UCLA Computer Science

STAR AI RESEARCH LAB UCLA

From Probabilistic Circuits to Probabilistic Programs and Back

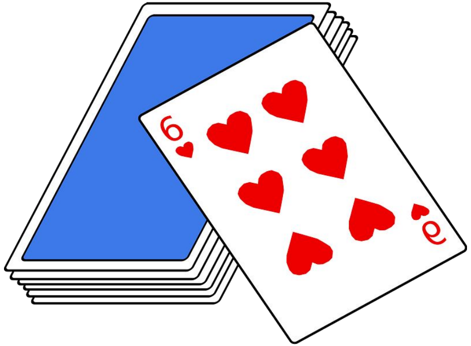
Guy Van den Broeck

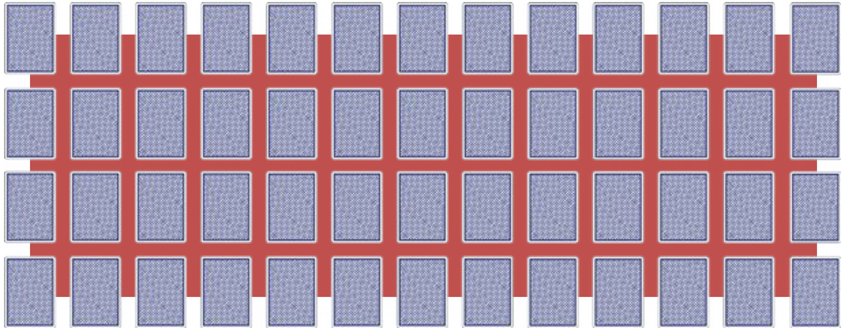
ICAART - Feb 6, 2021

Trying to be provocative

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.





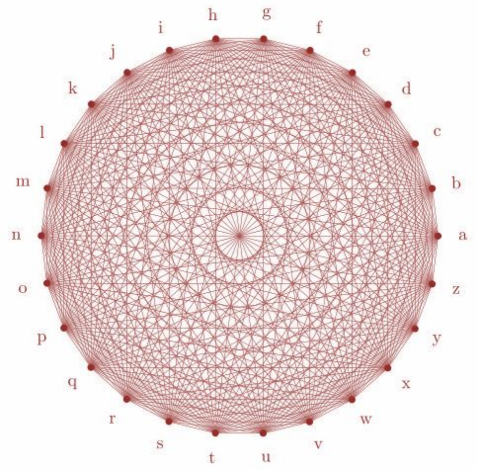
[VdB KRR15]

Trying to be provocative

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

3.14 Smokes(x) ∧ Friends(x,y)
⇒ Smokes(y)



[VdB KRR15]

Trying to be provocative

Probabilistic graphical models is how we do probabilistic AI!

Graphical models of variable-level (in)dependence are a broken abstraction.

Bean Machine

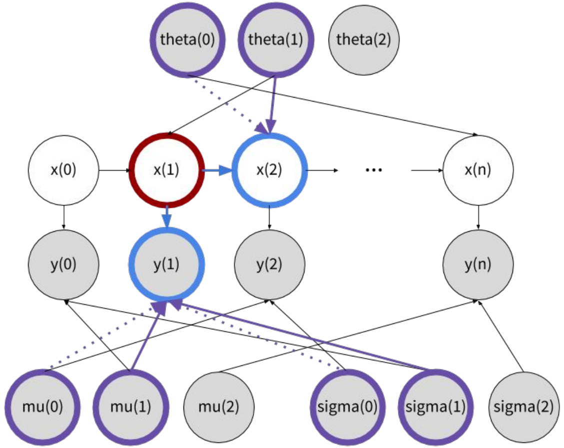
$$\mu_k \sim \text{Normal}(\alpha, \beta)$$

$$\sigma_k \sim \text{Gamma}(\nu, \rho)$$

$$\theta_k \sim \text{Dirichlet}(\kappa)$$

$$x_i \sim \begin{cases} \text{Categorical}(init) & \text{if } i = 0 \\ \text{Categorical}(\theta_{x_{i-1}}) & \text{if } i > 0 \end{cases}$$

$$y_i \sim \text{Normal}(\mu_{x_i}, \sigma_{x_i})$$



[Tehrani et al. PGM20]

Computational Abstractions

Let us think of probability distributions as objects that are computed.

Abstraction = Structure of Computation

Two examples:

1. Probabilistic Circuits
2. Probabilistic Programs

Probabilistic Circuits

Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

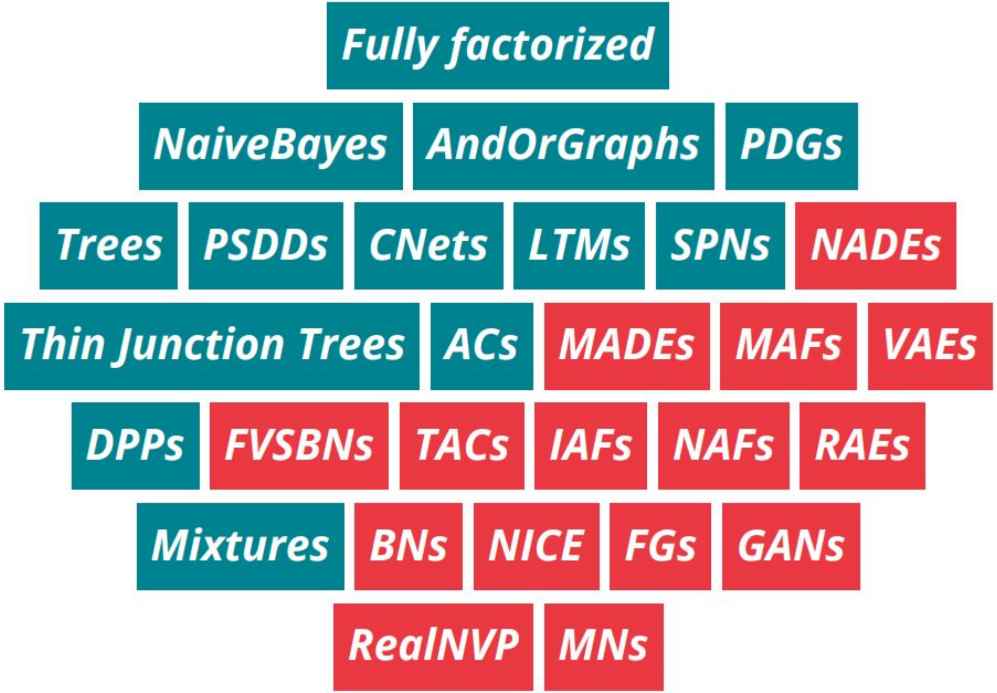
Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

The Alphabet Soup of probabilistic models



Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

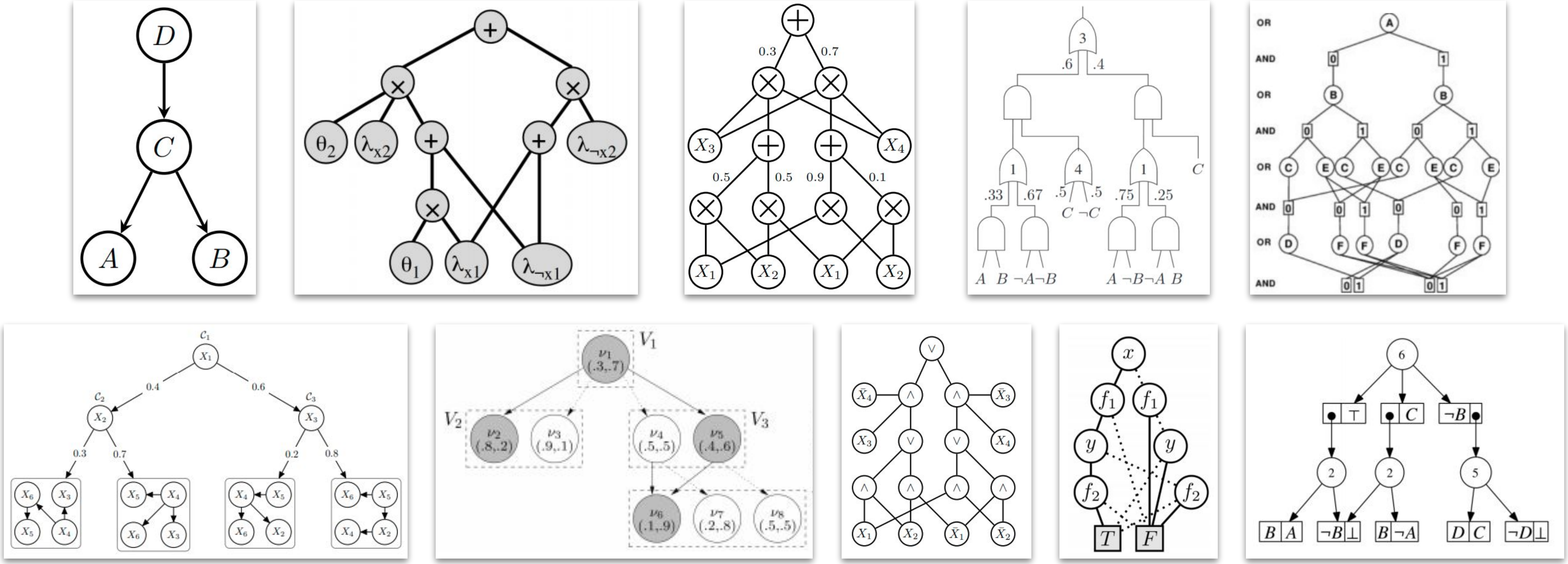
DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

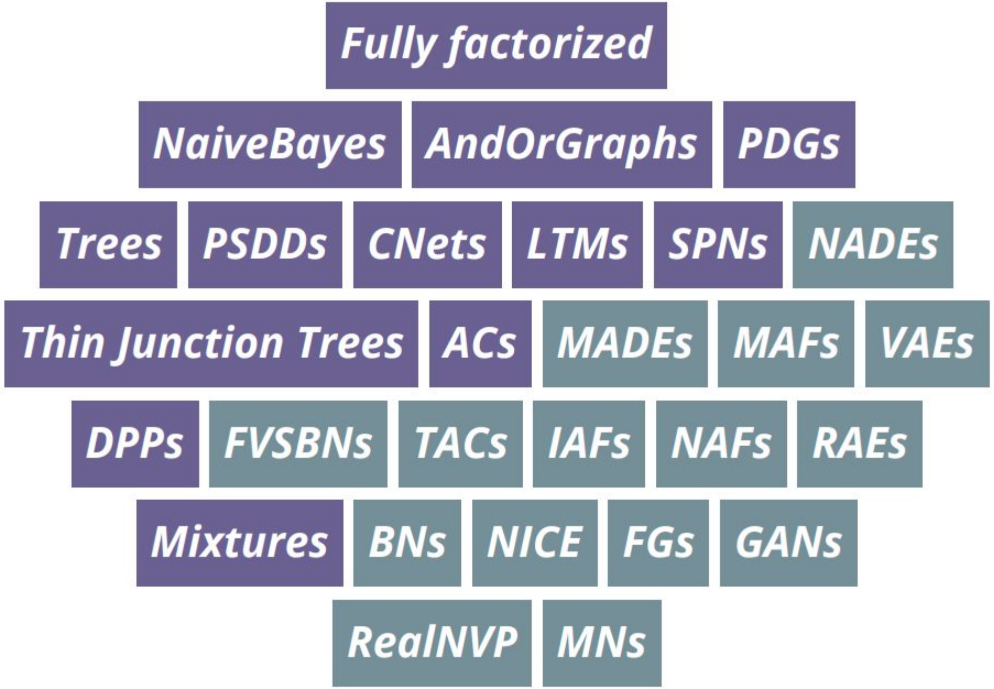
Intractable and *tractable* models

Tractable Probabilistic Models



"*Every talk needs a joke and a literature overview slide, not necessarily distinct*"

- after Ron Graham



Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

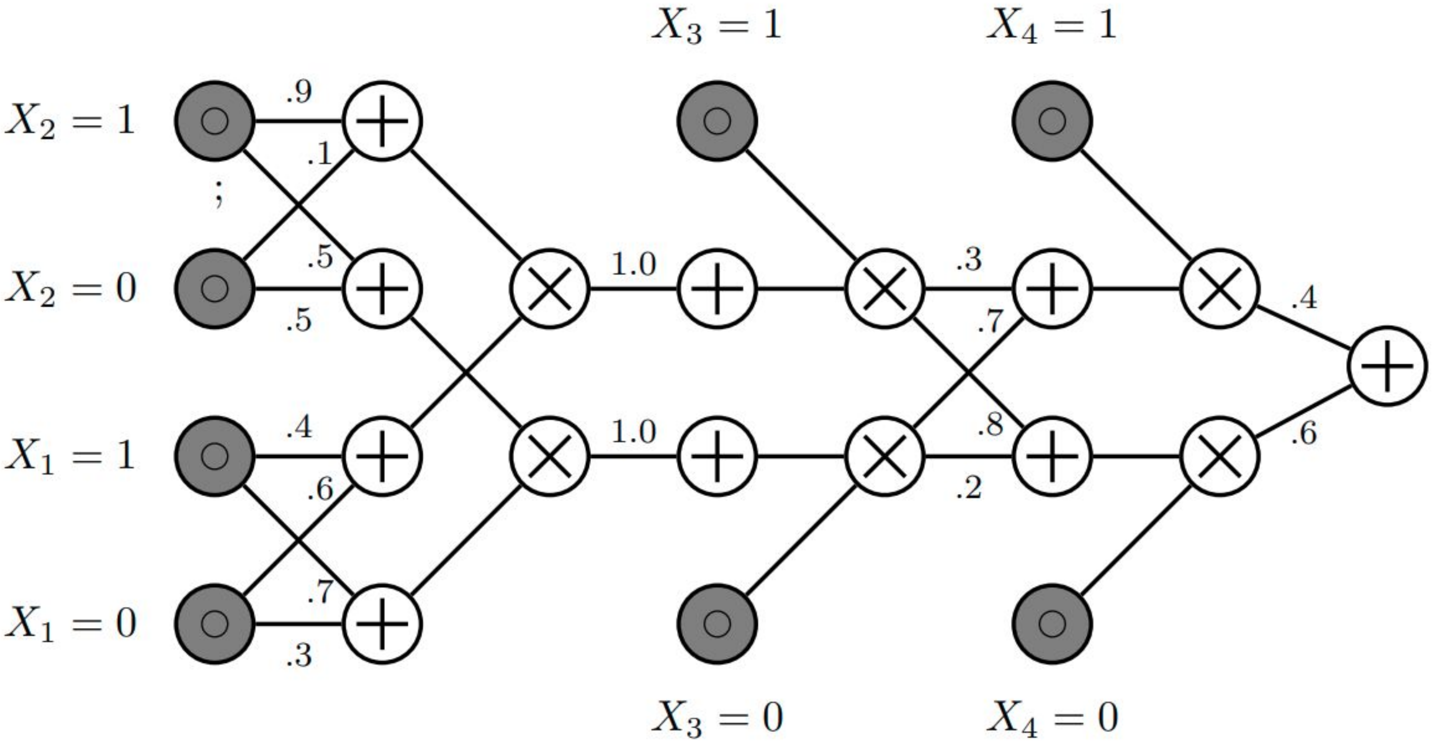
Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

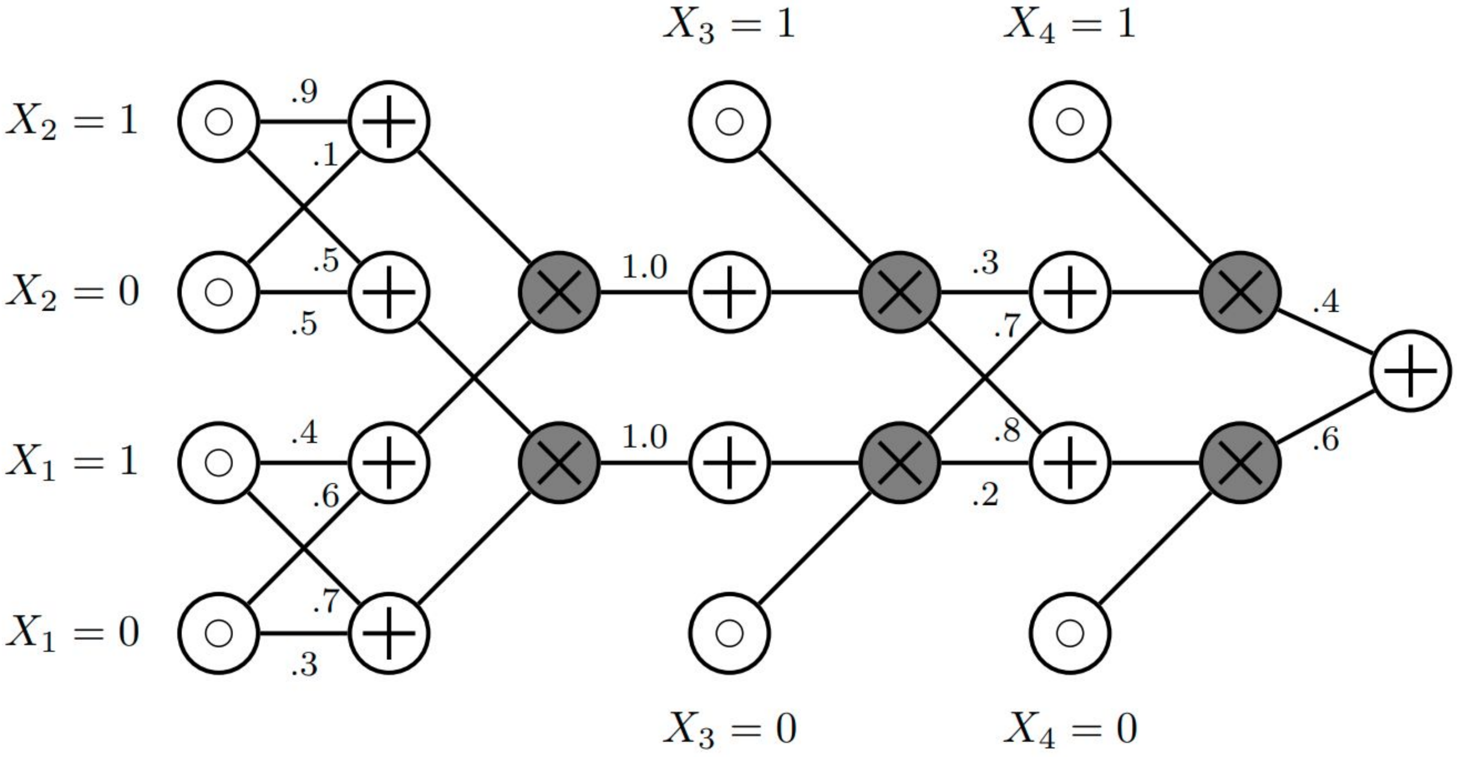
Mixtures BNs NICE FGs GANs

RealNVP MNs

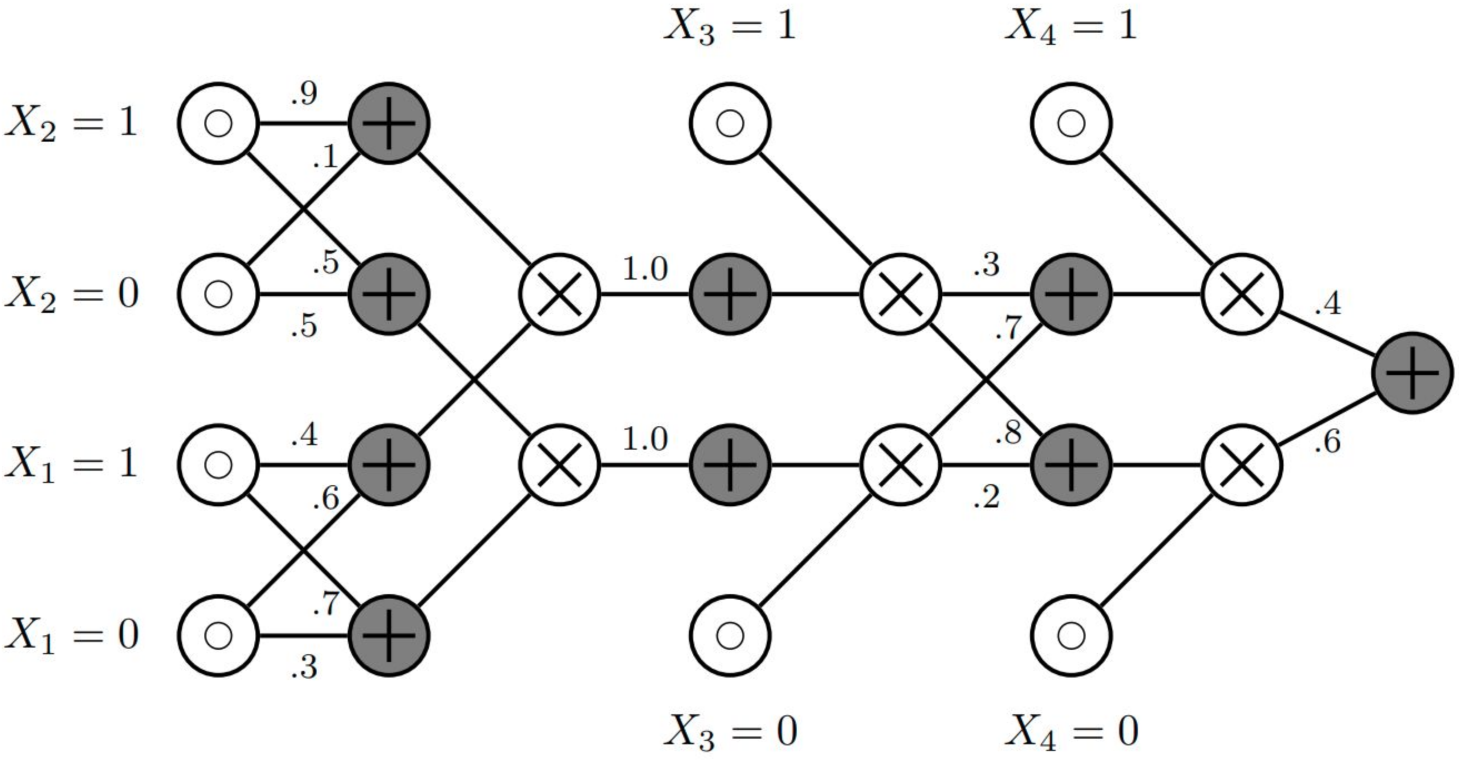
a *unifying framework* for tractable models



Input nodes c are tractable (simple) distributions, e.g., univariate gaussian or indicator $p_c(X=1) = [X=1]$



Product nodes are factorizations $\prod_{c \in \mathsf{in}(n)} \mathrm{p}_c(\mathbf{x})$



Sum nodes are mixture models $\sum_{c\in\mathsf{in}(n)} \theta_{n,c}\, \mathrm{p}_c(\mathbf{x})$

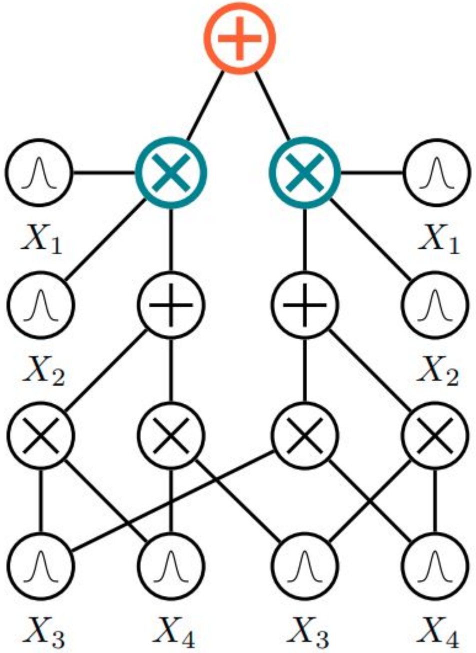
Smoothness + decomposability = tractable MAR

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*



[Darwiche & Marquis JAIR 2001, Poon & Domingos UAI11]

Smoothness + decomposability = tractable MAR

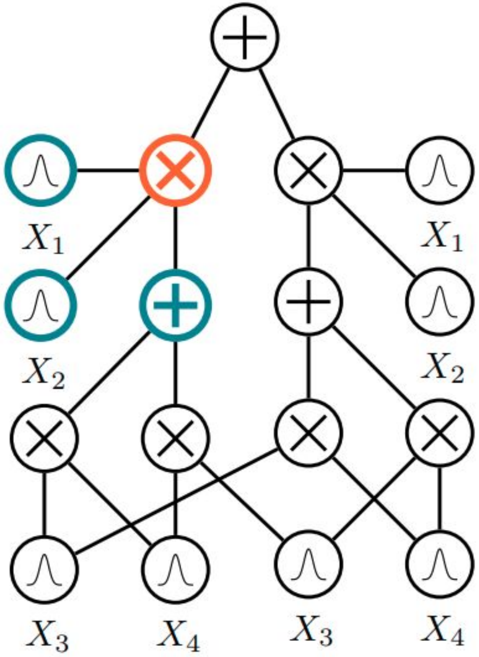
If $p(\mathbf{x},\mathbf{y},\mathbf{z}) = p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})$, (***decomposability***):

$$\int\int\int p(\mathbf{x},\mathbf{y},\mathbf{z})d\mathbf{x}d\mathbf{y}d\mathbf{z} =$$

$$= \int\int\int p(\mathbf{x})p(\mathbf{y})p(\mathbf{z})d\mathbf{x}d\mathbf{y}d\mathbf{z} =$$

$$= \int p(\mathbf{x})d\mathbf{x} \int p(\mathbf{y})d\mathbf{y} \int p(\mathbf{z})d\mathbf{z}$$

$\Rightarrow$ *integrals decompose into easier ones*



Smoothness + decomposability = tractable MAR

Forward pass evaluation for MAR

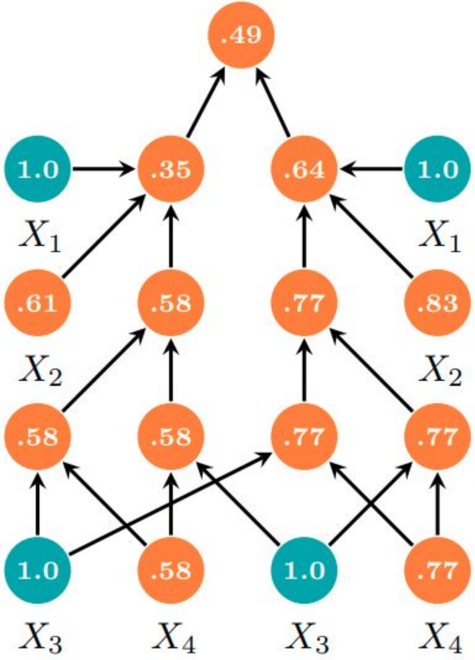
$\Rightarrow$ *linear in circuit size!*

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

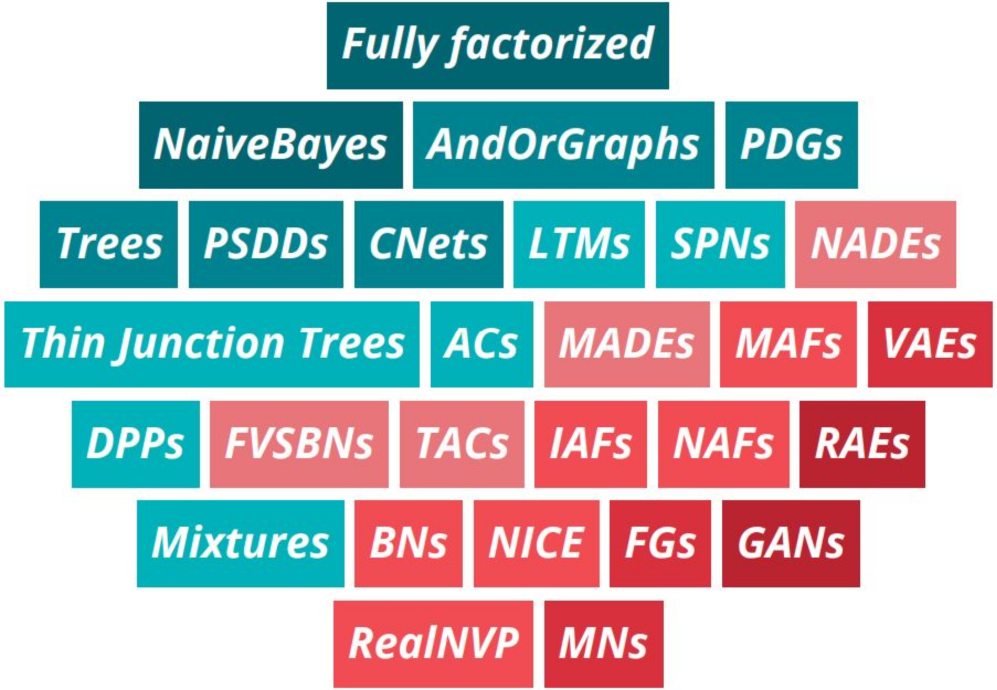
- leafs over X_1 and X_3 output $\boldsymbol{Z}_i = \int p(x_i) dx_i$

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

- leafs over X_2 and X_4 output ***EVI***
- feedforward evaluation (bottom-up)



$\mathcal{P}$ \ $\mathcal{Q}$:	MAR marginal queries	CON conditional queries	MOM moments (mean,...)	MAP maximum a posteriori	MMAP marginal MAP	ENT entropy	DIV divergences (KLD,...)	EXP expected predictions
smoothness SMO	✓	✓	✓	✗	✗	✓	✓	✓
decomposability DEC	✓	✓	✓	✗	✗	✗	✗	✗
consistency CON	✗	✗	✗	✓	✓	✗	✗	✗
determinism DET	✗	✗	✗	✓	✗	✗	✗	✗
marginal determinism MAR-DET	✗	✗	✗	✗	✓	✓	✓	✗
structured decomposability STR-DEC	✗	✗	✗	✗	✗	✓	✗	✗
paired str. decomposability P-STR-DEC	✗	✗	✗	✗	✗	✗	✓	✓



Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

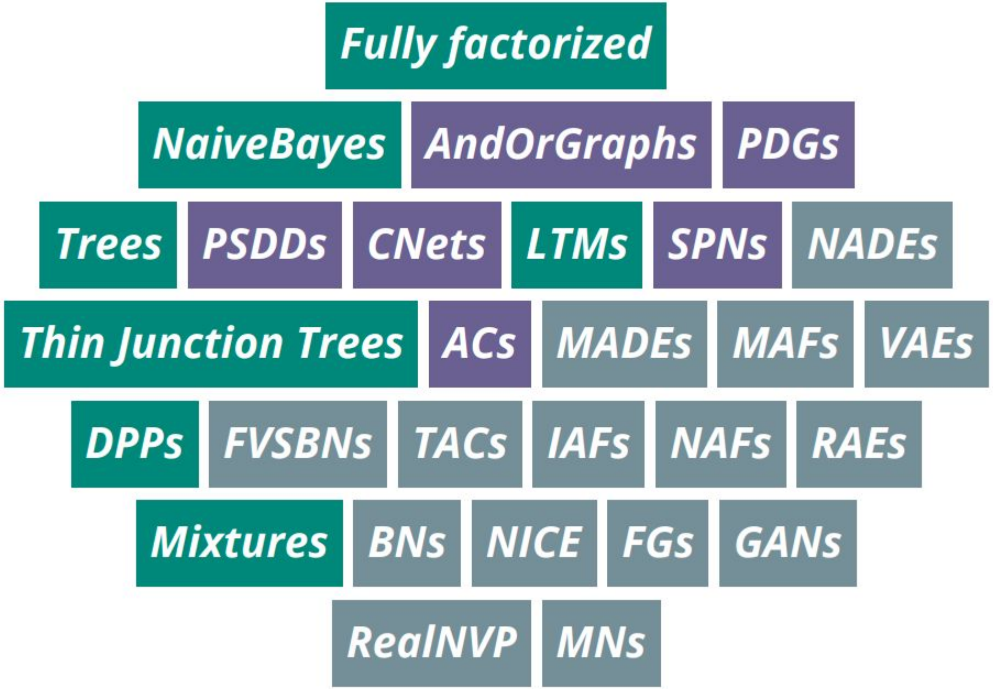
DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

RealNVP MNs

tractability is a spectrum

	smooth	*dec.*	det.	str.dec.
Arithmetic Circuits (ACs) *[Darwiche 2003]*	✓	✓	✓	✗
Sum-Product Networks (SPNs) *[Poon et al. 2011]*	✓	✓	✗	✗
Cutset Networks (CNets) *[Rahman et al. 2014]*	✓	✓	✓	✗
Probabilistic Decision Graphs *[Jaeger 2004]*	✓	✓	✓	✓
(Affine) ADDs *[Hoey et al. 1999; Sanner et al. 2005]*	✓	✓	✓	✓
AndOrGraphs *[Dechter et al. 2007]*	✓	✓	✓	✓
PSDDs *[Kisa et al. 2014a]*	✓	✓	✓	✓



Fully factorized

NaiveBayes AndOrGraphs PDGs

Trees PSDDs CNets LTMs SPNs NADEs

Thin Junction Trees ACs MADEs MAFs VAEs

DPPs FVSBNs TACs IAFs NAFs RAEs

Mixtures BNs NICE FGs GANs

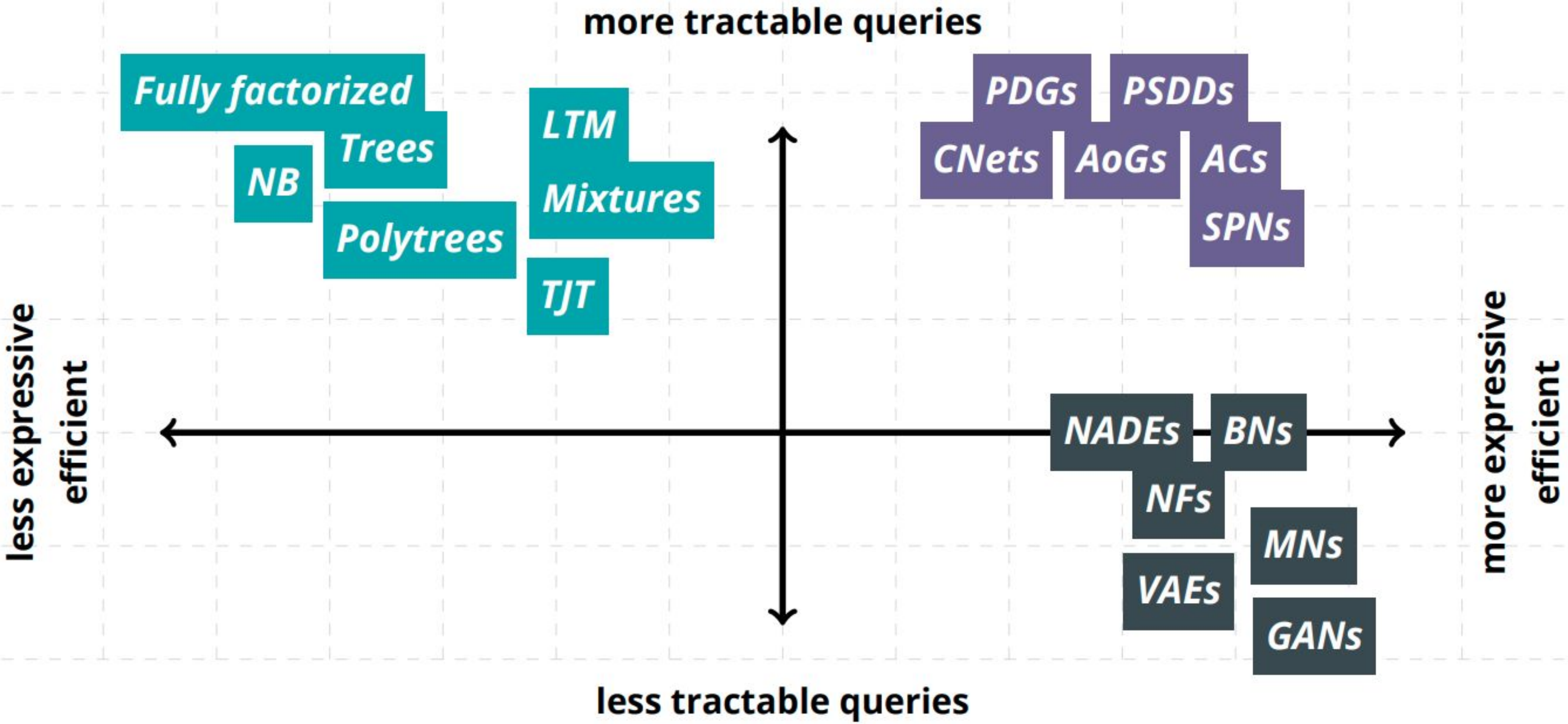
RealNVP MNs

Expressive **models without** ***compromises***

How expressive are probabilistic circuits?

density estimation benchmarks

dataset	best circuit	BN	MADE	VAE	dataset	best circuit	BN	MADE	VAE
nltcs	**-5.99**	-6.02	-6.04	**-5.99**	***dna***	**-79.88**	-80.65	-82.77	-94.56
msnbc	**-6.04**	**-6.04**	-6.06	-6.09	***kosarek***	**-10.52**	-10.83	-	-10.64
kdd	-2.12	-2.19	**-2.07**	-2.12	***msweb***	-9.62	-9.70	**-9.59**	-9.73
plants	**-11.84**	-12.65	-12.32	-12.34	***book***	-33.82	-36.41	-33.95	**-33.19**
audio	-39.39	-40.50	-38.95	**-38.67**	***movie***	-50.34	-54.37	-48.7	**-47.43**
jester	-51.29	**-51.07**	-52.23	-51.54	***webkb***	-149.20	-157.43	-149.59	**-146.9**
netflix	-55.71	-57.02	-55.16	**-54.73**	***cr52***	-81.87	-87.56	-82.80	**-81.33**
accidents	-26.89	**-26.32**	-26.42	-29.11	***c20ng***	-151.02	-158.95	-153.18	**-146.9**
retail	**-10.72**	-10.87	-10.81	-10.83	***bbc***	**-229.21**	-257.86	-242.40	-240.94
pumbs*	-22.15	**-21.72**	-22.3	-25.16	***ad***	-14.00	-18.35	**-13.65**	-18.81



more tractable queries

Fully factorized

Trees

NB

Polytrees

LTM

Mixtures

TJT

PDGs

PSDDs

CNets

AoGs

ACs

SPNs

less expressive
efficient

more expressive
efficient

NADEs

BNs

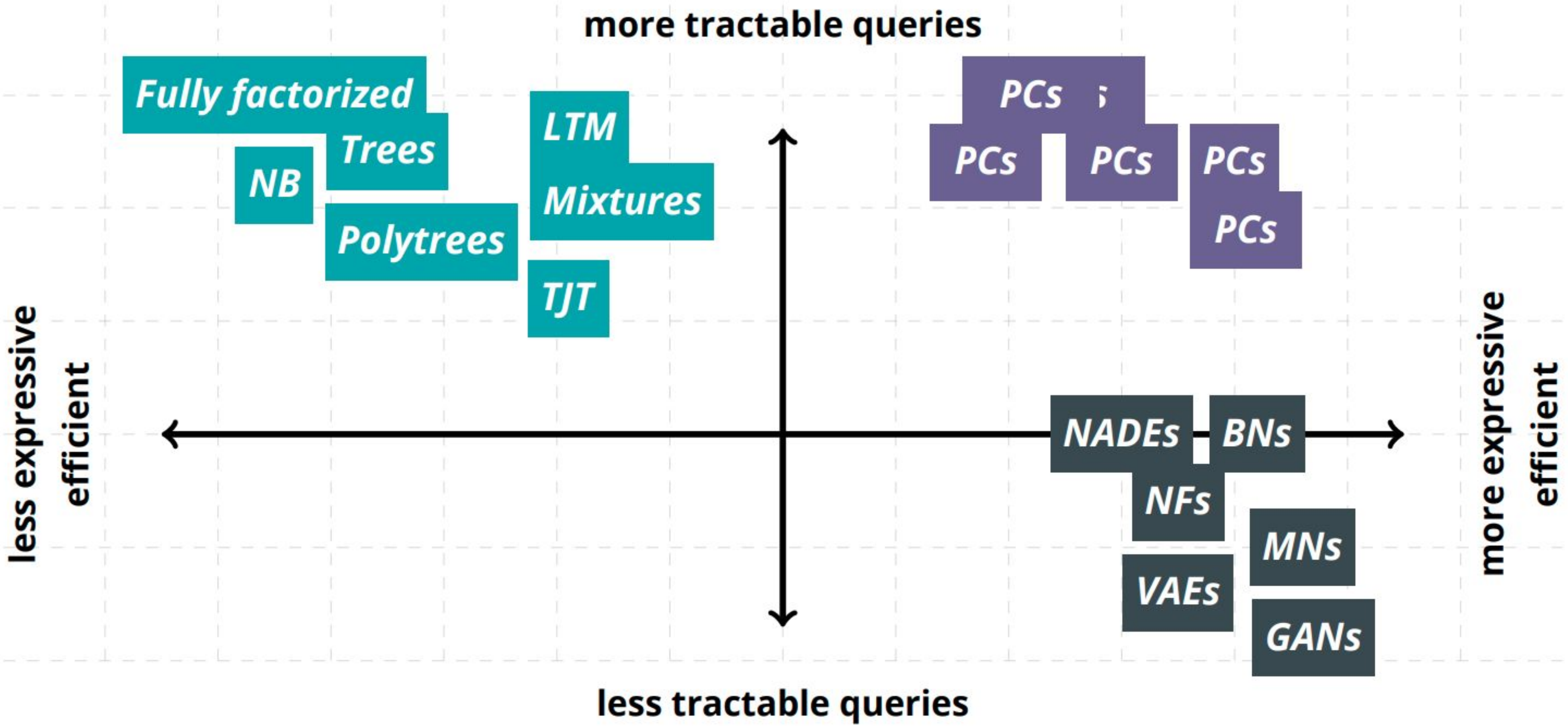
NFs

MNs

VAEs

GANs

less tractable queries



more tractable queries

Fully factorized

Trees

NB

Polytrees

LTM

Mixtures

TJT

PCs

PCs

PCs

PCs

PCs

less expressive
efficient

more expressive
efficient

NADEs

BNs

NFs

MNs

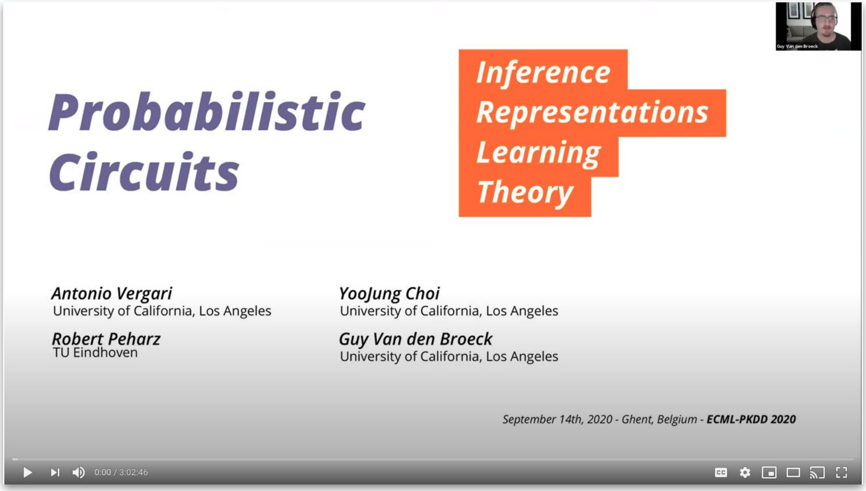
VAEs

GANs

less tractable queries

Want to learn more?

Tutorial (3h)



Probabilistic Circuits

Inference
Representations
Learning
Theory

Antonio Vergari
University of California, Los Angeles

YooJung Choi
University of California, Los Angeles

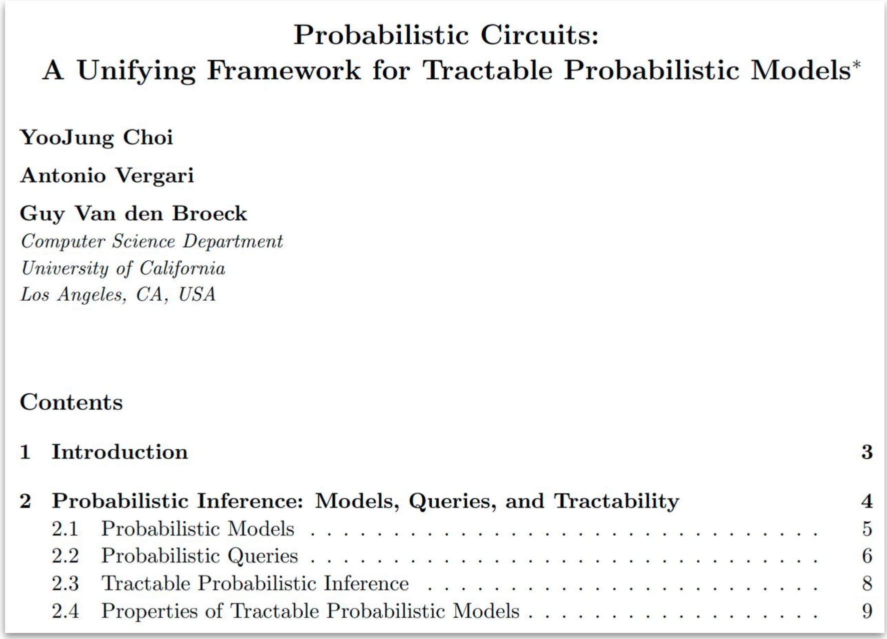
Robert Peharz
TU Eindhoven

Guy Van den Broeck
University of California, Los Angeles

September 14th, 2020 - Ghent, Belgium - **ECML-PKDD 2020**

https://youtu.be/2RAG5-L9R70

Overview Paper (80p)



Probabilistic Circuits:
A Unifying Framework for Tractable Probabilistic Models*

YooJung Choi
Antonio Vergari
Guy Van den Broeck
Computer Science Department
University of California
Los Angeles, CA, USA

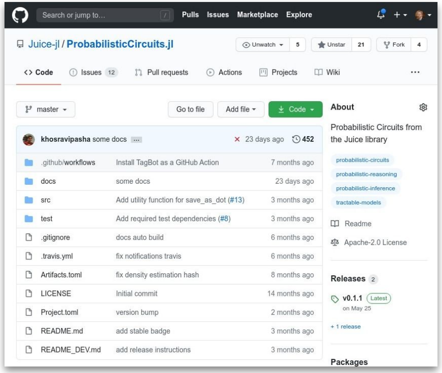
Contents

http://starai.cs.ucla.edu/papers/ProbCirc20.pdf

Training PCs in Julia with Juice.jl

Training maximum likelihood parameters of probabilistic circuits

```
julia> using ProbabilisticCircuits;
julia> data, structure = load(...);
julia> num_examples(data)
17,412
julia> num_edges(structure)
270,448
julia> @btime estimate_parameters(structure , data);
63 ms
```



Juice-jl / ProbabilisticCircuits.jl

khosravipasha some docs ... 23 days ago 452

.github/workflows	Install TagBot as a GitHub Action	7 months ago
docs	some docs	23 days ago
src	Add utility function for save_as_dot (#13)	3 months ago
test	Add required test dependencies (#8)	3 months ago
.gitignore	docs auto build	6 months ago
.travis.yml	fix notifications travis	6 months ago
Artifacts.toml	fix density estimation hash	8 months ago
LICENSE	Initial commit	14 months ago
Project.toml	version bump	2 months ago
README.md	add stable badge	3 months ago
README_DEV.md	add release instructions	3 months ago

About

Probabilistic Circuits from the Juice library

probabilistic-circuits
probabilistic-reasoning
probabilistic-inference
tractable-models

Readme

Apache-2.0 License

Releases 2

v0.1.1 Latest
on May 25

+ 1 release

Packages

Custom SIMD and CUDA kernels to parallelize over layers and training examples.

[https://github.com/Juice-jl/]

Probabilistic circuits seem awfully general.

Are all tractable probabilistic models probabilistic circuits?

Determinantal Point Processes (DPPs)

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

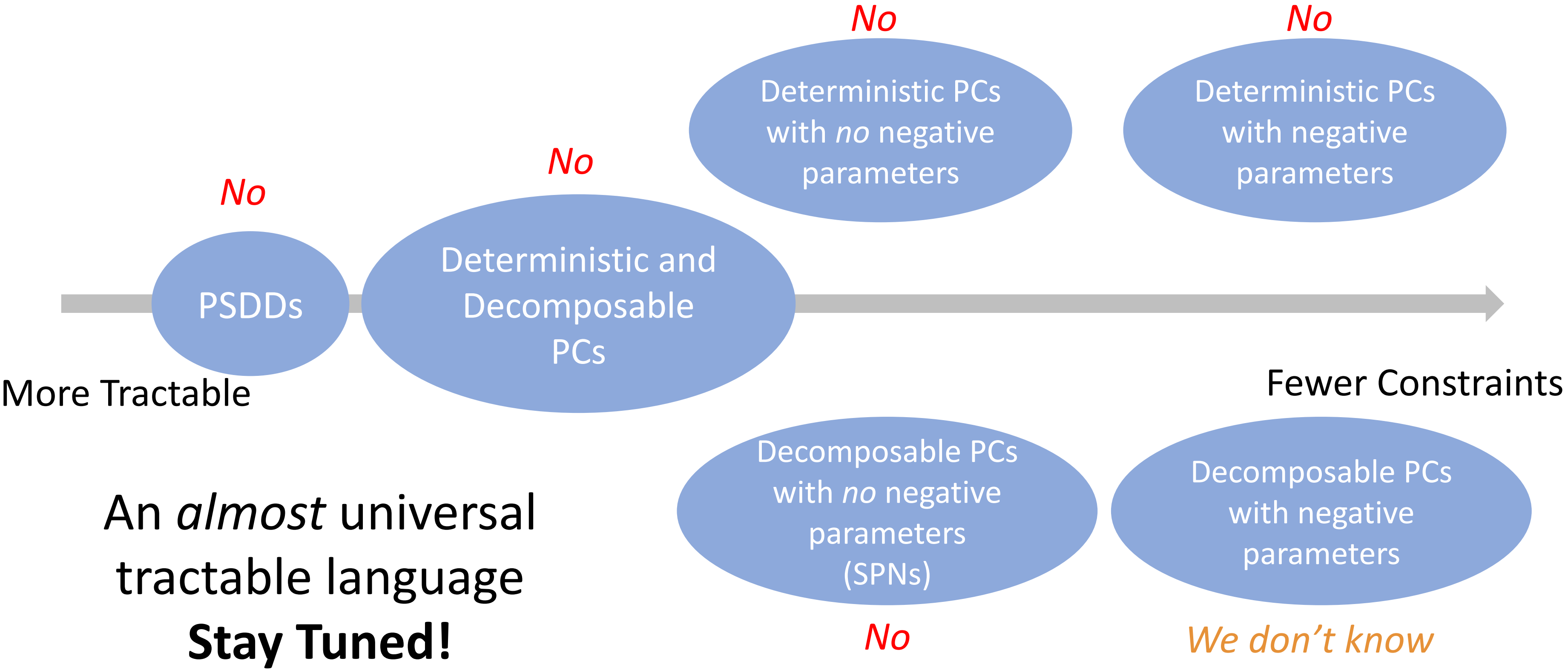
$$L = \begin{bmatrix} 1 & 0.9 & 0.8 & 0 \\ 0.9 & 0.97 & 0.96 & 0 \\ 0.8 & 0.96 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Computing marginal probabilities is *tractable.*

[Zhang et al. UAI20]

We cannot tractably represent DPPs with classes of PCs … yet



No

PSDDs

No

Deterministic and Decomposable PCs

No

Deterministic PCs with *no* negative parameters

No

Deterministic PCs with negative parameters

Decomposable PCs with *no* negative parameters (SPNs)

No

Decomposable PCs with negative parameters

We don't know

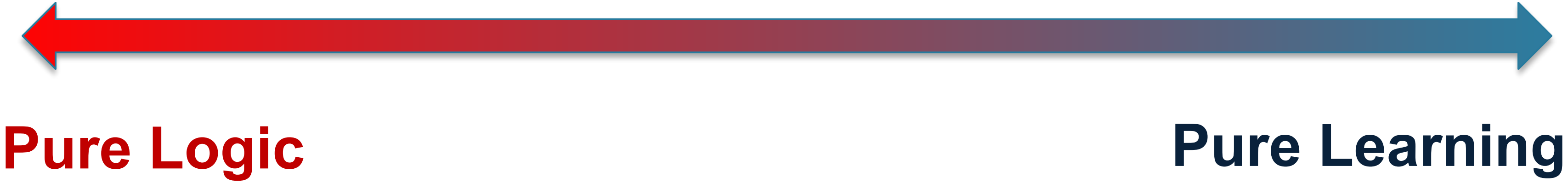
More Tractable

Fewer Constraints

An *almost* universal tractable language
Stay Tuned!

[Zhang et al. UAI20; Martens & Medabalimi Arxiv15]

The AI Dilemma



Pure Logic

Pure Learning

The AI Dilemma

Pure Logic

Pure Learning

- Slow thinking: deliberative, cognitive, model-based, extrapolation
- Amazing achievements until this day
- “*Pure logic is brittle*”

  noise, uncertainty, incomplete knowledge, …

The AI Dilemma

Pure Logic

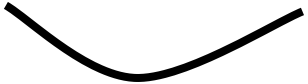
Pure Learning

- Fast thinking: instinctive, perceptive, model-free, interpolation
- Amazing achievements recently
- “*Pure learning is brittle*”

  bias, algorithmic fairness, interpretability, explainability, adversarial attacks, unknown unknowns, calibration, verification, missing features, missing labels, data efficiency, shift in distribution, general robustness and safety

  fails to incorporate a sensible model of the world

Pure Logic Probabilistic World Models Pure Learning



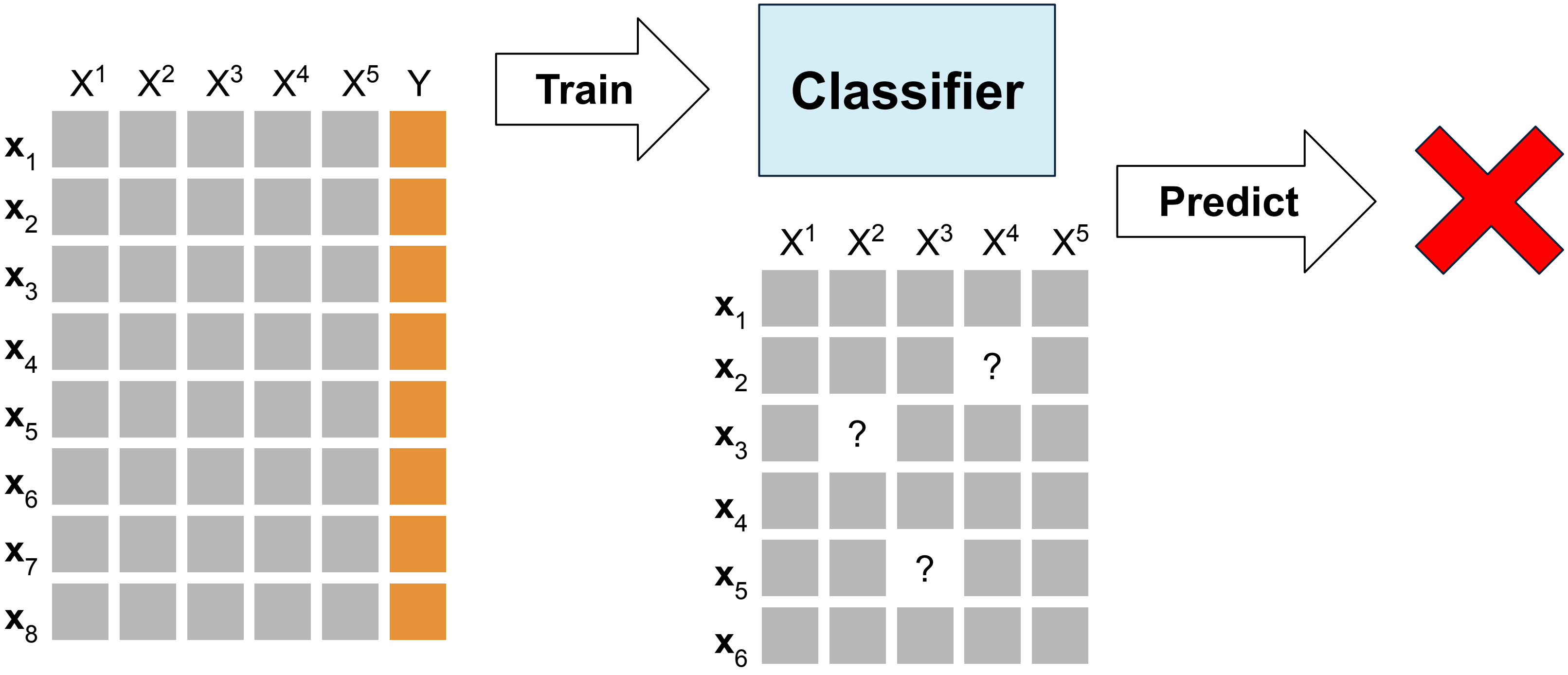
A New Synthesis of Learning and Reasoning

- “*Pure learning is brittle*”

bias, **algorithmic fairness**, interpretability, **explainability**, adversarial attacks, unknown unknowns, calibration, verification, **missing features**, missing labels, data efficiency, shift in distribution, general robustness and safety

fails to incorporate a sensible model of the world

Prediction with Missing Features



Test with missing features

Expected Predictions

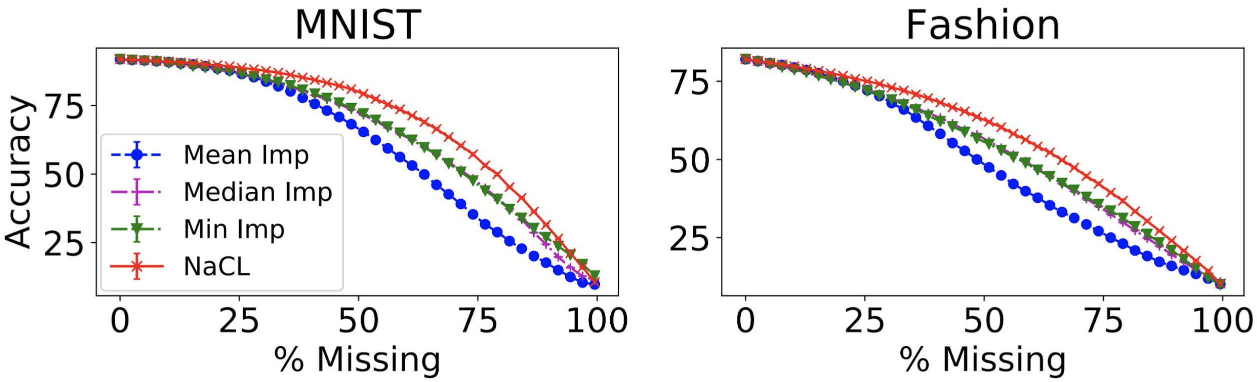
Consider **all possible complete inputs** and **reason** about the *expected* behavior of the classifier

$$\mathbb{E}_{\mathbf{x}^m \sim p(\mathbf{x}^m|\mathbf{x}^o)}\left[f\left(\mathbf{x}^m\mathbf{x}^o\right)\right]$$

$\boldsymbol{x}^{\mathrm{o}}$ = observed features
$\boldsymbol{x}^{\mathrm{m}}$ = missing features

Experiment:

- *f(x)* = logistic regres.
- *p(x)* = naive Bayes



[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]

What about complex feature distributions?

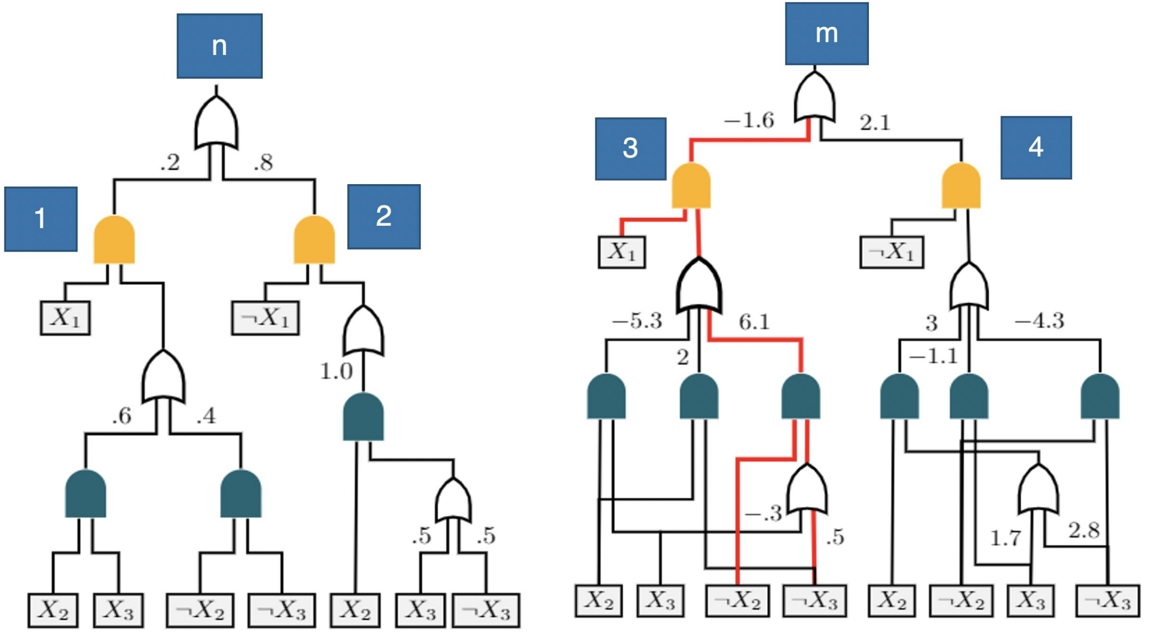
- feature distribution is a probabilistic circuits
- classifier is a compatible regression circuit

☺

Recursion that “breaks down” the computation.

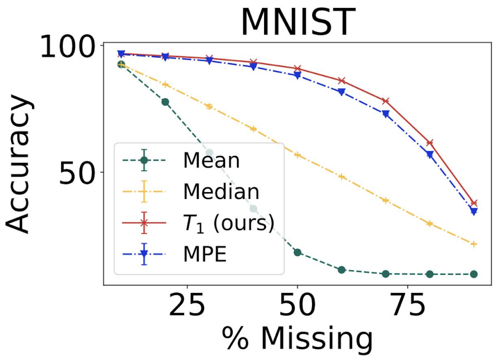
Expectation of function m w.r.t. dist. n ?

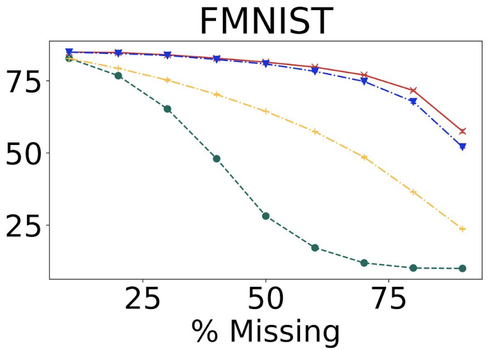
Solve subproblems: (1,3), (1,4), (2,3), (2,4)

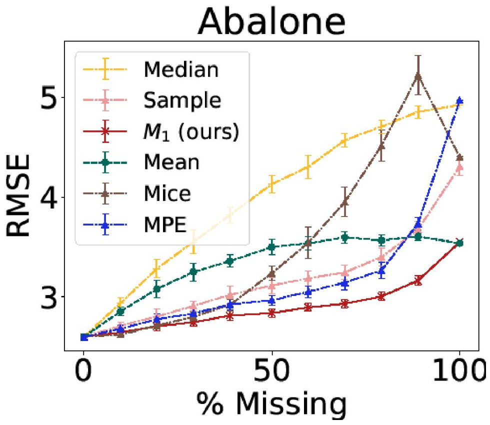


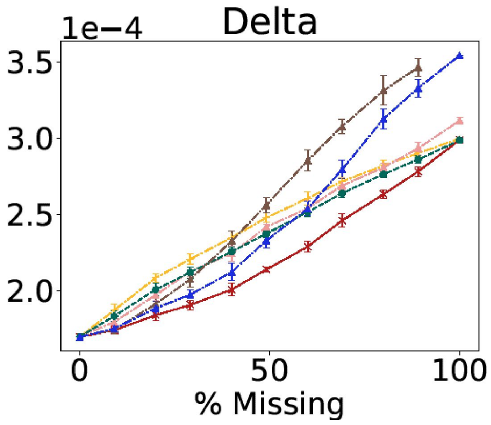
[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]

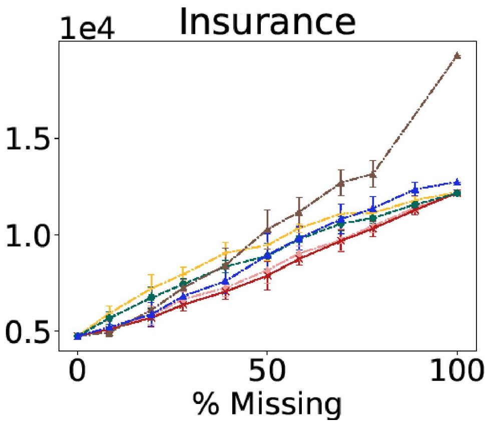
Probabilistic Circuits for Missing Data











[Khosravi et al. IJCAI19, NeurIPS20, Artemiss20]

ADV inference in Julia with Juice.jl

```julia
using ProbabilisticCircuits
pc = load_prob_circuit(zoo_psdd_file("insurance.psdd"));
rc = load_logistic_circuit(zoo_lc_file("insurance.circuit"), 1);
```

Is the predictive model biased by gender?

```julia
groups = make_observations([["male"], ["female"]])
exps, _ = Expectation(pc, rc, groups);
println("Female  : \$ $(exps[2])");
println("Male    : \$ $(exps[1])");
println("Diff    : \$ $(exps[2] - exps[1])");
Female  : $ 14170.125469335406
Male    : $ 13196.548926381849
Diff    : $ 973.5765429535568
```

Model-Based Algorithmic Fairness: FairPC

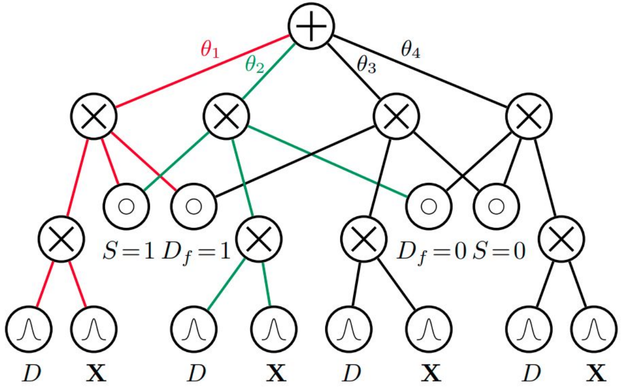
Learn classifier given

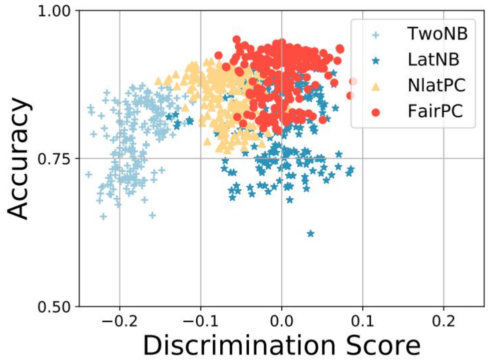
- features S and X
- training labels/decisions D

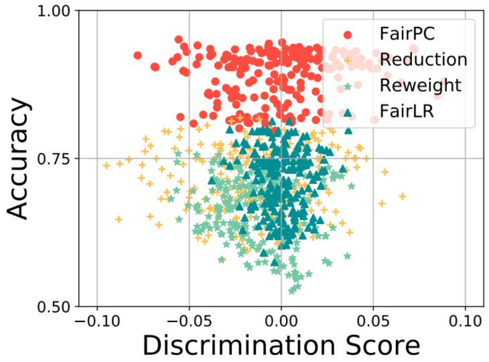
Group fairness by demographic parity:

Fair decision D_f should be independent of the sensitive attribute S

Discover the latent fair decision D_f by learning a PC.







[Choi et al. AAAI21]

Probabilistic Sufficient Explanations

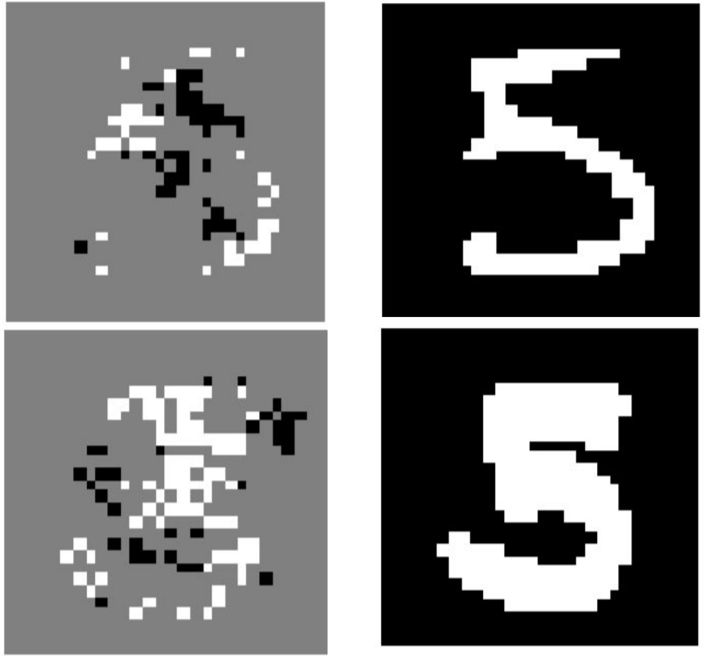
<u>Goal</u>: explain an instance of classification (a specific prediction)

Explanation is a subset of features, s.t.

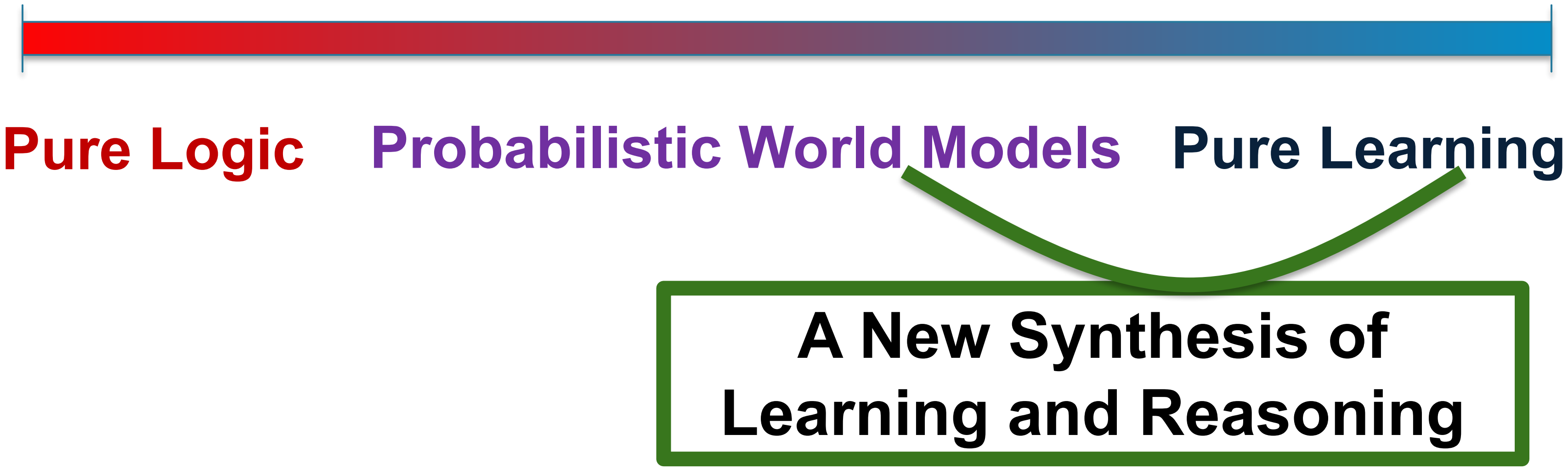
1. The explanation is "probabilistically sufficient"

   Under the feature distribution, given the explanation, the classifier is likely to make the observed prediction.

2. It is minimal and "simple"



[Khosravi et al. IJCAI19, Wang et al. XXAI20]



Pure Logic Probabilistic World Models Pure Learning

A New Synthesis of Learning and Reasoning

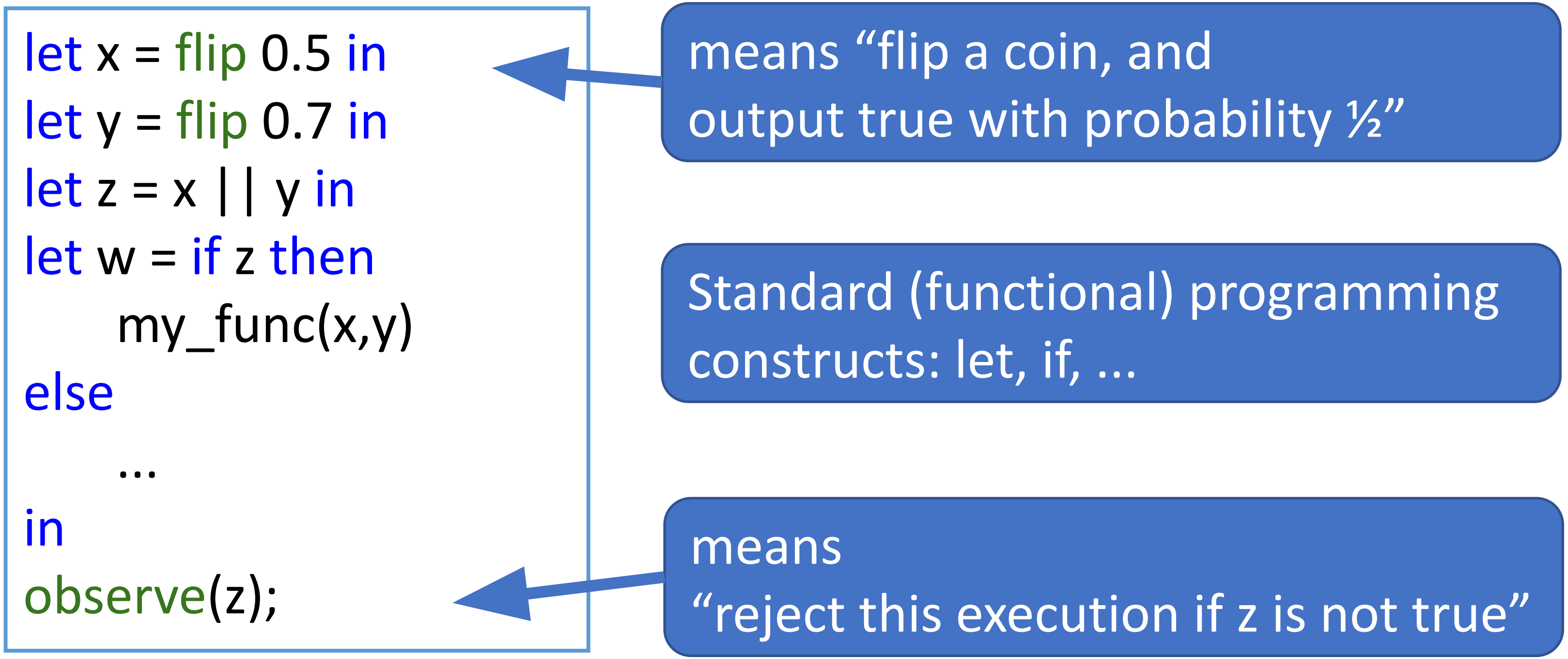
“*Pure learning is brittle*”

bias, **algorithmic fairness**, interpretability, **explainability**, adversarial attacks, unknown unknowns, calibration, verification, **missing features**, missing labels, data efficiency, shift in distribution, general robustness and safety

We need to incorporate a sensible probabilistic model of the world

Probabilistic Programs

What are probabilistic programs?



```
let x = flip 0.5 in
let y = flip 0.7 in
let z = x || y in
let w = if z then
      my_func(x,y)
else
      ...
in
observe(z);
```

means “flip a coin, and output true with probability ½”

Standard (functional) programming constructs: let, if, ...

means “reject this execution if z is not true”

Why Probabilistic Programming?

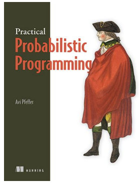
PPLs are proliferating

Pyro

Edward

HackPPL

Stan


Practical Probabilistic Programming

Avi Pfeffer

Figaro

Venture, Church, IBAL, WebPPL, Infer.NET, Tensorflow Probability, ProbLog, PRISM, LPADs, CPLogic, CLP(BN), ICL, PHA, Primula, Storm, Gen, PRISM, PSI, Bean Machine, etc.   *… and many many more*

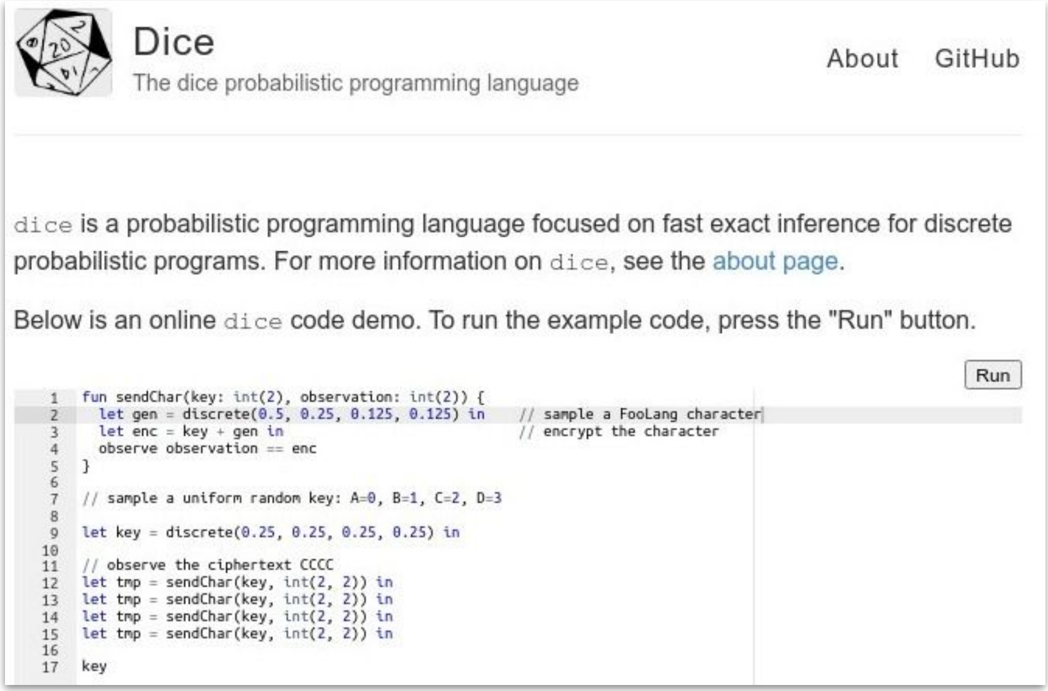
Programming languages are humanity's biggest knowledge representation achievement!

Programs should be AI models

Dice probabilistic programming language

http://dicelang.cs.ucla.edu/

https://github.com/SHoltzen/dice



dice is a probabilistic programming language focused on fast exact inference for discrete probabilistic programs. For more information on dice, see the about page.

Below is an online dice code demo. To run the example code, press the "Run" button.

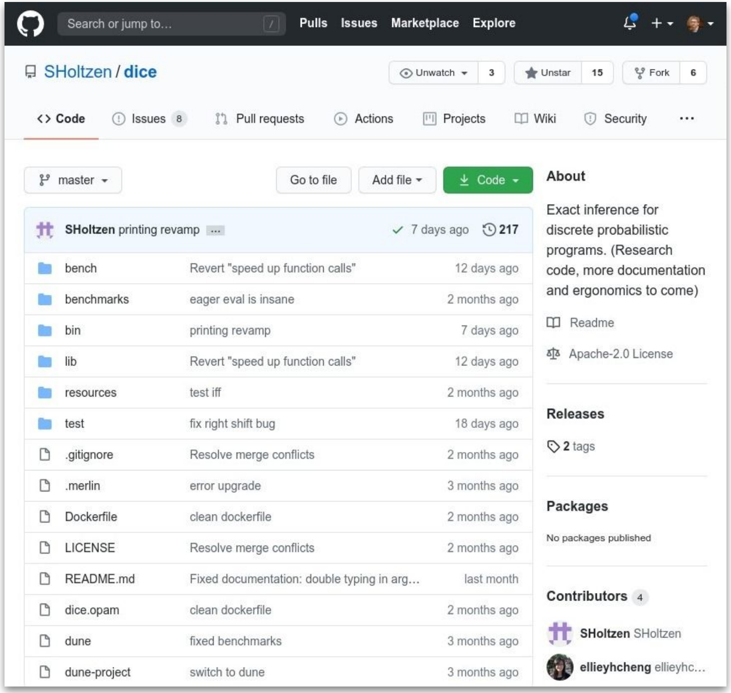
```
fun sendChar(key: int(2), observation: int(2)) {
  let gen = discrete(0.5, 0.25, 0.125, 0.125) in    // sample a FooLang character
  let enc = key + gen in                            // encrypt the character
  observe observation == enc
}

// sample a uniform random key: A=0, B=1, C=2, D=3

let key = discrete(0.25, 0.25, 0.25, 0.25) in

// observe the ciphertext CCCC
let tmp = sendChar(key, int(2, 2)) in
let tmp = sendChar(key, int(2, 2)) in
let tmp = sendChar(key, int(2, 2)) in
let tmp = sendChar(key, int(2, 2)) in

key
```



SHoltzen printing revamp ... ✓ 7 days ago 217

bench	Revert "speed up function calls"	12 days ago
benchmarks	eager eval is insane	2 months ago
bin	printing revamp	7 days ago
lib	Revert "speed up function calls"	12 days ago
resources	test iff	2 months ago
test	fix right shift bug	18 days ago
.gitignore	Resolve merge conflicts	2 months ago
.merlin	error upgrade	3 months ago
Dockerfile	clean dockerfile	2 months ago
LICENSE	Resolve merge conflicts	2 months ago
README.md	Fixed documentation: double typing in arg...	last month
dice.opam	clean dockerfile	2 months ago
dune	fixed benchmarks	3 months ago
dune-project	switch to dune	3 months ago

About

Exact inference for discrete probabilistic programs. (Research code, more documentation and ergonomics to come)

Readme

Apache-2.0 License

Releases

2 tags

Packages

No packages published

Contributors 4

SHoltzen SHoltzen

ellieyhcheng ellieyhc...

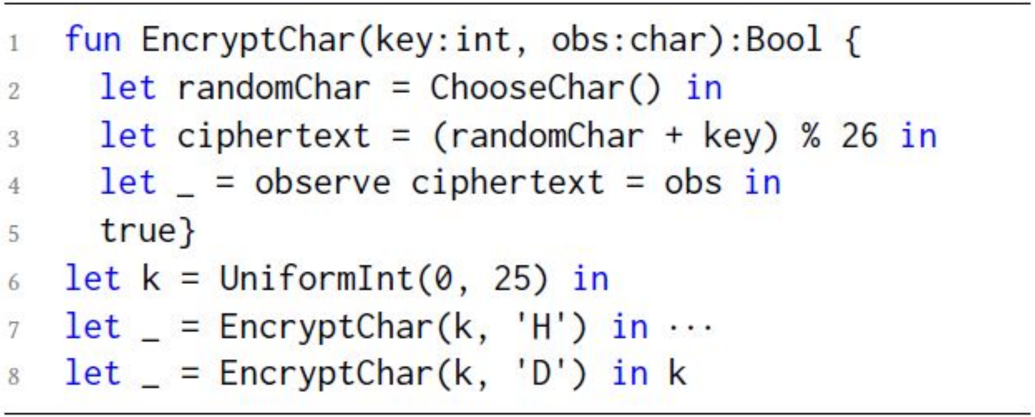
[Holtzen et al. OOPSLA20]

Why should I care?

Better abstraction than probabilistic graphical models:

- Beyond variable-level dependencies (contextual)
- modularity through functions
  reuse (cf. relational graphical models)
- intuitive language for local structure; arithmetic
- data structures
- first-class observations

First-Class Observations

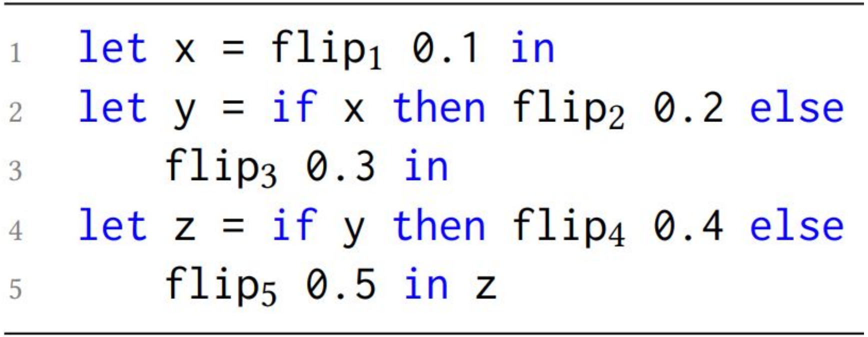


```
fun EncryptChar(key:int, obs:char):Bool {
  let randomChar = ChooseChar() in
  let ciphertext = (randomChar + key) % 26 in
  let _ = observe ciphertext = obs in
  true}
let k = UniformInt(0, 25) in
let _ = EncryptChar(k, 'H') in ···
let _ = EncryptChar(k, 'D') in k
```

Frequency Analyzer for a Caesar cipher in Dice

Probabilistic Program Inference

Key ingredient: **factorization**
…. aka the product nodes



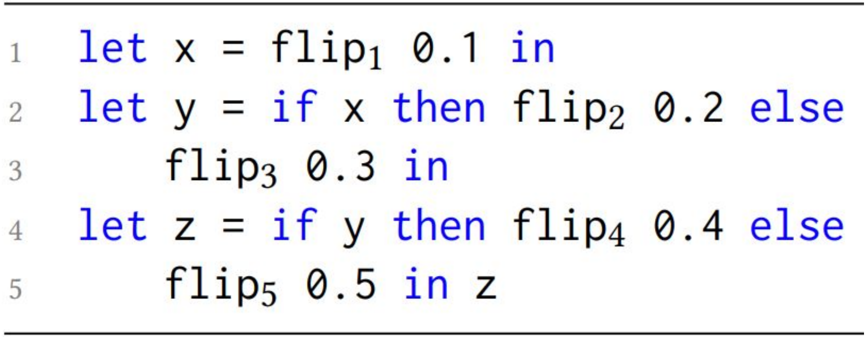
```
let x = flip_1 0.1 in
let y = if x then flip_2 0.2 else
    flip_3 0.3 in
let z = if y then flip_4 0.4 else
    flip_5 0.5 in z
```

$$\underbrace{0.1}_{x=T} \cdot \underbrace{0.2}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.1}_{x=T} \cdot \underbrace{0.8}_{y=F} \cdot \underbrace{0.5}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.3}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.7}_{y=F} \cdot \underbrace{0.5}_{z=T}$$

$$\Rightarrow \underbrace{0.1}_{x=T} \cdot \left(\underbrace{0.2}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.8}_{y=F} \cdot \underbrace{0.5}_{z=T} \right) + \underbrace{0.9}_{x=F} \cdot \left(\underbrace{0.3}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.7}_{y=F} \cdot \underbrace{0.5}_{z=T} \right)$$

Symbolic Compilation in Dice

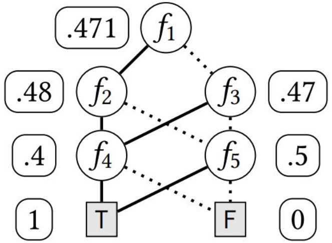
- Construct Boolean formula
- Satisfying assignments ≈ paths
- Variables are flips
- Associate weights with flips
- Compile factorized circuit



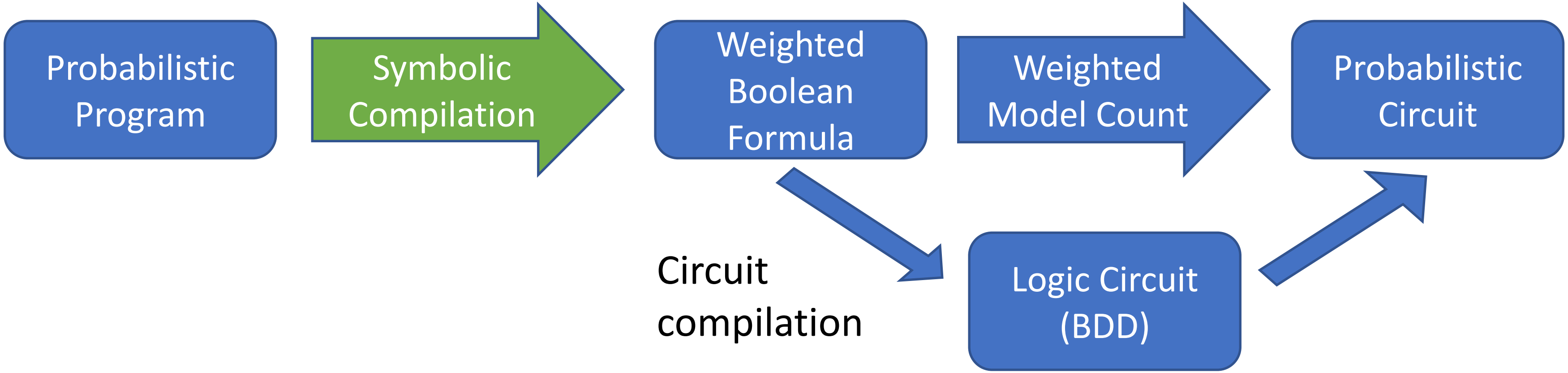
```
let x = flip₁ 0.1 in
let y = if x then flip₂ 0.2 else
    flip₃ 0.3 in
let z = if y then flip₄ 0.4 else
    flip₅ 0.5 in z
```

$$\underbrace{0.1}_{x=T} \cdot \underbrace{0.2}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.1}_{x=T} \cdot \underbrace{0.8}_{y=F} \cdot \underbrace{0.5}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.3}_{y=T} \cdot \underbrace{0.4}_{z=T} + \underbrace{0.9}_{x=F} \cdot \underbrace{0.7}_{y=F} \cdot \underbrace{0.5}_{z=T}$$

➡ $f_1 f_2 f_4 \vee f_1 \bar{f}_2 f_5 \vee \bar{f}_1 f_3 f_4 \vee \bar{f}_1 \bar{f}_3 f_5$ ➡

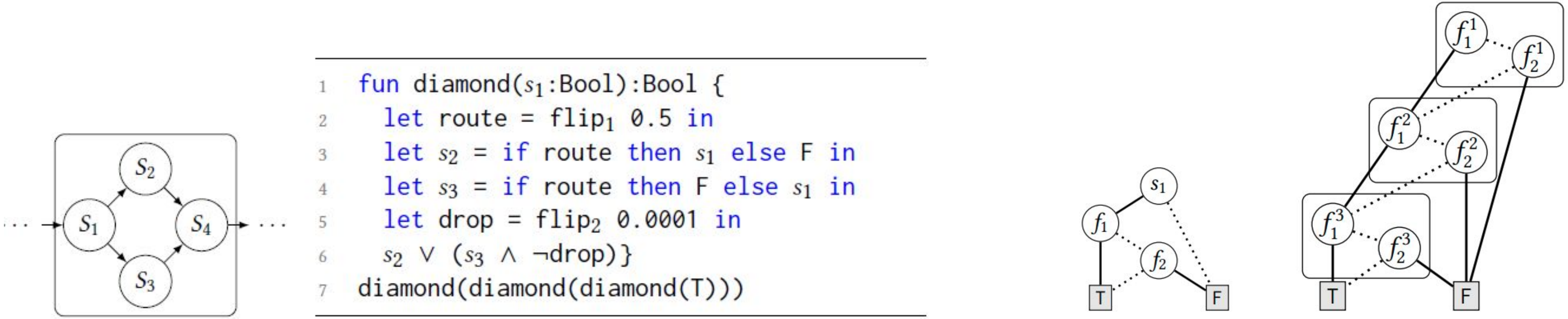


Symbolic Compilation to Probabilistic Circuits



State of the art for discrete probabilistic program inference!

Factorized Inference in Dice



```
fun diamond(s₁:Bool):Bool {
  let route = flip₁ 0.5 in
  let s₂ = if route then s₁ else F in
  let s₃ = if route then F else s₁ in
  let drop = flip₂ 0.0001 in
  s₂ ∨ (s₃ ∧ ¬drop)}
diamond(diamond(diamond(T)))
```

(a) Network diagram.

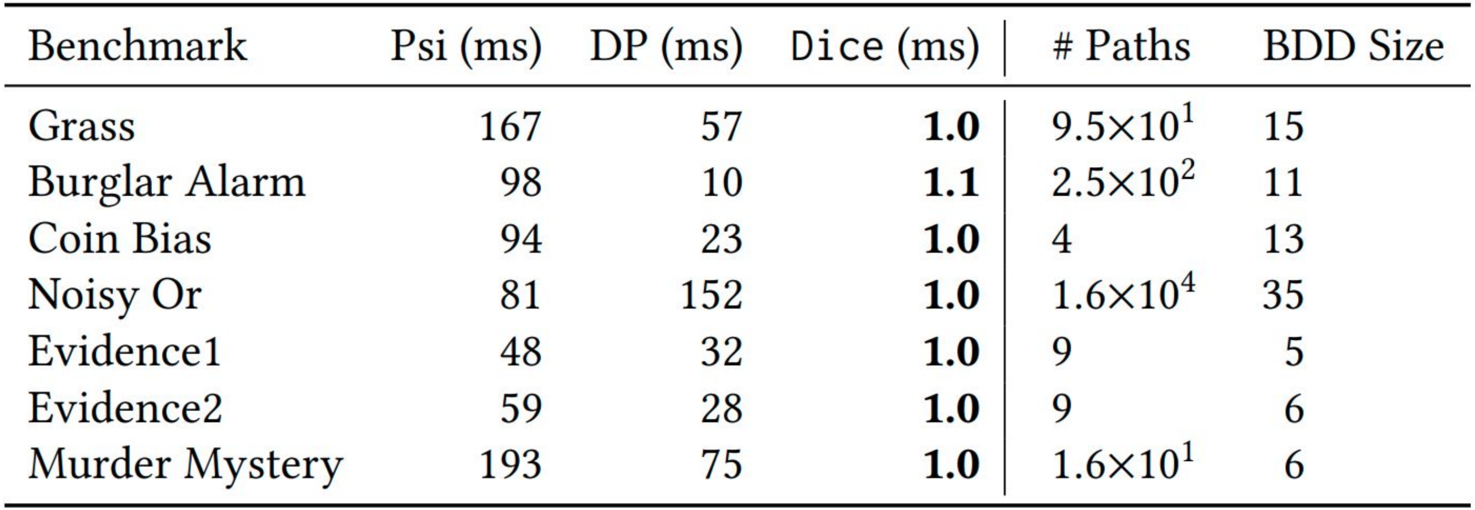
(b) Probabilistic program defining the network.

(c) diamond function.

(d) Final BDD.

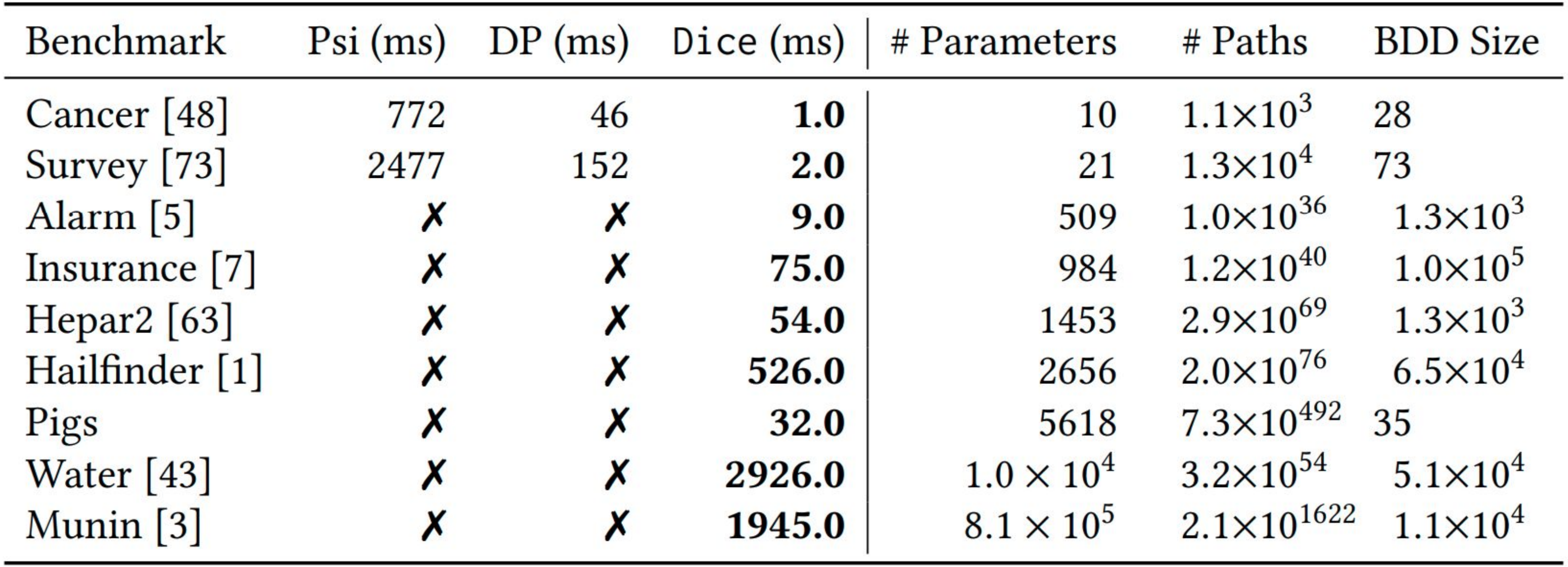
Network Verification

PPL benchmarks from PL community



Benchmark	Psi (ms)	DP (ms)	Dice (ms)	# Paths	BDD Size
Grass	167	57	**1.0**	9.5×10^1	15
Burglar Alarm	98	10	**1.1**	2.5×10^2	11
Coin Bias	94	23	**1.0**	4	13
Noisy Or	81	152	**1.0**	1.6×10^4	35
Evidence1	48	32	**1.0**	9	5
Evidence2	59	28	**1.0**	9	6
Murder Mystery	193	75	**1.0**	1.6×10^1	6

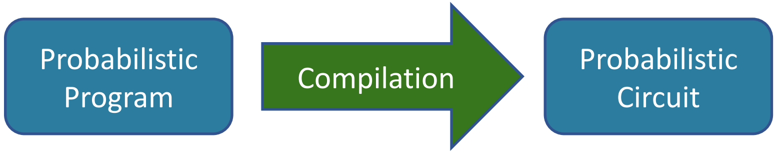
Scalable Inference



Benchmark	Psi (ms)	DP (ms)	Dice (ms)	# Parameters	# Paths	BDD Size
Cancer [48]	772	46	**1.0**	10	$1.1{\times}10^{3}$	28
Survey [73]	2477	152	**2.0**	21	$1.3{\times}10^{4}$	73
Alarm [5]	✗	✗	**9.0**	509	$1.0{\times}10^{36}$	$1.3{\times}10^{3}$
Insurance [7]	✗	✗	**75.0**	984	$1.2{\times}10^{40}$	$1.0{\times}10^{5}$
Hepar2 [63]	✗	✗	**54.0**	1453	$2.9{\times}10^{69}$	$1.3{\times}10^{3}$
Hailfinder [1]	✗	✗	**526.0**	2656	$2.0{\times}10^{76}$	$6.5{\times}10^{4}$
Pigs	✗	✗	**32.0**	5618	$7.3{\times}10^{492}$	35
Water [43]	✗	✗	**2926.0**	1.0×10^{4}	$3.2{\times}10^{54}$	$5.1{\times}10^{4}$
Munin [3]	✗	✗	**1945.0**	8.1×10^{5}	$2.1{\times}10^{1622}$	$1.1{\times}10^{4}$

Conclusions

- Are we already in the age of computational abstractions?
- **Probabilistic circuits** for learning deep <u>tractable</u> probabilistic models
- **Probabilistic programs** as the new probabilistic knowledge representation language
- Two computational abstractions go hand in hand



Probabilistic Program

Compilation

Probabilistic Circuit

Thanks

My students/postdoc who did the real work are graduating.

There are some awesome people on the academic job market!