</b>	0.323	0.347	0.376	0.368	0.355	0.372
	Expand1	<b>0.615</b>	0.637	0.676	0.716	0.695	0.641	0.699
	Expand2	<b>0.635</b>	0.641	0.666	0.720	0.691	0.638	0.690
	V-strip	<b>0.672</b>	0.676	0.711	0.760	0.721	0.674	0.725
	H-strip	0.679	0.686	0.722	0.766	0.726	<b>0.674</b>	0.724
Average		<b>0.474</b>	0.481	0.518	0.596	0.591	0.489	0.571



# Qualitative results on high-resolution image datasets



# Outline

1. What are probabilistic circuits?  
*tractable deep generative models*
2. What are they useful for?  
*controlling generative AI*
3. **What is the underlying theory?**  
***probability generating polynomials***

Probabilistic circuits seem awfully general.

*Are all tractable probabilistic models  
probabilistic circuits?*



# Enter: Determinantal Point Processes (DPPs)

DPPs are models where probabilities are specified by (sub)determinants

$$L = \begin{bmatrix} 1 & 0.9 & 0.8 & 0 \\ 0.9 & 0.97 & 0.96 & 0 \\ 0.8 & 0.96 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})$$

# Enter: Determinantal Point Processes (DPPs)

DPPs are models where probabilities are specified by (sub)determinants

$$L = \begin{bmatrix} 1 & 0.9 & 0.8 & 0 \\ 0.9 & 0.97 & 0.96 & 0 \\ 0.8 & 0.96 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Global Negative Dependence

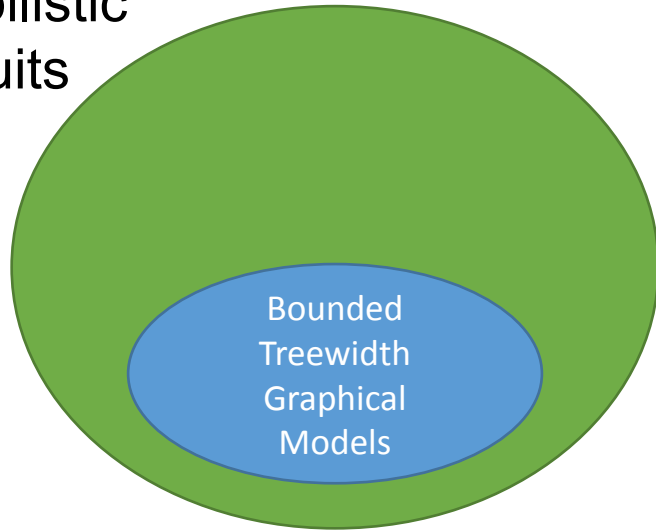
Diversity in recommendation systems

Tractable likelihoods and marginals

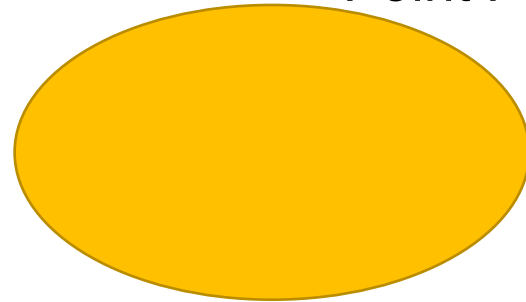
$$\Pr_L(X_1 = 1, X_2 = 0, X_3 = 1, X_4 = 0) = \frac{1}{\det(L + I)} \det(L_{\{1,2\}})$$

# *Are all tractable probabilistic models probabilistic circuits?*

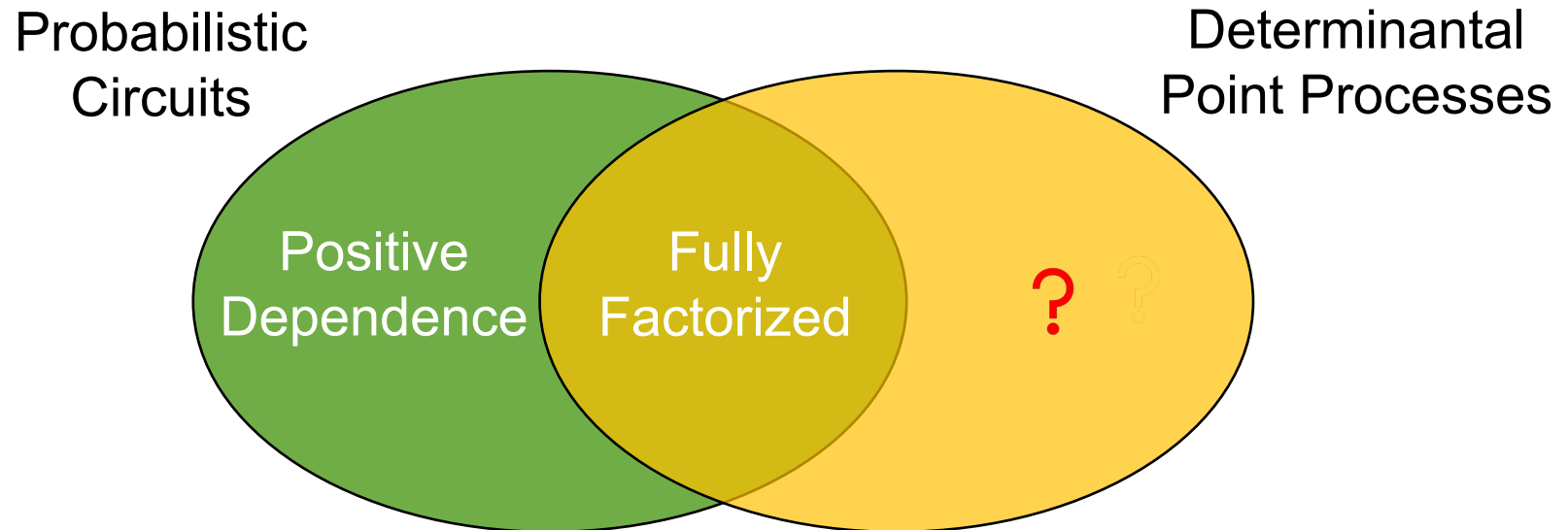
Probabilistic  
Circuits



Determinantal  
Point Processes



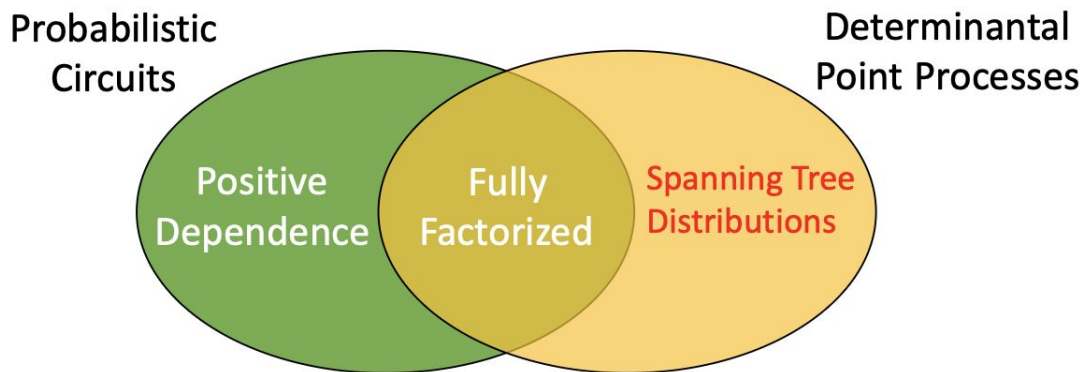
# *Are all tractable probabilistic models probabilistic circuits?*



# A separation between PCs and DPPs

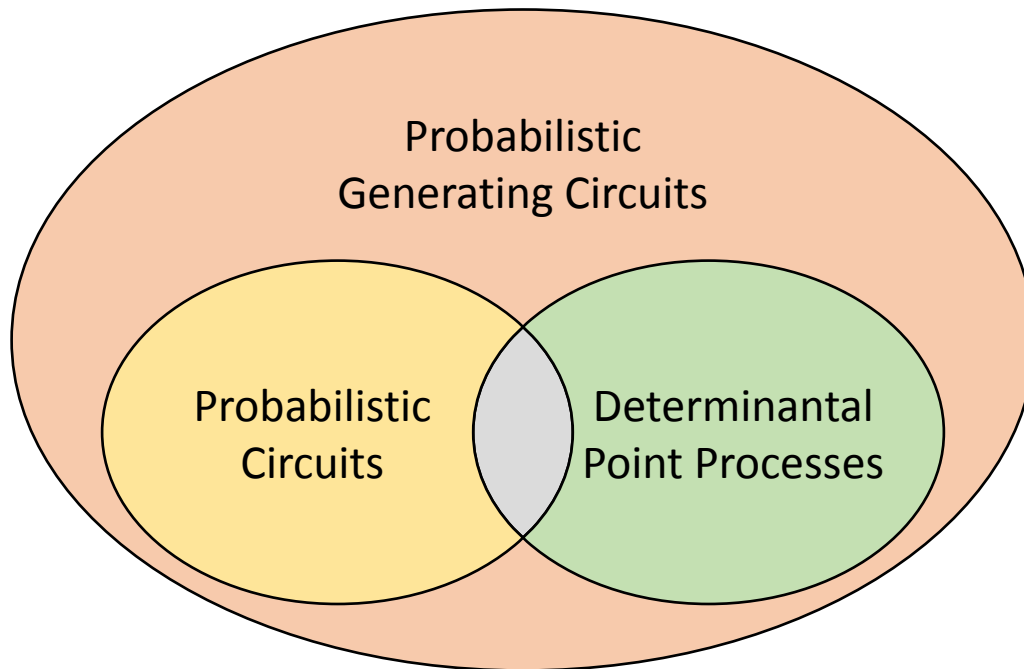
**Theorem** (Martens and Medabalimi, 2014). *Let  $P_n$  be the uniform distribution over spanning trees on  $K_n$ . For  $n \geq 20$ , the size of any smooth and decomposable PC that represents  $P_n$  is at least  $2^{n/30240}$ .*

**Theorem** (Snell, 1995). *The uniform distribution over spanning trees on the complete graph  $K_n$  is a DPP over  $\binom{n}{2}$  edges.*





# Probabilistic Generating Circuits



A Tractable Unifying Framework for PCs and DPPs

# Probability Generating Functions

$X_1$	$X_2$	$X_3$	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16



$$g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 + 0.48z_2z_3 + 0.12z_2 + 0.08z_3 + 0.02.$$

# Probability Generating Functions

$X_1$	$X_2$	$X_3$	$\Pr_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16



$$g_\beta = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 + 0.48z_2z_3 + 0.12z_2 + 0.08z_3 + 0.02.$$

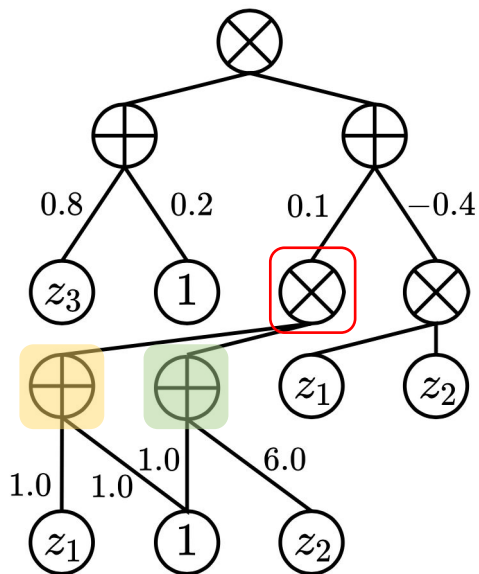


$$g_\beta = (0.1(z_1 + 1))(6z_2 + 1) - 0.4z_1z_2)(0.8z_3 + 0.2)$$

# Probabilistic Generating Circuits (PGCs)

↓

$$g_{\beta} = (0.1(z_1 + 1)(6z_2 + 1) - 0.4z_1z_2)(0.8z_3 + 0.2)$$

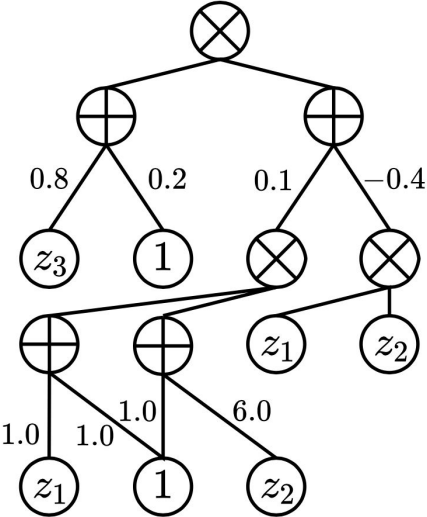


1. Sum nodes  $\oplus$  with weighted edges to children.
2. Product nodes  $\otimes$  with unweighted edges to children.
3. Leaf nodes:  $z_i$  or constant.

# PGCs Support Tractable Likelihoods

*How to extract the right monomial's coefficient?*

$\Pr(X_1 = 1, X_2 = 0, \dots) = ?$

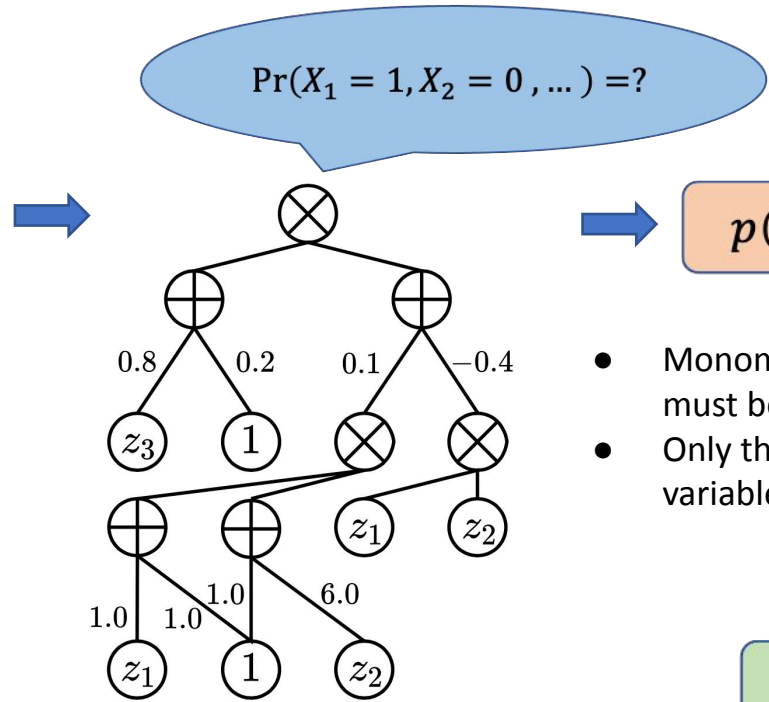


# PGCs Support Tractable Likelihoods

*How to extract the right monomial's coefficient?*

Purely symbolic

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \end{cases}$$



complexity  
 $O(\text{circuit size} \times \text{degree})$

$$p(t) = \alpha_k t^k + \dots + \alpha_1 t$$

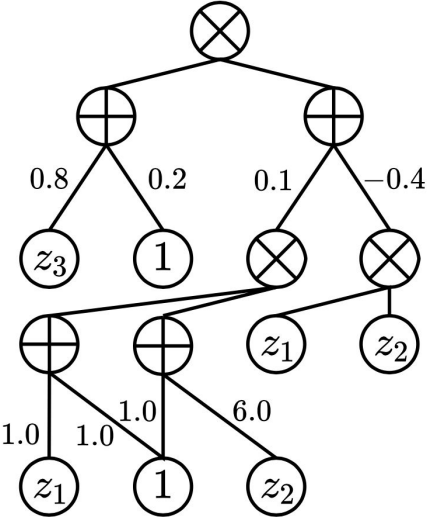
- Monomials setting to true variables that must be false are 0-ed out
- Only the monomial that sets all required variables to true has max degree.

$\alpha_k$  gives the answer

# PGCs Support Tractable Marginals

*How to sum the right monomial's coefficients?*

$\Pr(X_1 = 1, X_2 = 0, \dots) = ?$



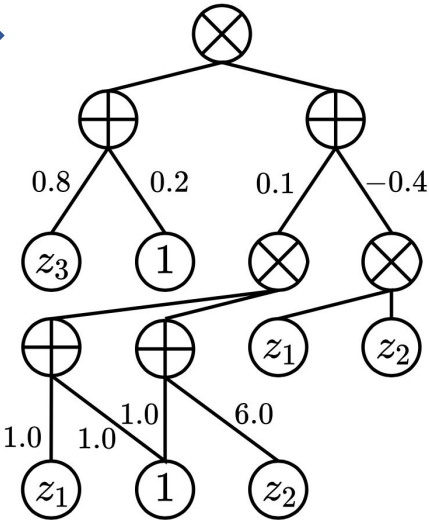
# PGCs Support Tractable Marginals

How to sum the right monomial's coefficients?

Purely symbolic

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

$\Pr(X_1 = 1, X_2 = 0, \dots) = ?$



$$p(t) = \alpha_k t^k + \dots + \alpha_1 t$$

- Monomials setting to true variables that must be false are 0-ed out
- Other monomials contribute to result.
- Only monomials that set all required variables to true have max degree.



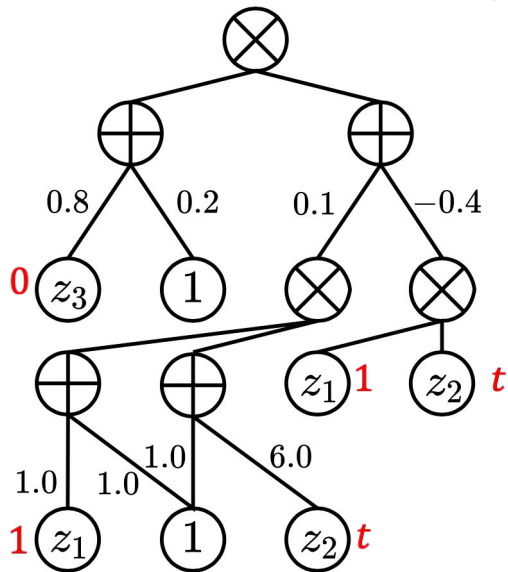
$\alpha_k$  gives the answer



# Example

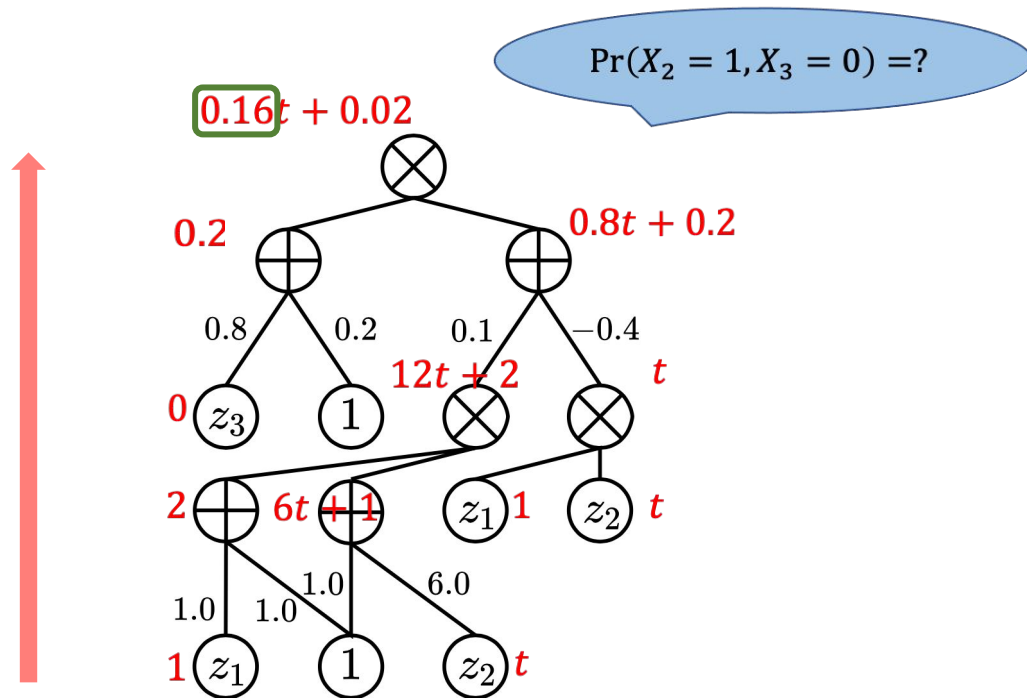
$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$

Pr( $X_2 = 1, X_3 = 0$ ) = ?



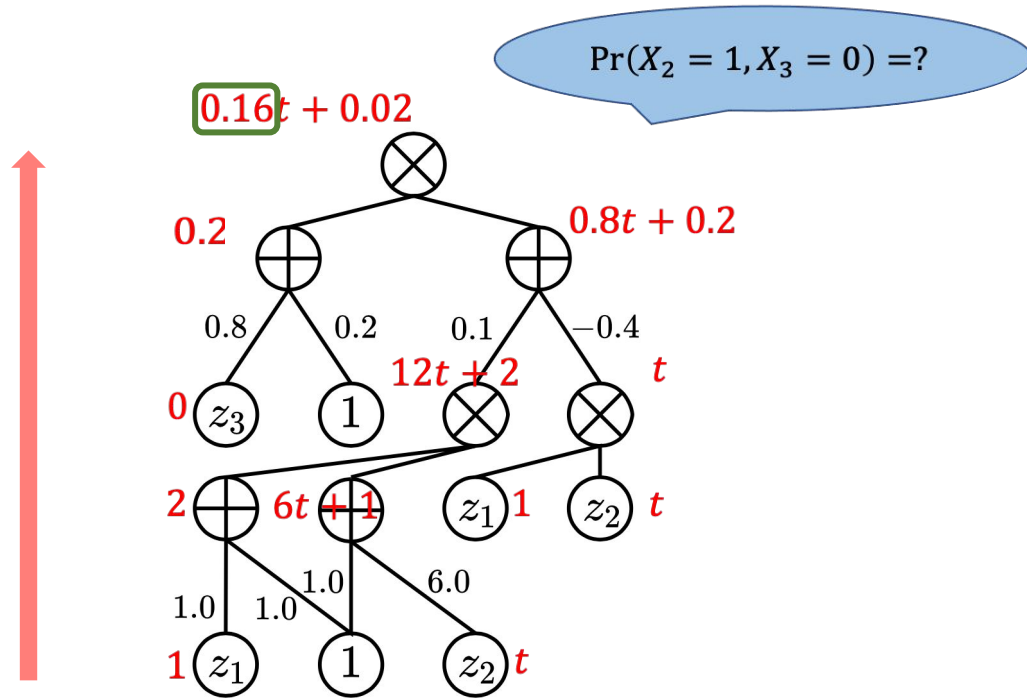
# Example

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$



# Example

$$z_i = \begin{cases} t & \text{if } X_i = 1 \\ 0 & \text{if } X_i = 0 \\ 1 & \text{if } X_i = ? \end{cases}$$



$X_1$	$X_2$	$X_3$	$\text{Pr}_\beta$
0	0	0	0.02
0	0	1	0.08
0	1	0	0.12
0	1	1	0.48
1	0	0	0.02
1	0	1	0.08
1	1	0	0.04
1	1	1	0.16

# Probabilistic circuits are probabilistic generating circuits

PCs represents probability mass functions:

$$m_{\beta} = 0.16X_1X_2X_3 + 0.04X_1X_2\bar{X}_3 + 0.08X_1\bar{X}_2X_3 + 0.02X_1\bar{X}_2\bar{X}_3 \\ + 0.48\bar{X}_1X_2X_3 + 0.12\bar{X}_1X_2\bar{X}_3 + 0.08\bar{X}_1\bar{X}_2X_3 + 0.02\bar{X}_1\bar{X}_2\bar{X}_3$$

PGCs represent probability generating functions:

$$g_{\beta} = 0.16z_1z_2z_3 + 0.04z_1z_2 + 0.08z_1z_3 + 0.02z_1 \\ + 0.48z_2z_3 + 0.12z_2 + 0.08z_1z_3 + 0.02$$

Given a smooth & decomposable PC, by setting  $\bar{X}_i$  to 1, and  $X_i$  to  $z_i$ , we obtain an equivalent PGC

# DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel  $L$  is given by:

$$g_L = \frac{1}{\det(L + I)} \det(I + L \text{diag}(z_1, \dots, z_n)).$$

We need it as a sum of products to obtain a Probabilistic Generating Circuit

# DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel  $L$  is given by:

$$g_L = \frac{1}{\det(L + I)} \det(I + L \text{diag}(z_1, \dots, z_n)).$$

Constant

We need it as a sum of products to obtain a Probabilistic Generating Circuit

# DPPs are probabilistic generating circuits

The generating polynomial for a DPP with kernel  $L$  is given by:

$$g_L = \frac{1}{\det(L + I)} \det(I + L \text{diag}(z_1, \dots, z_n)).$$

Constant

Division-free determinant algorithm  
(Samuelson-Berkowitz algorithm)

$g_L$  can be represented as a PGC of size  $O(n^4)$

# Beyond DPPs: Strongly Rayleigh Distributions

DPPs are strongly Rayleigh distributions

**Definition.** *A probability distribution over binary random variables  $X_1, \dots, X_n$  (or equivalently, subsets of  $[n] := \{1, 2, \dots, n\}$ ) is strongly Rayleigh if its probability generating polynomial  $g$  is real-stable; that is, for  $z_i \in \mathbb{C}$ , if  $\text{Im}(z_i) > 0$  for all  $z_i$ , then  $g(z_1, \dots, z_n) \neq 0$ .*

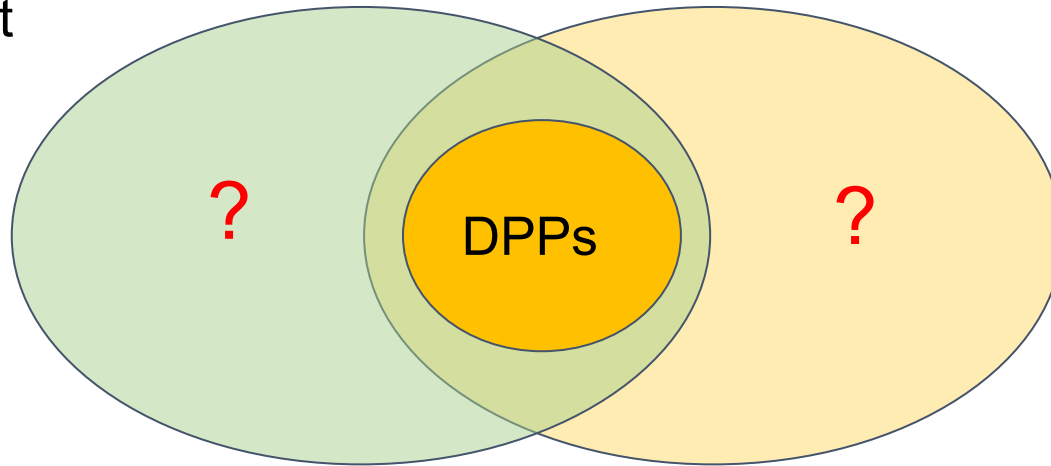
We can efficiently sample from strongly Rayleigh distributions by MCMC (with polynomial bound on mixing time)

**Theorem** (Li et al., 2016). *Let  $\pi$  be a strongly Rayleigh distribution over  $[n], \dots$*



# Relationship between PGCs and SR Distributions

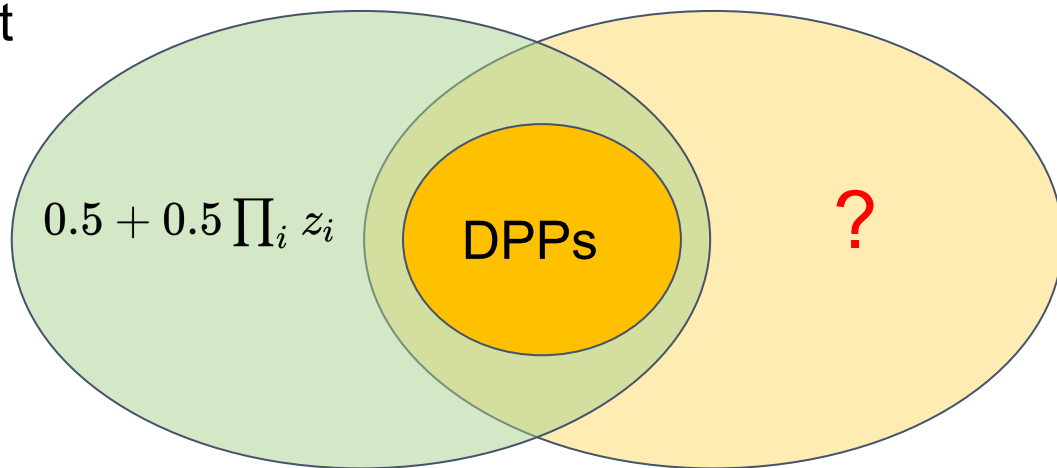
Compact  
PGCs



SR  
Distributions

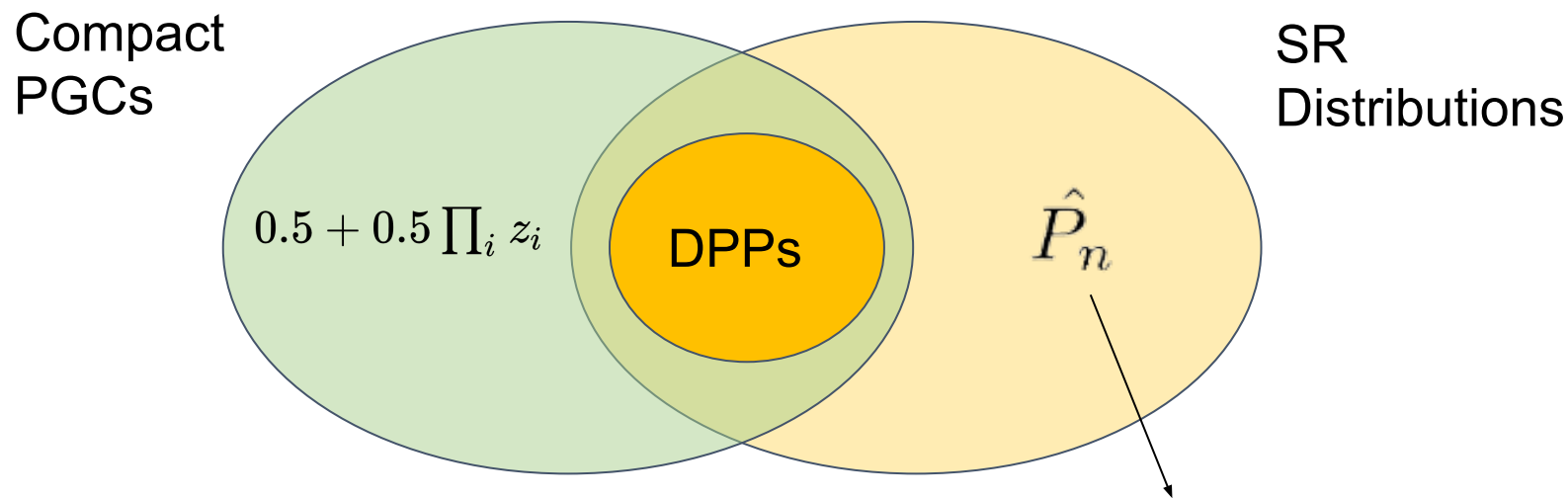
# Relationship between PGCs and SR Distributions

Compact  
PGCs



SR  
Distributions

# Relationship between PGCs and SR Distributions



**Theorem** (Bläser, 2023). *Assuming  $P^{\#P} \not\subseteq P/\text{Poly}$ . Let  $\hat{P}_n$  be the normalized  $P_n$ , then  $\hat{P}_n$  cannot be represented as polynomial-size PGCs.*

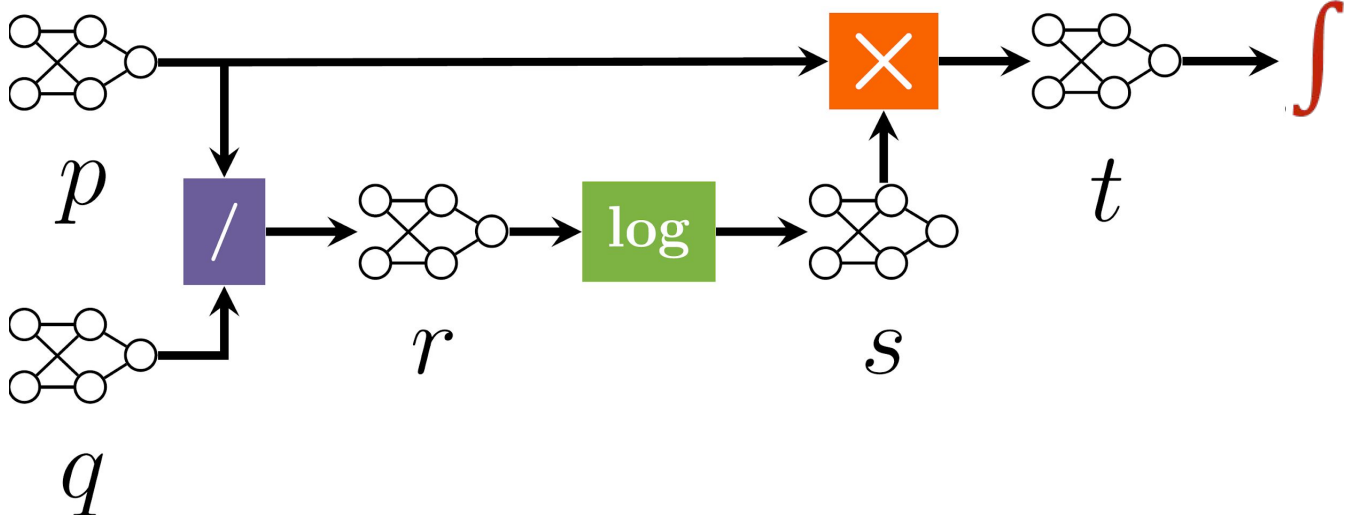
Probabilistic **generating** circuits seem awfully general.

*Are all tractable probabilistic models probabilistic **generating** circuits?*



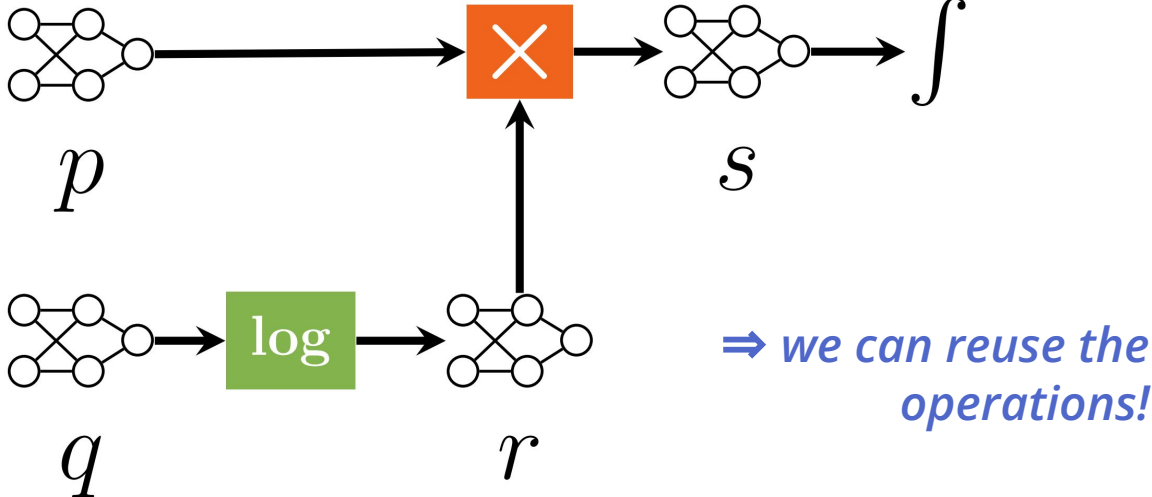
# Queries as pipelines: KLD

$$\text{KLD}(p \parallel q) = \int p(\mathbf{x}) \times \log((p(\mathbf{x})/q(\mathbf{x})))d\mathbf{X}$$



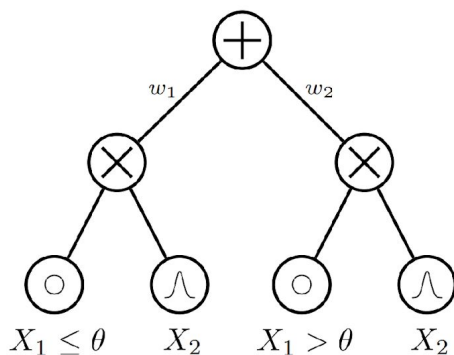
# Queries as pipelines: Cross Entropy

$$H(p, q) = \int p(\mathbf{x}) \times \log(q(\mathbf{x})) d\mathbf{X}$$



# Determinism

A sum node is **deterministic** if only one of its children outputs non-zero for any input

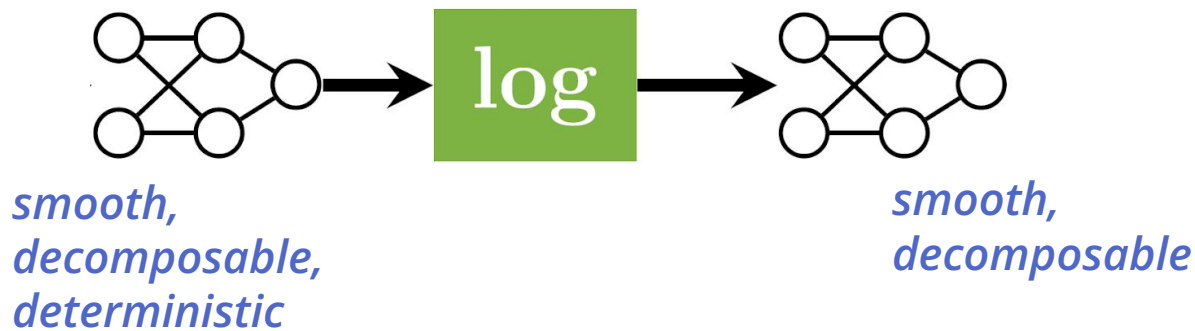


**deterministic circuit**

$\Rightarrow$  allows tractable MAP inference

$$\operatorname{argmax}_{\mathbf{x}} p(\mathbf{x})$$

Operation	Tractability	
	Input conditions	Output conditions
LOG	Sm, Dec, Det	Sm, Dec





# Tractable circuit operations

Operation		Tractability		Hardness
		Input properties	Output properties	
SUM	$\theta_1 p + \theta_2 q$	(+Cmp)	(+SD)	NP-hard for Det output
PRODUCT	$p \cdot q$	Cmp (+Det, +SD)	Dec (+Det, +SD)	#P-hard w/o Cmp
POWER	$p^n, n \in \mathbb{N}$	SD (+Det)	SD (+Det)	#P-hard w/o SD
	$p^\alpha, \alpha \in \mathbb{R}$	Sm, Dec, Det (+SD)	Sm, Dec, Det (+SD)	#P-hard w/o Det
QUOTIENT	$p/q$	Cmp; $q$ Det (+ $p$ Det,+SD)	Dec (+Det,+SD)	#P-hard w/o Det
LOG	$\log(p)$	Sm, Dec, Det	Sm, Dec	#P-hard w/o Det
EXP	$\exp(p)$	linear	SD	#P-hard

# Inference by tractable operations

*systematically derive* tractable inference algorithm of complex queries

	Query	Tract. Conditions	Hardness
CROSS ENTROPY	$-\int p(\mathbf{x}) \log q(\mathbf{x}) d\mathbf{X}$	Cmp, $q$ Det	#P-hard w/o Det
SHANNON ENTROPY	$-\sum p(\mathbf{x}) \log p(\mathbf{x})$	Sm, Dec, Det	coNP-hard w/o Det
RÉNYI ENTROPY	$(1 - \alpha)^{-1} \log \int p^\alpha(\mathbf{x}) d\mathbf{X}, \alpha \in \mathbb{N}$	SD	#P-hard w/o SD
MUTUAL INFORMATION	$(1 - \alpha)^{-1} \log \int p^\alpha(\mathbf{x}) d\mathbf{X}, \alpha \in \mathbb{R}_+$	Sm, Dec, Det	#P-hard w/o Det
KULLBACK-LEIBLER DIV.	$\int p(\mathbf{x}, \mathbf{y}) \log(p(\mathbf{x}, \mathbf{y}) / (p(\mathbf{x})p(\mathbf{y})))$	Sm, SD, Det*	coNP-hard w/o SD
RÉNYI'S ALPHA DIV.	$\int p(\mathbf{x}) \log(p(\mathbf{x}) / q(\mathbf{x})) d\mathbf{X}$	Cmp, Det	#P-hard w/o Det
ITAKURA-SAITO DIV.	$(1 - \alpha)^{-1} \log \int p^\alpha(\mathbf{x}) q^{1-\alpha}(\mathbf{x}) d\mathbf{X}, \alpha \in \mathbb{N}$	Cmp, $q$ Det	#P-hard w/o Det
CAUCHY-SCHWARZ DIV.	$(1 - \alpha)^{-1} \log \int p^\alpha(\mathbf{x}) q^{1-\alpha}(\mathbf{x}) d\mathbf{X}, \alpha \in \mathbb{R}$	Cmp, Det	#P-hard w/o Det
SQUARED LOSS	$\int [p(\mathbf{x}) / q(\mathbf{x}) - \log(p(\mathbf{x}) / q(\mathbf{x})) - 1] d\mathbf{X}$	Cmp, Det	#P-hard w/o Det
	$-\log \frac{\int p(\mathbf{x}) q(\mathbf{x}) d\mathbf{X}}{\sqrt{\int p^2(\mathbf{x}) d\mathbf{X} \int q^2(\mathbf{x}) d\mathbf{X}}}$	Cmp	#P-hard w/o Cmp
	$\int (p(\mathbf{x}) - q(\mathbf{x}))^2 d\mathbf{X}$	Cmp	#P-hard w/o Cmp

# Conclusions

1. What are probabilistic circuits?

*tractable deep generative models*

2. What are they useful for?

*controlling generative AI*

3. What is the underlying theory?

*probability generating polynomials*

# Thanks

*This was the work of many wonderful students/postdocs/collaborators!*



Honghua  
Zhang



Meihua  
Dang



Anji  
Liu

References: <http://starai.cs.ucla.edu/publications/>