UCLA Computer Science

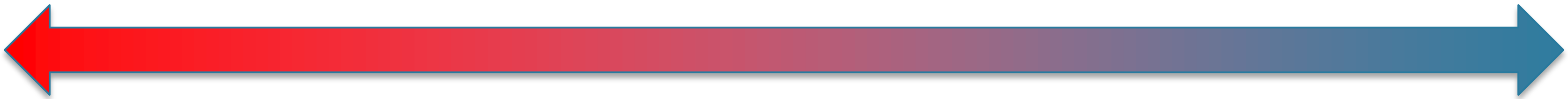
STAR AI RESEARCH LAB UCLA

Querying Advanced Probabilistic Models: From Relational Embeddings to Probabilistic Programs

Guy Van den Broeck

StarAI Workshop @ AAAI, Feb 7, 2020

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

The **FALSE** AI Dilemma

So all hope is lost?

Probabilistic World Models

- Joint distribution P(X)
- Wealth of representations: can be causal, relational, etc.
- Knowledge + data
- Reasoning + learning

Pure Logic **Probabilistic World Models** **Pure Learning**

A New Synthesis of Learning and Reasoning

Tutorial on Probabilistic Circuits
This afternoon: 2pm-6pm
Sutton Center, 2nd floor

Pure Logic Probabilistic World Models Pure Learning

High-Level Probabilistic Representations

1

Probabilistic Databases Meets
Relational Embeddings:
Symbolic Querying of Vector Spaces

2

Modular Exact Inference for
Discrete Probabilistic Programs

What we'd like to do…

Has anyone published a paper with both Erdos and Einstein

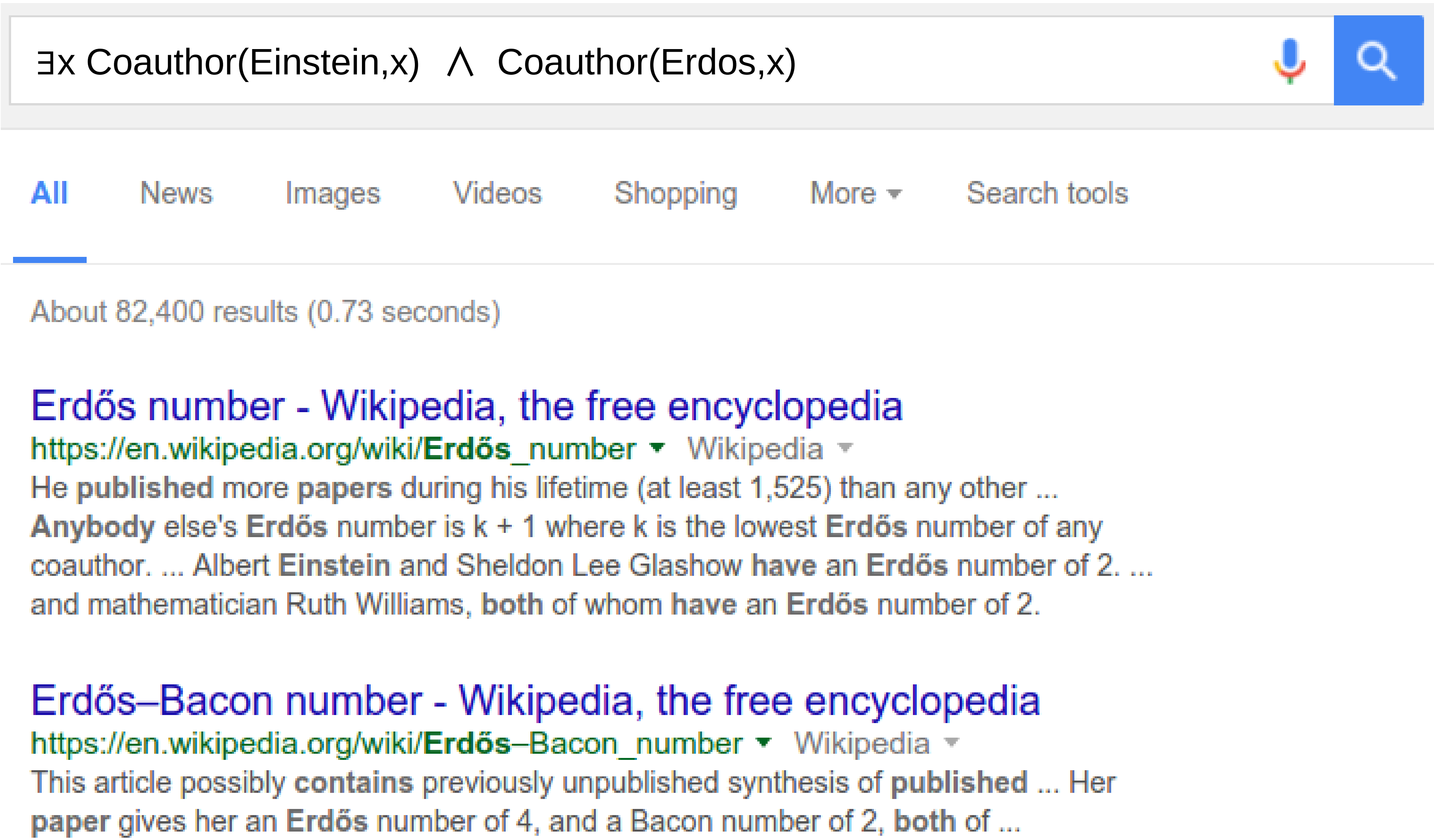
All News Images Videos Shopping More Search tools

About 82,400 results (0.73 seconds)

Erdős number - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Erdős**_number Wikipedia
He **published** more **papers** during his lifetime (at least 1,525) than any other ...
Anybody else's **Erdős** number is k + 1 where k is the lowest **Erdős** number of any
coauthor. ... Albert **Einstein** and Sheldon Lee Glashow **have** an **Erdős** number of 2. ...
and mathematician Ruth Williams, **both** of whom **have** an **Erdős** number of 2.

Erdős–Bacon number - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Erdős**–Bacon_number Wikipedia
This article possibly **contains** previously unpublished synthesis of **published** ... Her
paper gives her an **Erdős** number of 4, and a Bacon number of 2, **both** of ...

What we'd like to do…



∃x Coauthor(Einstein,x) ∧ Coauthor(Erdos,x)

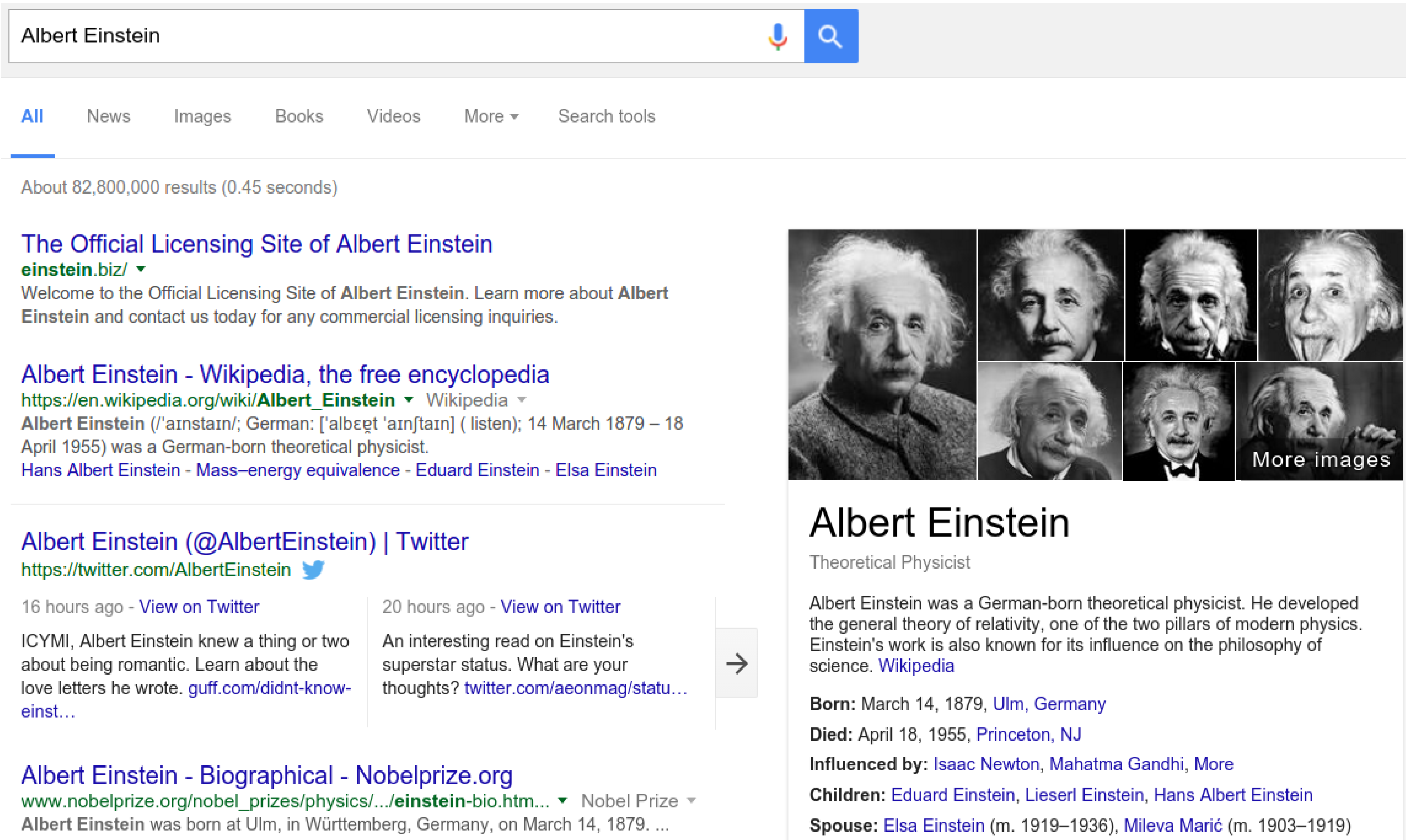
All News Images Videos Shopping More Search tools

About 82,400 results (0.73 seconds)

Erdős number - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Erdős**_number Wikipedia
He **published** more **papers** during his lifetime (at least 1,525) than any other ... **Anybody** else's **Erdős** number is k + 1 where k is the lowest **Erdős** number of any coauthor. ... Albert **Einstein** and Sheldon Lee Glashow **have** an **Erdős** number of 2. ... and mathematician Ruth Williams, **both** of whom **have** an **Erdős** number of 2.

Erdős–Bacon number - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Erdős**–Bacon_number Wikipedia
This article possibly **contains** previously unpublished synthesis of **published** ... Her **paper** gives her an **Erdős** number of 4, and a Bacon number of 2, **both** of ...

Einstein is in the Knowledge Graph



Erdős is in the Knowledge Graph

Paul Erdos

All Images Videos Books News More Search tools

About 333,000 results (0.35 seconds)

Paul Erdős - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**Paul_Erdős** Wikipedia
Paul Erdős was a Hungarian Jewish mathematician. He was one of the most prolific mathematicians of the 20th century. He was known both for his social ...
Fan Chung - Ronald Graham - Béla Bollobás - Category:Paul Erdős

The Man Who Loved Only Numbers - The New York Times
https://www.nytimes.com/books/.../hoffman-man.ht... The New York Times
Paul Erdös was one of those very special geniuses, the kind who comes along only once in a very long while yet he chose, quite consciously I am sure, to share ...

Paul Erdos | Hungarian mathematician | Britannica.com
www.britannica.com/biography/**Paul-Erdos** Encyclopaedia Britannica
Paul Erdős, (born March 26, 1913, Budapest, Hungary—died September 20, 1996, Warsaw, Poland), Hungarian “freelance” mathematician (known for his work ...

Paul Erdős - University of St Andrews
www-groups.dcs.st-and.ac.uk/~history/Biographies/**Erdos**.html
Paul Erdős came from a Jewish family (the original family name being Engländer) although neither of his parents observed the Jewish religion. Paul's father ...

[PDF] Paul Erdős Mathematical Genius, Human - UnTruth.org
www.untruth.org/~josh/math/**Paul**%20**Erdös**%20bio-rev2.pdf
by J Hill - 2004 - Related articles

Paul Erdős

Mathematician

Paul Erdős was a Hungarian Jewish mathematician. He was one of the most prolific mathematicians of the 20th century. He was known both for his social practice of mathematics and for his eccentric lifestyle.
Wikipedia

Born: March 26, 1913, Budapest, Hungary

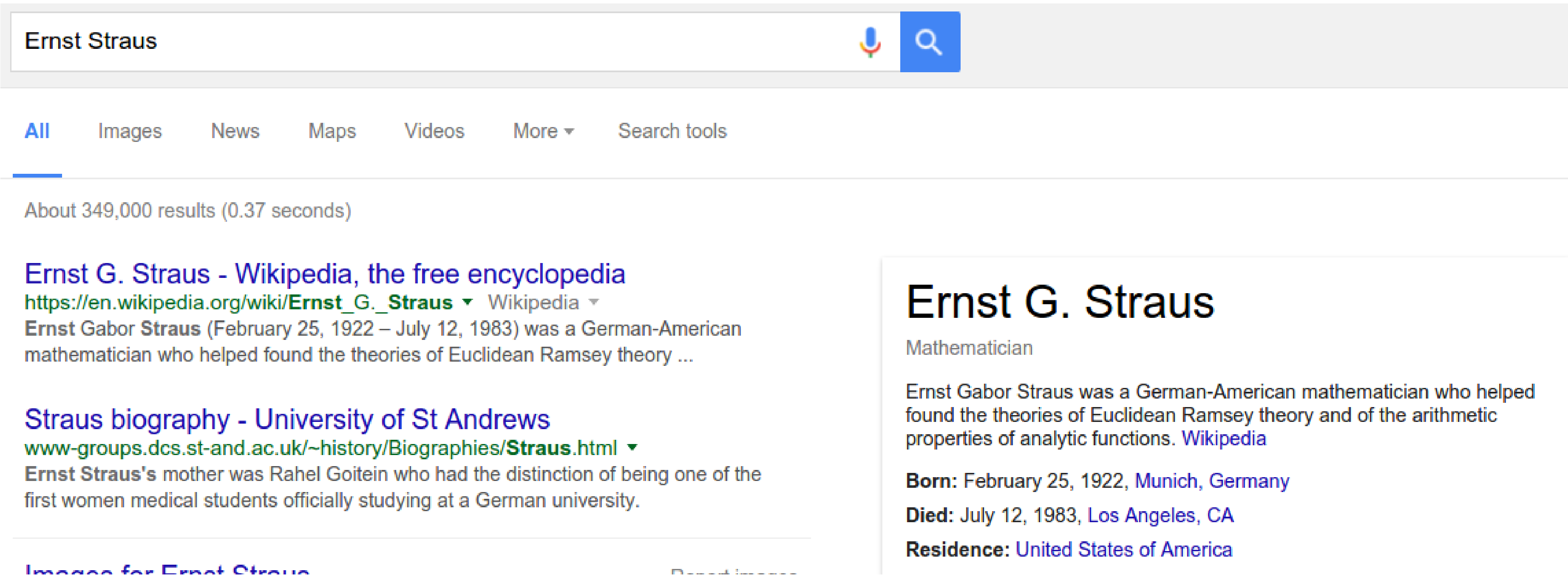
Died: September 20, 1996, Warsaw, Poland

Education: Eötvös Loránd University (1934)

Books: Probabilistic Methods in Combinatorics, More

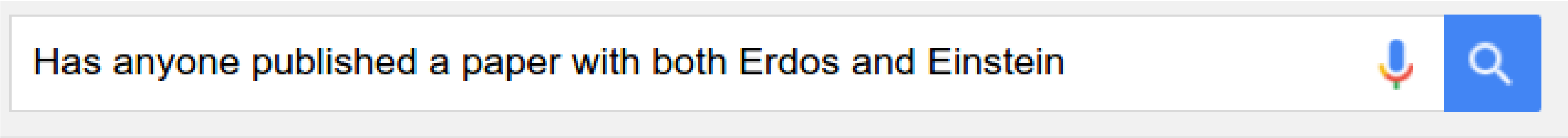
Notable students: Béla Bollobás, Alexander Soifer, George B. Purdy, Joseph Kruskal

This guy is in the Knowledge Graph



… and he published with both Einstein and Erdos!

Desired Query Answer



Ernst Straus

Barack Obama, …

Justin Bieber, …

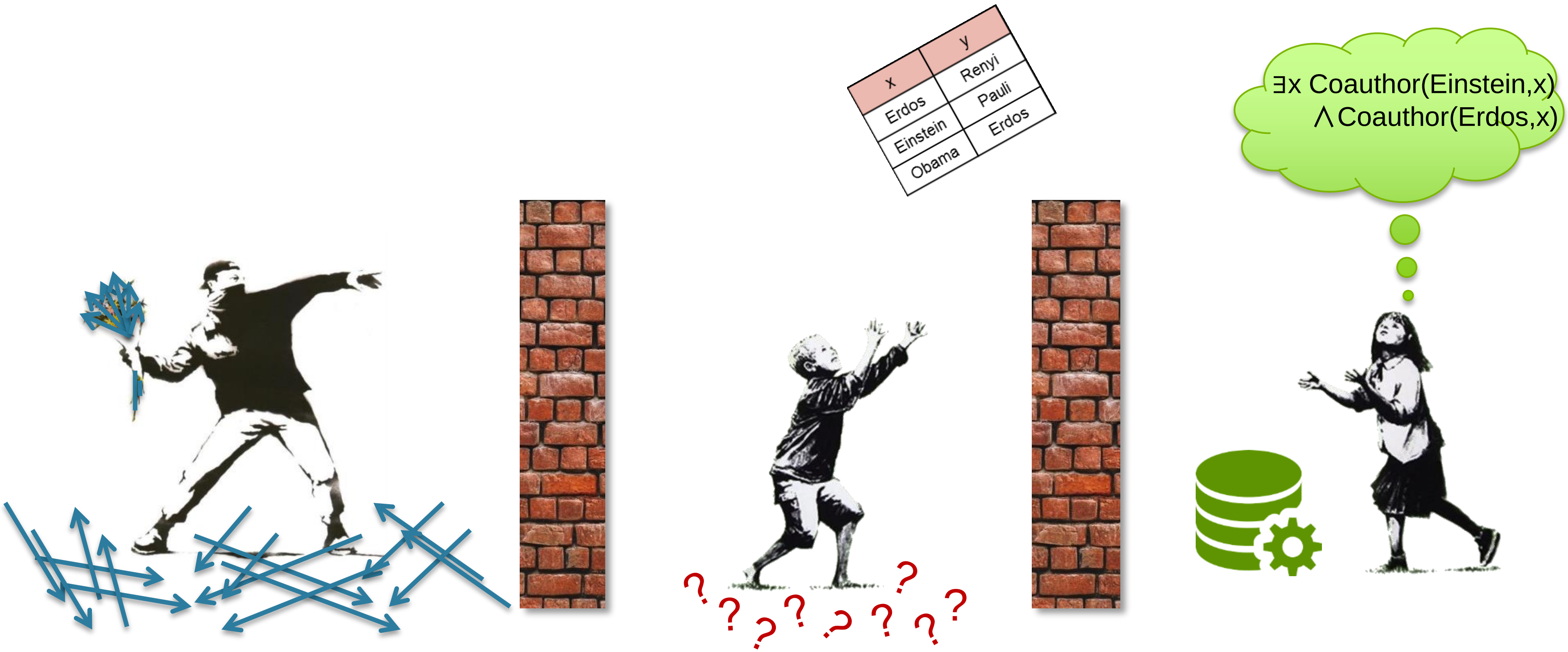
1. Fuse uncertain information from web

⇒ **Embrace probability!**

2. Cannot come from labeled data

⇒ **Embrace query eval!**

Cartoon Motivation



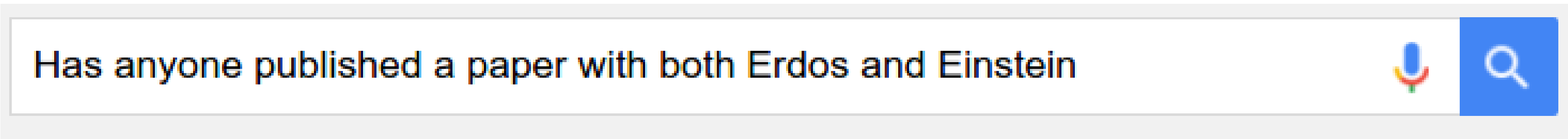
Relational
Embedding
Vectors

Curate
Knowledge
Graph

Query
in a
DBMS

Many exceptions in StarAI and PDB communities, but, we need to embed…

Probabilistic Databases



- Probabilistic database

Scientist

x	P
Erdos	0.9
Einstein	0.8
Pauli	0.6

Coauthor

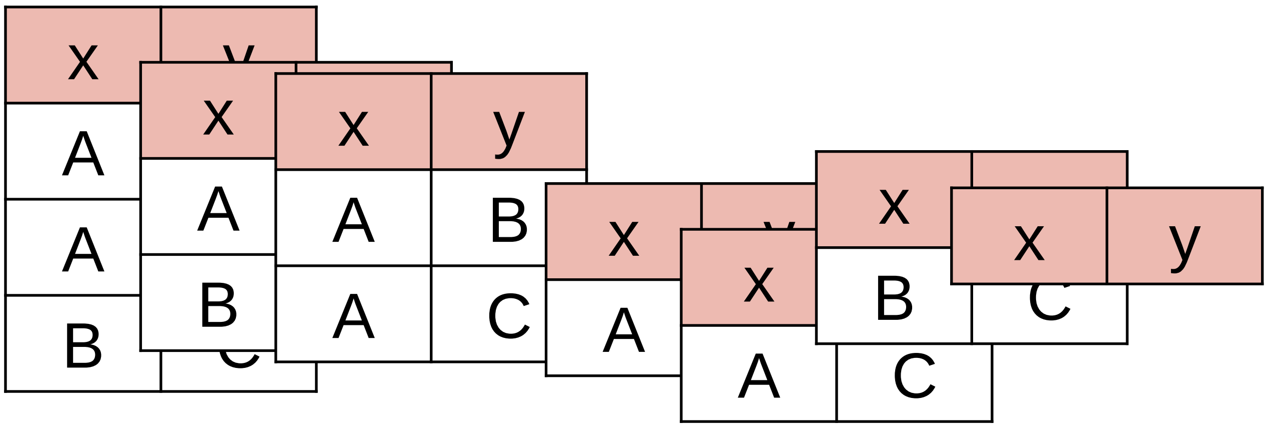
x	y	P
Erdos	Renyi	0.6
Einstein	Pauli	0.7
Obama	Erdos	0.1

- Learned from the web, large text corpora, ontologies, etc., using **statistical** machine learning.

[VdB&Suciu'17]

Probabilistic Databases Semantics

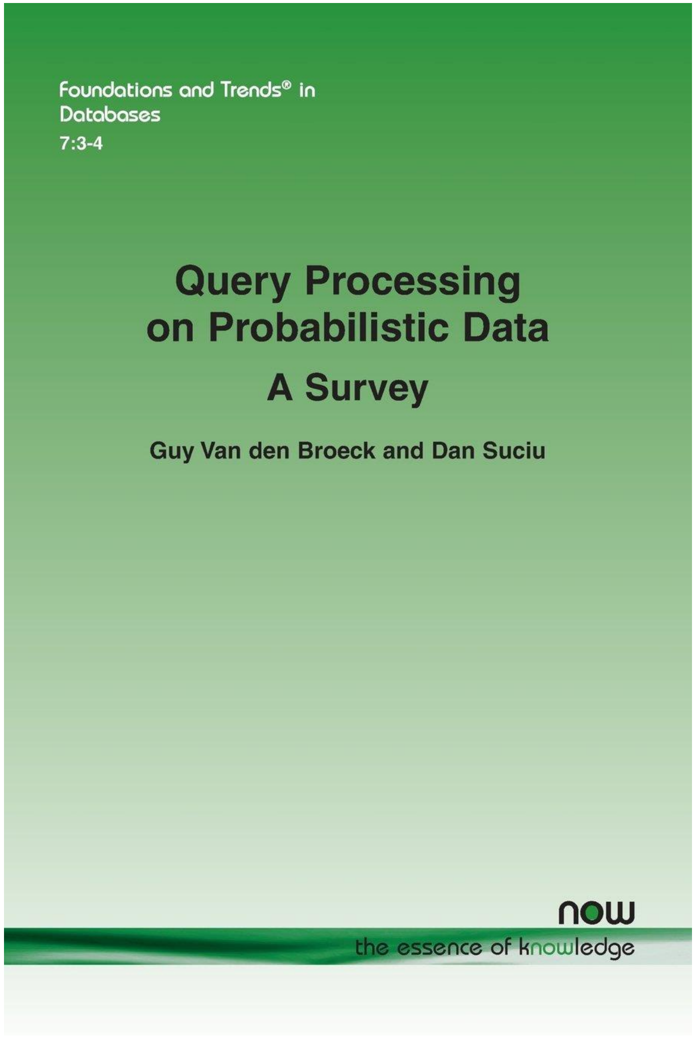
- All possible databases: $\Omega = \{\omega_1, \dots, \omega_n\}$



- Probabilistic database P assigns a probability to each: $P\colon \Omega \to [0,1]$
- Probabilities sum to 1: $\sum_{\omega \in \Omega} P(\omega) = 1$

[VdB&Suciu'17]

Commercial Break



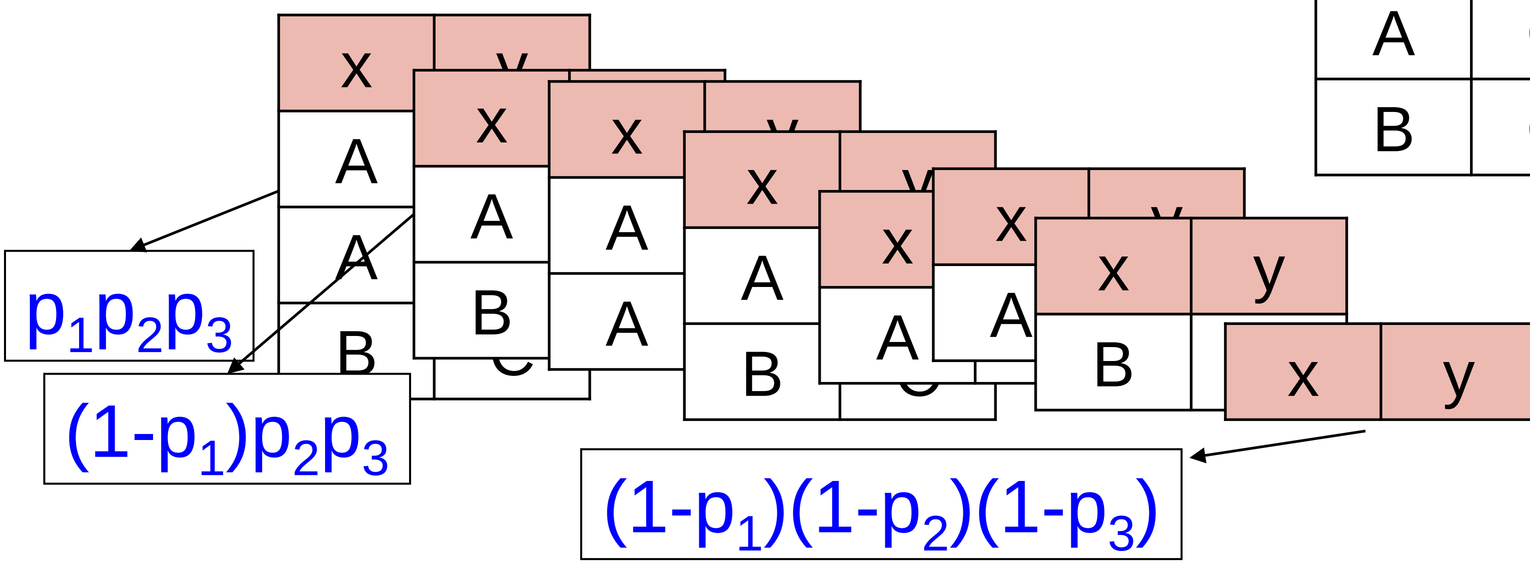
- Survey book
  http://www.nowpublishers.com/article/Details/DBS-052
- IJCAI 2016 tutorial
  http://web.cs.ucla.edu/~guyvdb/talks/IJCAI16-tutorial/

How to specify all these numbers?

- Only specify marginals:
  $P(Coauthor(Alice, Bob)) = 0.23$
- Assume tuple-independence

Coauthor

x	y	P
A	B	p_1
A	C	p_2
B	C	p_3



[VdB&Suciu'17]

Probabilistic Query Evaluation

$Q = \exists x \exists y \; \text{Scientist}(x) \wedge \text{Coauthor}(x,y)$

$$P(Q) = 1 - \{1 - p_1 * [\; 1-(1-q_1)*(1-q_2) \;]\} * \{1 - p_2 * [\; 1-(1-q_3)*(1-q_4)*(1-q_5) \;]\}$$

Scientist

x	P
A	p_1
B	p_2
C	p_3

Coauthor

x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Lifted Inference Rules

Preprocess Q (omitted),
Then apply rules (some have preconditions)

$P(\neg Q) = 1 - P(Q)$ — Negation

$P(Q1 \wedge Q2) = P(Q1)\, P(Q2)$
$P(Q1 \vee Q2) = 1 - (1 - P(Q1))\,(1 - P(Q2))$ — Decomposable $\wedge,\vee$

$P(\forall z\, Q) = \Pi_{A \in Domain}\ P(Q[A/z])$
$P(\exists z\, Q) = 1 - \Pi_{A \in Domain}\ (1 - P(Q[A/z]))$ — Decomposable $\exists,\forall$

$P(Q1 \wedge Q2) = P(Q1) + P(Q2) - P(Q1 \vee Q2)$
$P(Q1 \vee Q2) = P(Q1) + P(Q2) - P(Q1 \wedge Q2)$ — Inclusion/exclusion

Example Query Evaluation

$$Q = \exists x\, \exists y\, \text{Scientist}(x) \wedge \text{Coauthor}(x,y)$$

Decomposable ∃-Rule

$$P(Q) = 1 - \Pi_{A \in \text{Domain}} (1 - P(\text{Scientist}(A) \wedge \exists y\, \text{Coauthor}(A,y))$$

Check independence:

$\text{Scientist}(A) \wedge \exists y\, \text{Coauthor}(A,y)$

$\text{Scientist}(B) \wedge \exists y\, \text{Coauthor}(B,y)$

$$
\begin{aligned}
= 1 - &(1 - P(\text{Scientist}(A) \wedge \exists y\, \text{Coauthor}(A,y)) \\
&\times (1 - P(\text{Scientist}(B) \wedge \exists y\, \text{Coauthor}(B,y)) \\
&\times (1 - P(\text{Scientist}(C) \wedge \exists y\, \text{Coauthor}(C,y)) \\
&\times (1 - P(\text{Scientist}(D) \wedge \exists y\, \text{Coauthor}(D,y)) \\
&\times (1 - P(\text{Scientist}(E) \wedge \exists y\, \text{Coauthor}(E,y)) \\
&\times (1 - P(\text{Scientist}(F) \wedge \exists y\, \text{Coauthor}(F,y)) \\
&\quad \ldots
\end{aligned}
$$

Complexity PTIME

Limitations

$H_0 = \forall x \forall y \, \text{Smoker}(x) \lor \text{Friend}(x,y) \lor \text{Jogger}(y)$

The decomposable ∀-rule: $P(\forall z \, Q) = \prod_{A \in \text{Domain}} P(Q[A/z])$

… does not apply:

$H_0[\text{Alice}/x]$ and $H_0[\text{Bob}/x]$ are dependent:

Dependent

$\forall y \, (\text{Smoker}(\text{Alice}) \lor \text{Friend}(\text{Alice},y) \lor \text{Jogger}(y))$

$\forall y \, (\text{Smoker}(\text{Bob}) \lor \text{Friend}(\text{Bob},y) \lor \text{Jogger}(y))$

Lifted inference sometimes fails.

Are the Lifted Rules Complete?

Dichotomy Theorem for Unions of Conjunction Queries / Monotone CNF

- If lifted rules succeed, then PTIME query
- If lifted rules fail, then query is #P-hard

Lifted rules are complete for UCQ!

[Dalvi and Suciu;JACM'11]

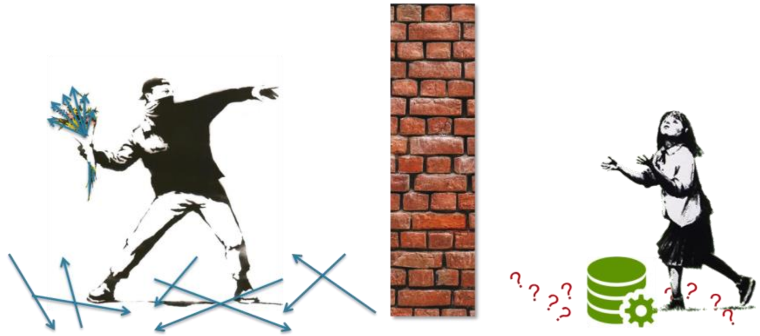
The Good, Bad, Ugly

- We understand querying <u>very well</u>! ☺
  - and it is often efficient (a rare property!)
  - but often also highly intractable ☹
- Tuple-independence is limiting unless reducing from a more expressive model ☹

  Can reduce from MLNs but then intractable…
- Where do probabilities come from? ☹ ☹

  An unspecified “statistical model”

Throwing Relational Embedding Models Over the Wall



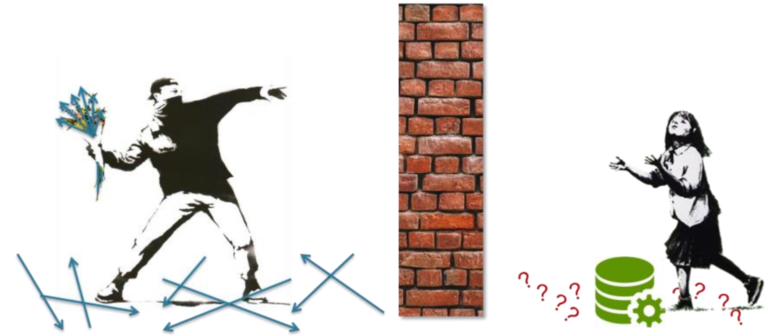
- Associate vector with
  - each relation R
  - each entity A, B, …
- Score S(head, relation, tail) (based on Euclidian, cosine, …)

Coauthor

x	y	S
A	B	.6
A	C	-.1
B	C	.4

Method	Entity Embedding	Relation Embedding	Triple Score
TransE (Bordes et al., 2013)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^d$	$\|\|v_h + v_R - v_t\|\|$
DistMult (Yang et al., 2014)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^d$	$\langle v_h, v_R, v_t \rangle$
Rescal (Nickel et al., 2011)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^{d \times d}$	$v_h^T v_R v_t$
ComplEx (Trouillon et al., 2016)	$v_h, v_t \in \mathbb{C}^d$	$v_R \in \mathbb{C}^d$	$\mathrm{Re}(\langle v_h, v_R, \bar{v_t} \rangle)$

Throwing Relational Embedding Models Over the Wall



Interpret scores as probabilities

High score ~ prob 1 ; Low score ~ prob 0

Coauthor

x	y	S
A	B	.6
A	C	-.1
B	C	.4

Coauthor

x	y	P
A	B	0.9
A	C	0.1
B	C	0.5

The Good, Bad, Ugly

- Where do probabilities come from?

  We *finally* know the "statistical model"! ☺

  Both capture marginals: a good match
- We still understand querying <u>very well</u>! ☺

  but it is often highly intractable ☹
- Tuple-independence is limiting ☹ ☹

  Relational embedding models do not attempt to capture dependencies in link prediction

A Second Attempt

- Let's simplify drastically!
- Assume each relation has the form

$$R(x, y) \Leftrightarrow T_R \wedge E(x) \wedge E(y)$$

- That is, there are latent relations
  - T_* to decide which relations can be true
  - E to decide which entities participate

Coauthor

x	y	P
A	B	0.9
A	C	0.1
B	C	0.5

$\sim$

T

P
0.2

,

E

x	P
A	0.2
B	0.5
C	0.3

Can this do link prediction?

- Predict Coauthor(Alice,Bob)

Coauthor

x	y	P
A	B	?

$\sim$ **T**

P
0.3

, **E**

x	P
A	0.2
B	0.5
C	0.3

- Rewrite query using

$$R(x,y) \Leftrightarrow T_R \wedge E(x) \wedge E(y)$$

- Apply standard lifted inference rules
- P(Coauthor(Alice,Bob)) $= 0.3 \cdot 0.2 \cdot 0.5$

The Good, ~~Bad,~~ Ugly

- Where do probabilities come from?

  We *finally* know the “statistical model”! ☺

- We still understand querying <u>very well</u>! ☺

  By rewriting R into E and T_R, every UCQ query becomes tractable! ☺ ☺ ☺ ☺ ☺

- Tuples sharing entities or relation symbols depend one each other

- The model is not very expressive ☹

A Third Attempt

- Mixture models of the second attempt

$$R(x, y) \Leftrightarrow T_R \wedge E(x) \wedge E(y)$$

  Now, there are latent relations T_R and E for each mixture component

- The Good: ☺
  - Still a clear statistical model
  - Every UCQ query is still tractable
  - Still captures tuple dependencies
  - Mixture can approximate any distribution

Can this do link prediction?

- Predict Coauthor(Alice,Bob) in each mixture component
  - $P_1(\text{Coauthor(Alice,Bob)}) = 0.3 \cdot 0.2 \cdot 0.5$
  - $P_2(\text{Coauthor(Alice,Bob)}) = 0.9 \cdot 0.1 \cdot 0.6$
  - Etc.
- Probability in mixture of *d* components

$$P(\text{Coauthor(Alice,Bob)}) = \frac{1}{d} 0.3 \cdot 0.2 \cdot 0.5 + \frac{1}{d} 0.9 \cdot 0.1 \cdot 0.6 + \cdots$$

How good is this?

Does it look familiar?

$$P(\text{Coauthor(Alice,Bob)})$$
$$= \frac{1}{d} 0.3 \cdot 0.2 \cdot 0.5 + \frac{1}{d} 0.9 \cdot 0.1 \cdot 0.6 + \cdots$$

Method	Entity Embedding	Relation Embedding	Triple Score
TransE (Bordes et al., 2013)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^d$	$\|\|v_h + v_R - v_t\|\|$
DistMult (Yang et al., 2014)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^d$	$\langle v_h, v_R, v_t \rangle$
Rescal (Nickel et al., 2011)	$v_h, v_t \in \mathbb{R}^d$	$v_R \in \mathbb{R}^{d \times d}$	$v_h^T v_R v_t$
ComplEx (Trouillon et al., 2016)	$v_h, v_t \in \mathbb{C}^d$	$v_R \in \mathbb{C}^d$	$\mathrm{Re}(\langle v_h, v_R, \bar{v}_t \rangle)$

How good is this?

- At link prediction: same as DistMult
- At queries on bio dataset [Hamilton]

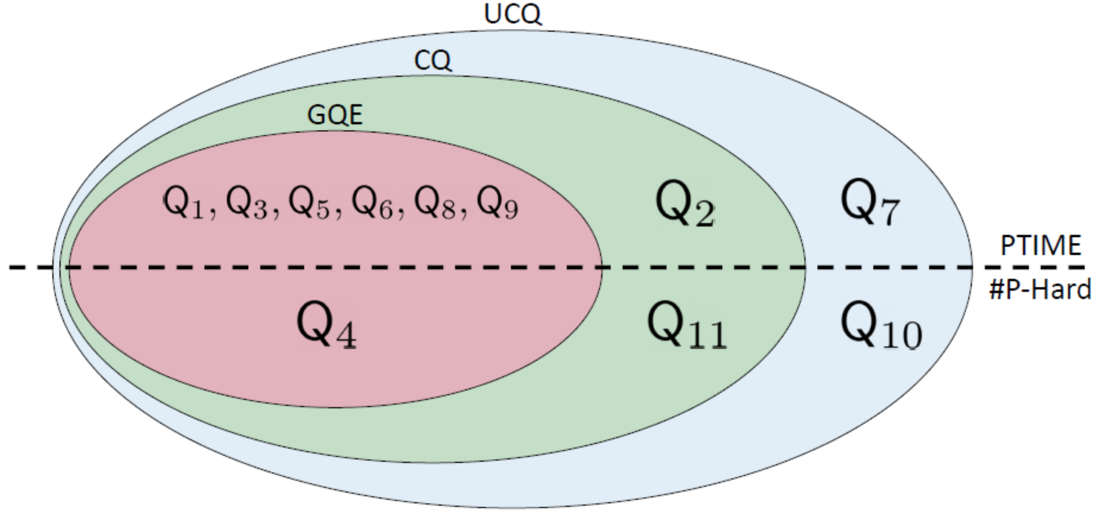
  Competitive,
  while having a consistent underlying distribution

  Ask Tal at his poster!

Method	AUC	APR
Bilinear	79.2	78.6
DistMult	86.7	87.5
TransE	78.3	81.6
TractOR-pos	75.0	84.5
TractOR	82.8	86.3

How expressive is this?

$\mathsf{Q}_1(t) =$	$R(a,t)$
$\mathsf{Q}_2(t) =$	$\exists x.R(a,x)$
$\mathsf{Q}_3(t) =$	$\exists x.R(a,x) \wedge S(x,t)$
$\mathsf{Q}_4(t) =$	$\exists x,y.R(a,x) \wedge S(x,y) \wedge T(y,t)$
$\mathsf{Q}_5(t) =$	$R(a,t) \wedge S(b,t)$
$\mathsf{Q}_6(t) =$	$R(a,t) \wedge S(b,t) \wedge T(c,t)$
$\mathsf{Q}_7(t) =$	$\exists x.R(a,x) \wedge S(x,t)$
	$\vee \exists y.R(a,y) \wedge T(y,t)$
$\mathsf{Q}_8(t) =$	$\exists x.R(a,x) \wedge S(x,t) \wedge T(b,t)$
$\mathsf{Q}_9(t) =$	$\exists x.R(a,x) \wedge S(b,x) \wedge T(x,t)$
$\mathsf{Q}_{10}(t) =$	$\exists x_1,y_1.R(a,x_1) \wedge S(x_1,y_1)$
	$\vee \exists x_2,y_2.S(x_2,y_2) \wedge T(y_2,t)$
$\mathsf{Q}_{11}(t) =$	$\exists x,y,z.R(a,x) \wedge S(x,y) \wedge T(y,z)$



GQE baseline are graph queries
translated to linear algebra
by Hamilton et al [2018]

First Conclusions

- We can give probabilistic database semantics to relational embedding models
  - Gives more meaningful query results
- By doing some solve some annoyances of the theoretical PDB framework
  - Tuple dependence
  - Clear connection to learning
  - While everything stays tractable
  - And the intractable becomes tractable
- Enables much more (train on Q, consistency)

What are probabilistic programs?

```
x ~ flip(0.5);
y ~ flip(0.7);
z := x || y;
if(z) {
    …
}
observe(z);
```

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

Standard programming language constructs

means “reject this execution if z is not true”

Why Probabilistic Programming?

- PPLs are proliferating

Pyro

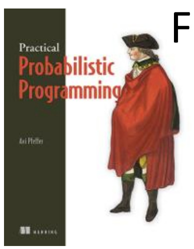
Edward

HackPPL

Venture, Church

Stan

Figaro



ProbLog, PRISM, LPADs, CPLogic, ICL, PHA, etc.

- They have many compelling benefits
  - Specify a probability model in a familiar language
  - Expressive and concise
  - Cleanly separates model from inference

The Challenge of PPL Inference

Most popular inference algorithms are **black box**

– Treat program as a map from inputs to outputs

Pyro Stan Edward

(black-box variational, Hamiltonian MC)

– Simplifying assumptions: differentiability, continuity

– Little to no effort to exploit program structure (automatic differentiation aside)

– Approximate inference ☹

Why Discrete Models?

1. Real programs have inherently discrete structure (e.g. if-statements)

2. Discrete structure is inherent in many domains (graphs, text/topic models, ranking, etc.)

3. Many existing PPLs assume smooth and differentiable densities and do not handle these programs correctly.

   Discrete probabilistic programming is the important unsolved open problem!

Prob. Logic Programming vs. PPL

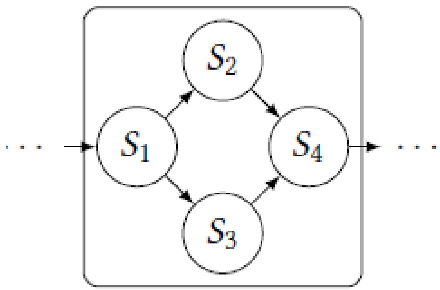
- What is easy for PLP is hard for PPL at large (discrete inference, semantics)
- What is easy for PPL at large is hard for PLP (continues densities, scaling up)

- This community has a lot to contribute.
- What I will present is heavily inspired by the StarAI community's work

Frequency Analyzer for a Caesar cipher in Dice

```
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
```

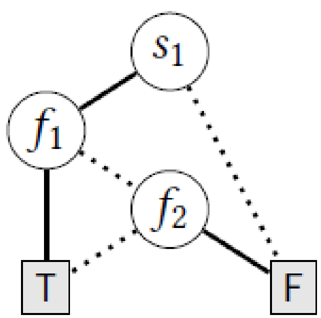
Example Dice Program in Network Verification



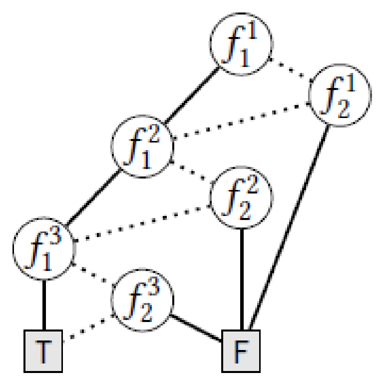
(a) Network diagram.

```
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)))
```

(b) Probabilistic program defining the network.



(c) Summary of `diamond`.



(d) BDD for the program.

Semantics

- The *program state* is a map from *variables* to *values*, denoted σ
- The goal of our semantics is to associate
  - statements in the syntax with
  - a probability distribution on states
- Notation: semantic brackets [[s]]

Sampling Semantics

- The simplest way to give a semantics to our language is to *run the program infinite times*

$x \sim \text{flip}(0.5);$

Draw samples

σ
x=true
x=false
x=true
x=false

- The probability distribution of the program is defined as the *long run average* of how often it ends in a particular state

Semantics of

$x \sim$ flip(0.5);
$y \sim$ flip(0.7);

x = true
y = true

ω_1

`0.5*0.7 = 0.35`

x = false
y = true

ω_2

`0.5*0.7 = 0.35`

x = false
y = false

ω_3

`0.5*0.3 = 0.15`

x = true
y = false

ω_4

`0.5*0.3 = 0.15`

Semantics of

```
x ~ flip(0.5);
y ~ flip(0.7);
observe(x || y);
```

x = true
y = true

ω_1
0.5*0.7 = 0.

x = false
e

.35

= fals
y false

ω_3
0.3 =

e
e

ω_4
0.5*0.3 = 0.15

Semantics: Throw away all executions that do not satisfy the condition x || y.

REJECTION SAMPLING SEMANTICS

Rejection Sampling Semantics

☺

- Extremely general: you only need to be able to run the program to implement a rejection-sampling semantics
- This how most AI researchers think about the meaning of their programs (?)

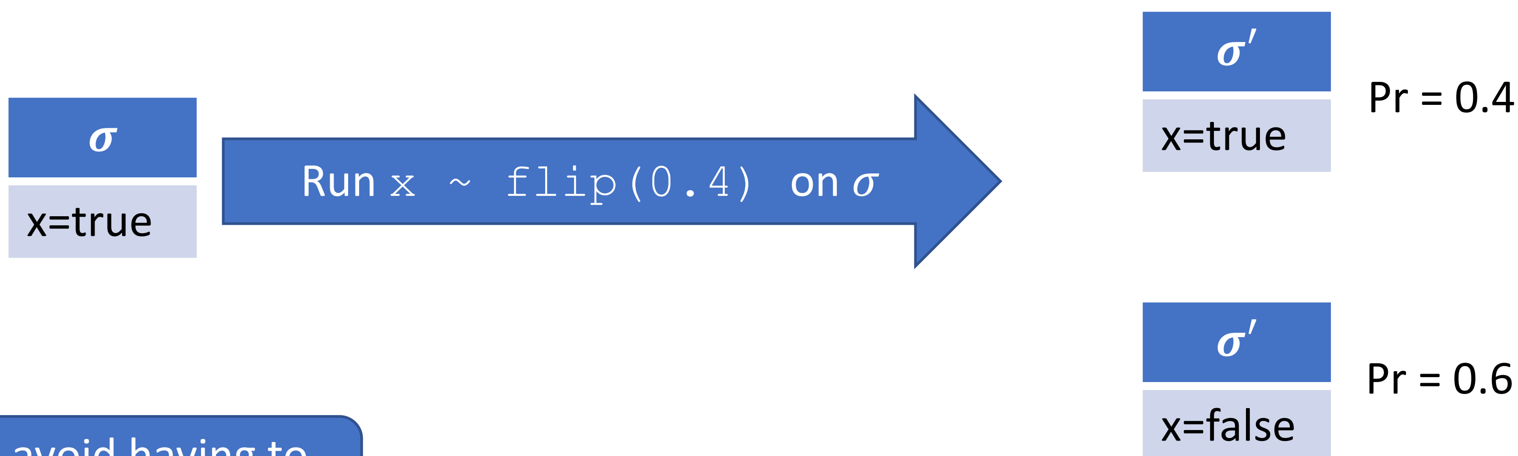
☹

- “Procedural”: the meaning of the program is whatever it executes to …not entirely satisfying…
- A sample is a full execution: a global property that makes it harder to think modularly about local meaning of code

Next: the gold standard in programming languages
denotational semantics

Denotational Semantics

- Idea: We don't have to *run* a flip statement to know what its distribution is
- For some input state σ and output state σ', we can directly compute the *probability of transitioning* from σ to σ' upon executing a flip statement:



We can avoid having to think about sampling!

Denotational Semantics of Flip

Idea: Directly define the probability of transitioning upon executing each statement

Call this its *denotation,* written $[\![\mathtt{s}]\!]$

$$[\![x \sim \mathtt{flip}(\theta)]\!](\sigma' \mid \sigma) \triangleq \begin{cases} \theta & \text{if } \sigma' = \sigma[x \mapsto T] \\ 1-\theta & \text{if } \sigma' = \sigma[x \mapsto F] \\ 0 & \text{otherwise} \end{cases}$$

Semantic bracket: associate semantics with syntax

Output state

Input State

Assign x to false in the state σ

Formal Denotational Semantics

$$[\![v]\!] \triangleq \delta(v) \tag{1}$$

$$[\![\mathtt{fst}\ (v_1, v_2)]\!] \triangleq \delta(v_1) \tag{2}$$

$$[\![\mathsf{snd}\ (v_1, v_2)]\!] \triangleq \delta(v_2) \tag{3}$$

$$[\![\mathtt{if}\ v\ \mathtt{then}\ \mathsf{e}_1\ \mathtt{else}\ \mathsf{e}_2]\!] \triangleq \begin{cases} [\![\mathsf{e}_1]\!] & \text{if } v = \mathsf{T} \\ [\![\mathsf{e}_2]\!] & \text{if } v = \mathsf{F} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$[\![\mathtt{flip}\ r]\!](v) \triangleq \begin{cases} r & \text{if } v = \mathsf{T} \\ 1 - r & \text{if } v = \mathsf{F} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

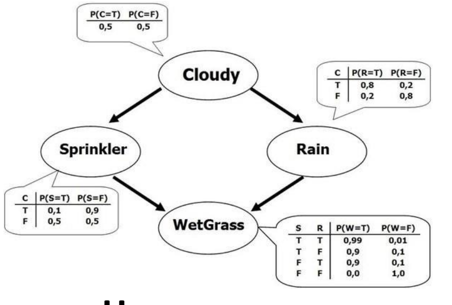
$$[\![\mathsf{observe}\ v_1]\!](v) \triangleq \begin{cases} 1 & \text{if } v_1 = \mathsf{T} \text{ and } v = \mathsf{T}, \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$[\![x(v)]\!] \triangleq T(x)(v) \tag{7}$$

$$[\![\mathtt{let}\ x = \mathsf{e}_1\ \mathtt{in}\ \mathsf{e}_2]\!](v) \triangleq \sum_{v'} [\![\mathsf{e}_1]\!](v') \times [\![\mathsf{e}_2[x \mapsto v']]\!](v) \tag{8}$$

The Challenge of PPL Inference

- Probabilistic inference is #P-*hard*
  - Implies there is likely no universal solution
- In practice inference is often feasible
  - Often relies on conditional independence
  - Manifests as *graph properties*
- *Why exact?*
  1. No error propagation
  2. Approximations are intractable in theory as well
  3. Approximates are known to mislead learners
  4. Core of effective approximation techniques
  5. Unaffected by low-probability observations



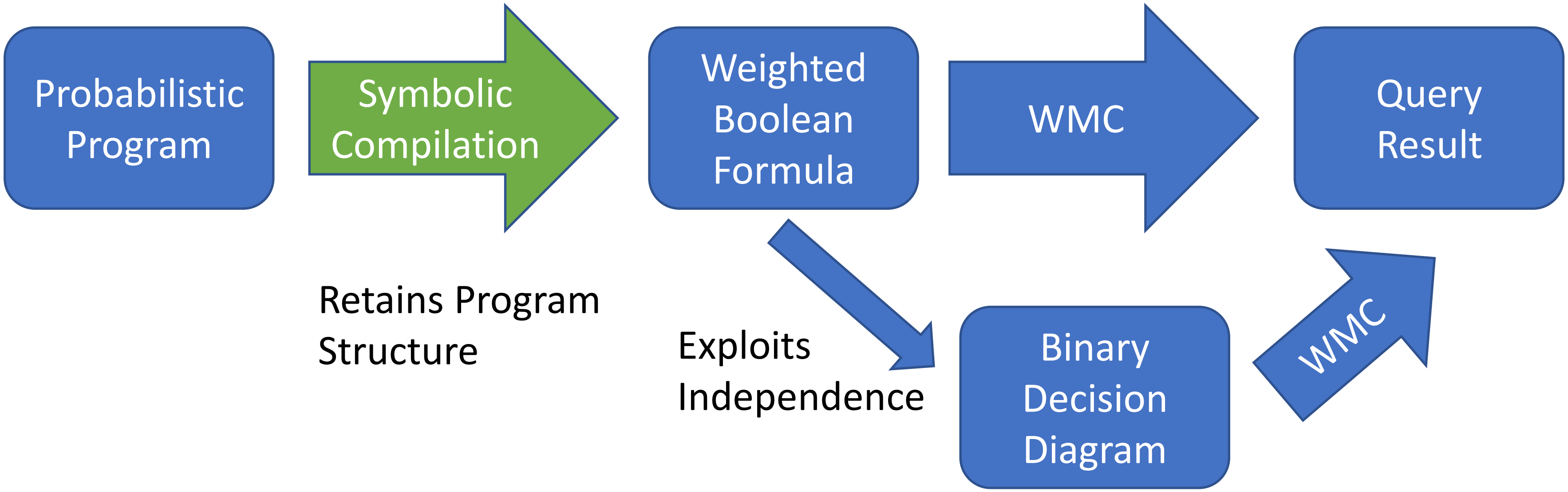
Techniques for exact inference

Exploits independence to decompose inference?

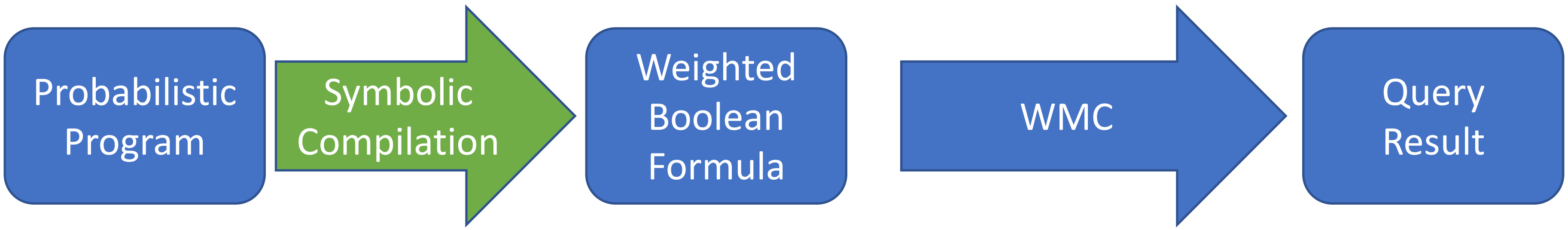
	No	Yes
Yes	**Graphical Model Compilation (Figaro, Infer.Net)**	**Symbolic compilation (Our work)**
No		**Path Enumeration (WebPPL, Psi)**

Keeps program structure?

Our Approach: Symbolic Compilation & WMC



Our Approach: Symbolic Compilation & WMC




```
x := flip(0.4);
```

l	$w(l)$
f_1	0.4
$\overline{f_1}$	0.6

$$(x' \Leftrightarrow f_1)$$

$$\mathrm{WMC}(\varphi, w) = \sum_{m \models \varphi} \prod_{l \in m} w(l).$$

$$\mathrm{WMC}\big((x' \Leftrightarrow f_1) \land x \land x', w\big)?$$

- A single model: $\mathrm{m} = x' \land x \land f_1$
- $w(x') * w(x) * w(f_1) = 0.4$

Provably Correct Compilation

$$\frac{\vdash v : \tau}{\Gamma \vdash v : \tau \rightsquigarrow (F_{\mathbf{r}}(v), \emptyset)} \quad \text{(C-Value)}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau \rightsquigarrow (\mathbf{r} \overset{\tau}{\Longleftrightarrow} \mathbf{x}, \emptyset)} \quad \text{(C-Ident)}$$

$$\frac{\Gamma(x) = \tau_1 \times \tau_2}{\Gamma \vdash \mathtt{fst}\ x : \tau_1 \rightsquigarrow (\mathbf{r} \overset{\tau_1}{\Longleftrightarrow} \mathbf{x}_l, \emptyset)} \quad \text{(C-Fst)}$$

$$\frac{\Gamma(x) = \tau_1 \times \tau_2}{\Gamma \vdash \mathtt{snd}\ x : \tau_2 \rightsquigarrow (\mathbf{r} \overset{\tau_2}{\Longleftrightarrow} \mathbf{x}_r, \emptyset)} \quad \text{(C-Snd)}$$

$$\frac{\Gamma(x_1) = \tau_1 \qquad \Gamma(x_2) = \tau_2}{\Gamma \vdash (x_1, x_2) : \tau_1 \times \tau_2 \rightsquigarrow (\mathbf{r}_l \overset{\tau_1}{\Longleftrightarrow} \mathbf{x}_1 \wedge \mathbf{r}_r \overset{\tau_2}{\Longleftrightarrow} \mathbf{x}_2, \emptyset)} \quad \text{(C-Tup)}$$

$$\frac{\Gamma \vdash \mathtt{e}_1 : \tau_1 \rightsquigarrow (\varphi_1, w_1) \qquad \Gamma \cup \{x : \tau_1\} \vdash \mathtt{e}_2 : \tau_2 \rightsquigarrow (\varphi_2, w_2)}{\Gamma \vdash \mathtt{let}\ x : \tau_1 = \mathtt{e}_1\ \mathtt{in}\ \mathtt{e}_2 : \tau_2 \rightsquigarrow (\exists \mathbf{x}.(\varphi_1[\mathbf{r} \overset{\tau_1}{\longmapsto} \mathbf{x}] \wedge \varphi_2), w_1 \uplus w_2)} \quad \text{(C-Let)}$$

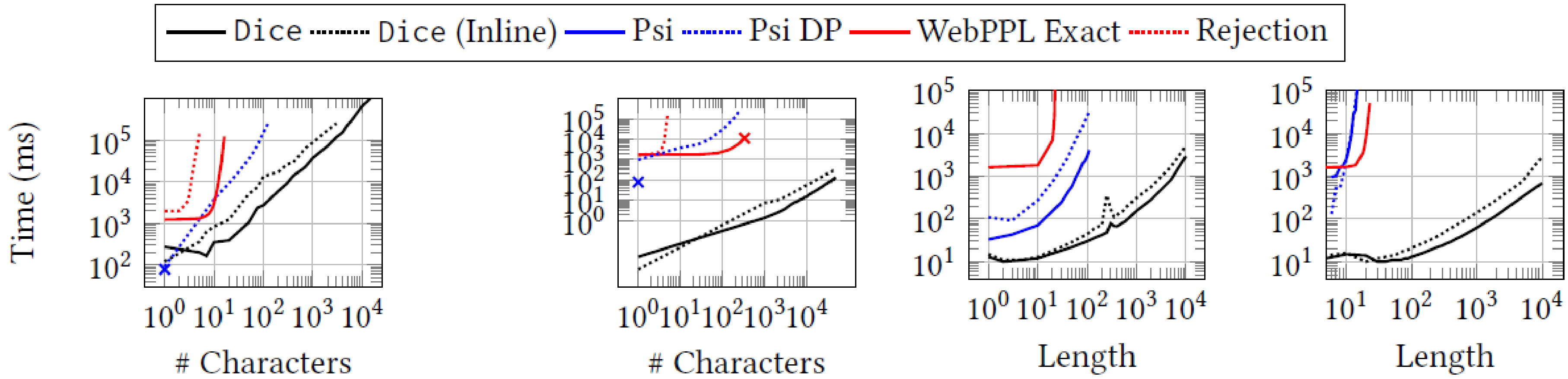
$$\frac{\text{fresh } \mathbf{f}}{\Gamma \vdash \mathtt{flip}\ r : \mathbf{Bool} \rightsquigarrow \left(\mathbf{r} \Leftrightarrow \mathbf{f}, (\mathbf{f} \mapsto r, \overline{\mathbf{f}} \mapsto 1 - r)\right)} \quad \text{(C-Flip)}$$

$$\frac{\Gamma \vdash g : \mathbf{Bool} \rightsquigarrow (\varphi_g, \emptyset) \qquad \Gamma \vdash \mathtt{e}_T : \tau \rightsquigarrow (\varphi_T, w_T) \qquad \Gamma \vdash \mathtt{e}_E : \tau \rightsquigarrow (\varphi_E, w_E)}{\Gamma \vdash \mathtt{if}\ g\ \mathtt{then}\ \mathtt{e}_T\ \mathtt{else}\ \mathtt{e}_E : \tau \rightsquigarrow \Big(\big((\varphi_g \mid F_{\mathbf{r}}(\mathsf{T})) \wedge \varphi_T\big) \vee \big((\varphi_g \mid F_{\mathbf{r}}(\mathsf{F})) \wedge \varphi_E\big), w_T \uplus w_E\Big)} \quad \text{(C-Ite)}$$

$$\frac{\Gamma \vdash g : \mathbf{Bool} \rightsquigarrow (\varphi, \emptyset)}{\Gamma \vdash \mathtt{observe}\ g : \mathbf{Bool} \rightsquigarrow (\varphi \wedge \mathbf{r}, \emptyset)} \quad \text{(C-Obs)}$$

$$\frac{\Phi(x_1) = (\mathbf{x}_{arg}, \varphi, w) \qquad \Gamma(x_1) = \tau_1 \longrightarrow \tau_2 \qquad \Gamma(x_2) = \tau_1 \qquad (\varphi', w') = \mathtt{RefreshFlips}(\varphi, w)}{\Gamma \vdash x_1(x_2) : \tau_2 \rightsquigarrow (\varphi'[\mathbf{x}_{arg} \overset{\tau_1}{\longmapsto} \mathbf{x}_2], w')} \quad \text{(C-FCall)}$$

Benchmarks



(a) Caesar cipher with errors. **(b)** Caesar cipher no errors. **(c)** Diamond network. **(d)** Ladder network.

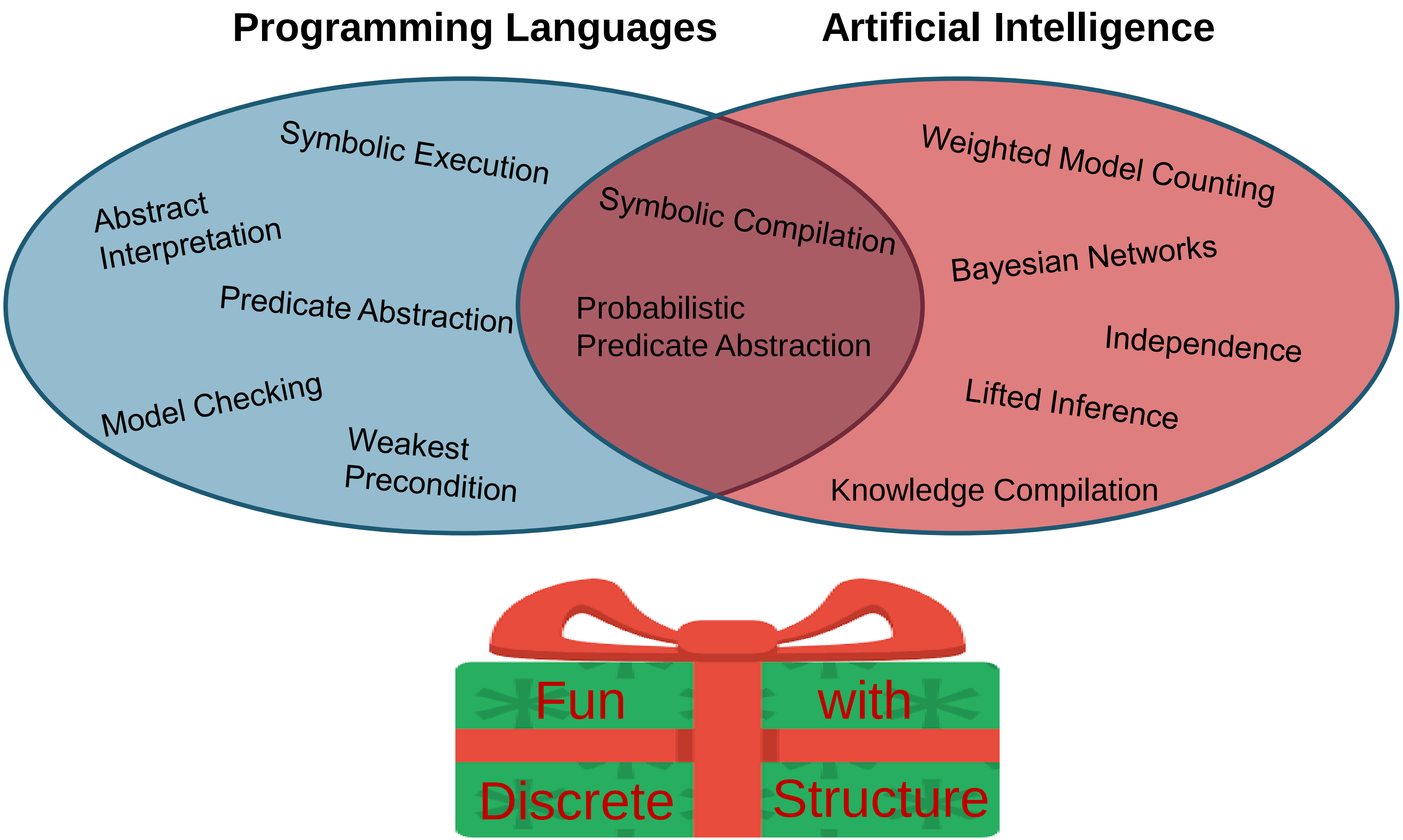
Benchmarks

#	Benchmark	# Paths	Default Psi	DP Psi	Dice	BDD Size
1	Grass	$10^{2.41}$	154	64	**1.06**	15
2	Burglar Alarm	$10^{1.98}$	152	10	**1.06**	11
3	Coin Bias	$10^{0.60}$	49	26	**0.993**	
4	Noisy Or	$10^{4.21}$	744	153	**1.11**	35
5	Evidence1	$10^{0.90}$	48	32	**1.05**	5
6	Evidence2	$10^{0.90}$			**1.07**	6
7	Murder Mystery	$10^{1.20}$	70	29	**1.03**	6
8	Digit Recognition	$10^{237.70}$	$3.6 \cdot 10^5$	4539	**70.29**	7896
9	Cancer [43]	$10^{3.06}$	455	85	**1.22**	46
10	Alarm [3]	$10^{36.01}$	✗	✗	**1058.265**	437658
11	Hailfinder [1]	$10^{76.26}$	✗	✗	**5529.999**	213745
12	Survey	$10^{4.14}$			**1.184**	116
13	Insurance [4]	$10^{40.92}$	✗	✗	**847.514**	232111
14	Hepar2 [52]	$10^{69.45}$	✗	✗	**204.067**	54860
15	Pigs	$10^{492.86}$			**465.597**	265379
16	Water	$10^{54.50}$			**138.966**	68352
17	Munin	$10^{1610.98}$			$2.62 \cdot 10^5$	22830303

Second Conclusions

- New state-of-the-art system for discrete probabilistic programs
- Exact inference yet very scalable
- Provably correct
- Modular compilation-based inference
- Try Dice out: https://github.com/SHoltzen/dice

Third Conclusions



Final Conclusions

Pure Logic **Probabilistic World Models** **Pure Learning**

Bring high-level
representations, general
knowledge, and
efficient high-level reasoning
to probabilistic models

References

- **Tal Friedman** and Guy Van den Broeck**.** Probabilistic Databases Meets Relational Embeddings: Symbolic Querying of Vector Spaces (coming soon)
- **Steven Holtzen, Todd Millstein** and Guy Van den Broeck. Symbolic Exact Inference for Discrete Probabilistic Programs, *In Proceedings of the ICML Workshop on Tractable Probabilistic Modeling (TPM)*, 2019.
- Steven Holtzen, Guy Van den Broeck and Todd Millstein. Sound Abstraction and Decomposition of Probabilistic Programs, *In Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.
- Steven Holtzen, Todd Millstein and Guy Van den Broeck. Probabilistic Program Abstractions, *In Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017.
- https://github.com/SHoltzen/dice

…with slides stolen from Steven Holtzen and Tal Friedman.

Thanks