Lifted Probabilistic Inference in Relational Models

Guy Van den Broeck Dan Suciu
KU Leuven U. of Washington

About the Tutorial

Slides available online.
Bibliography is at the end.
Your speakers:

http://www.guyvdb.eu/
https://homes.cs.washington.edu/~suciu/

I work in AI

I work in DB

About the Tutorial

- The tutorial is about
  - deep connections between AI and DBs
  - a unified view on probabilistic reasoning
  - a logical approach to Lifted Inference

- The tutorial is NOT an exhaustive overview of lifted algorithms for graphical models (see references at the end)

Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

Part 1: Motivation

- Why do we need relational representations of uncertainty?

- Why do we need lifted inference algorithms?

Why Relational Data?

- Our data is already relational!
  - Companies run relational databases
  - Scientific data is relational:
    - Large Hadron Collider generated 25PB in 2012
    - LSST Telescope will produce 30TB per night

- Big data is big business:
  - Oracle: $7.1BN in sales
  - IBM: $3.2BN in sales
  - Microsoft: $2.6BN in sales

≈ GDP of Iceland!

[Gartner'06]

Why Probabilistic Relational Data?

- Relational data is increasingly probabilistic
  - NELL machine reading (>50M tuples)
  - Google Knowledge Vault (>2BN tuples)
  - DeepDive (>7M tuples)

- Data is inferred from unstructured information using statistical models
  - Learned from the web, large text corpora, ontologies, etc.
  - The learned/extracted data is relational

[Carlson'10, Dong'14, Niu'12]

Representation: Probabilistic Databases

- Tuple-independent probabilistic databases

Actor:

Name	Prob
Brando	0.9
Cruise	0.8
Coppola	0.1

WorkedFor:

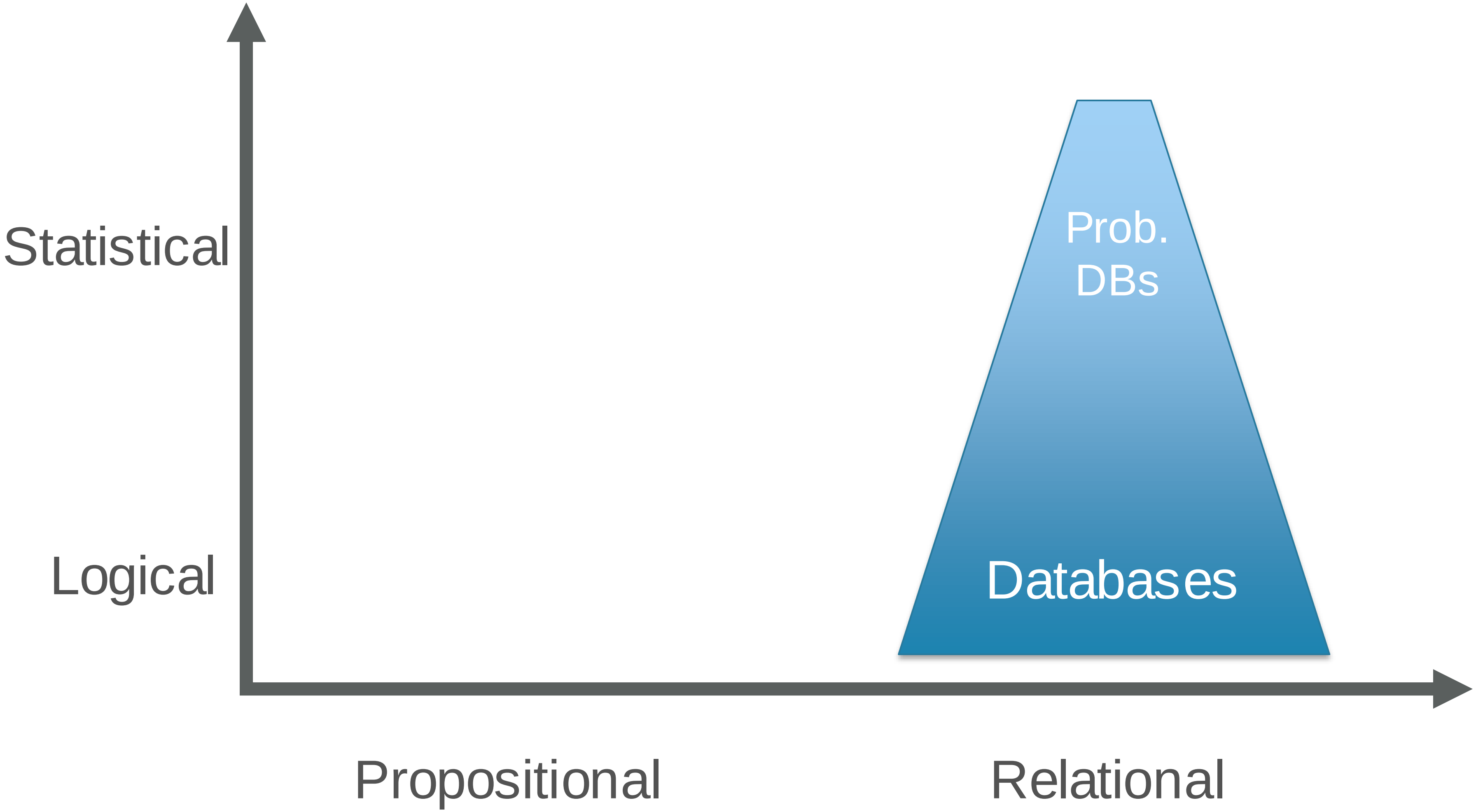
Actor	Director	Prob
Brando	Coppola	0.9
Coppola	Brando	0.2
Cruise	Coppola	0.1

- Query: SQL or First Order Logic

```
SELECT Actor.name
FROM Actor, WorkedFor
WHERE Actor.name = WorkedFor.actor
```

$$Q(x) = \exists y\ \mathrm{Actor}(x) \wedge \mathrm{WorkedFor}(x,y)$$

Summary


Statistical
Logical
Prob.
DBs
Databases
Propositional
Relational

Representations in AI and ML

Statistical

Cloudy	Rain	P(Rain|Cloudy)
T	T	0.80
T	F	0.20
F	T	0.01
F	F	0.99

Logical

Rain $\Rightarrow$ Cloudy

Propositional

Relational

Relational Representations

- Example: First-Order Logic

$$\forall x,y,\ \overbrace{\underbrace{\text{Smokes}(x)}_{\text{Atom}} \wedge \text{Friends}(x,\underset{\substack{\uparrow\\ \text{Logical Variable}}}{y}) \Rightarrow \text{Smokes}(y)}^{\text{Formula}}$$

- Logical variables have domain of constants

  x,y range over domain People = {Alice,Bob}

- Ground formula has no logical variables

  Smokes(Alice) ∧ Friends(Alice,Bob) ⇒ Smokes(Bob)

Representations in AI and ML

Statistical

Cloudy	Rain	P(Rain|Cloudy)
T	T	0.80
T	F	0.20
F	T	0.01
F	F	0.99

Logical

Rain ⇒ Cloudy

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

Propositional

Relational

Why Statistical Relational Models?

- Probabilistic graphical models
  - ✔ Quantify uncertainty and noise
  - ✖ Not very expressive
    Rules of chess in ~100,000 pages
- First-order logic
  - ✔ Very expressive
    Rules of chess in 1 page
  - ✔ Good match for abundant relational data
  - ✖ Hard to express uncertainty and noise

Example: Markov Logic

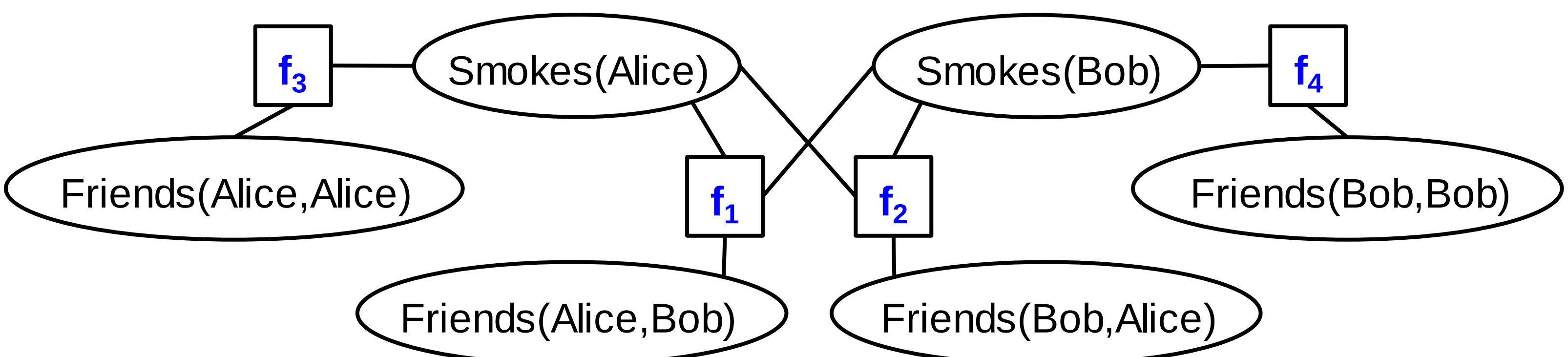
- Weighted First-Order Logic

Weight or Probability

$$3.14 \quad \overbrace{\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)}^{\text{FOL Formula}}$$

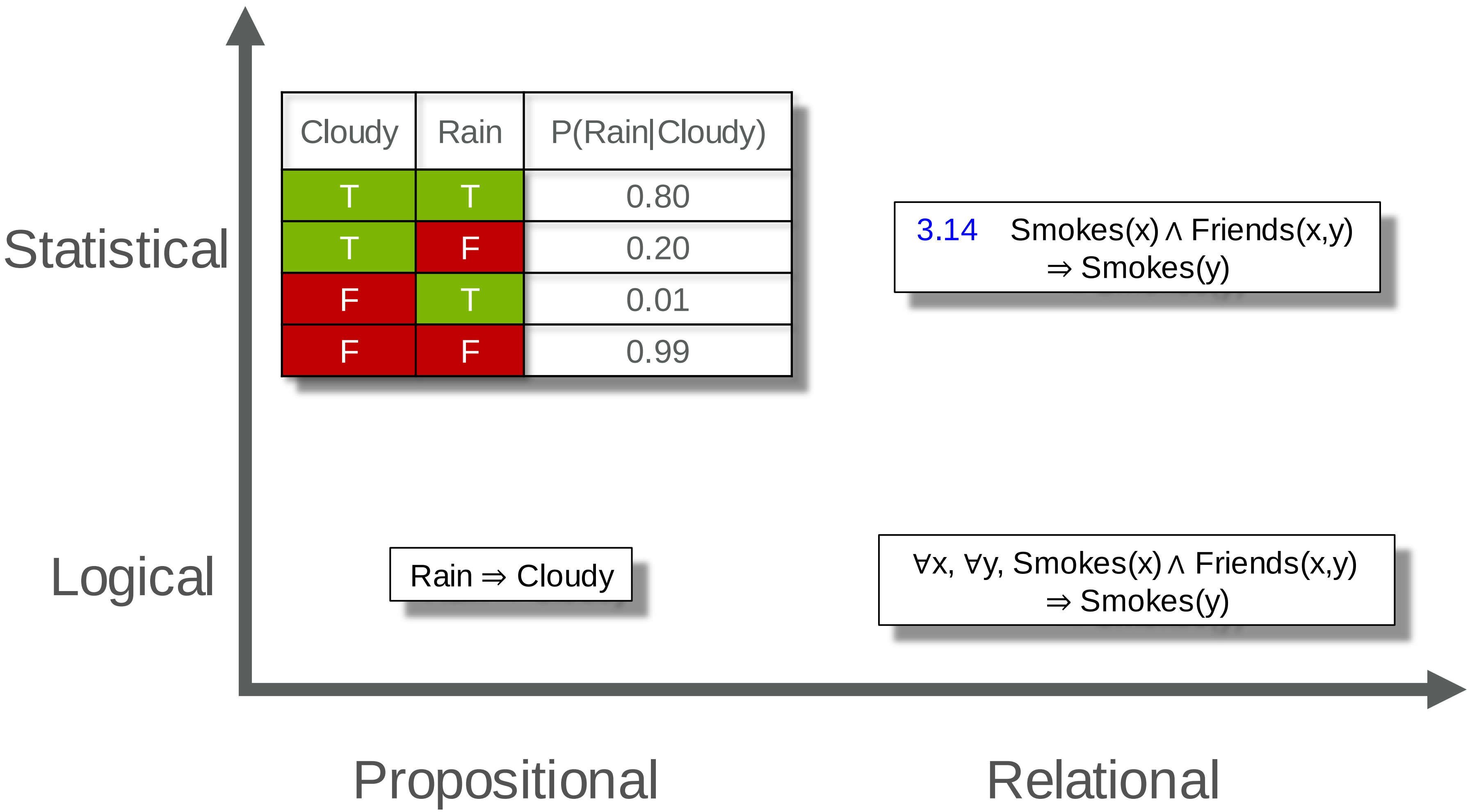
- Ground atom/tuple = random variable in {true,false}

  e.g., Smokes(Alice), Friends(Alice,Bob), etc.
- Ground formula = factor in propositional factor graph



[Richardson'06]

Representations in AI and ML



Cloudy	Rain	P(Rain|Cloudy)
T	T	0.80
T	F	0.20
F	T	0.01
F	F	0.99

Representations in AI and ML

Statistical

Cloudy	Rain	P(Rain|Cloudy)
T	T	0.80
T	F	0.20
F	T	0.01
F	F	0.99

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

Logical

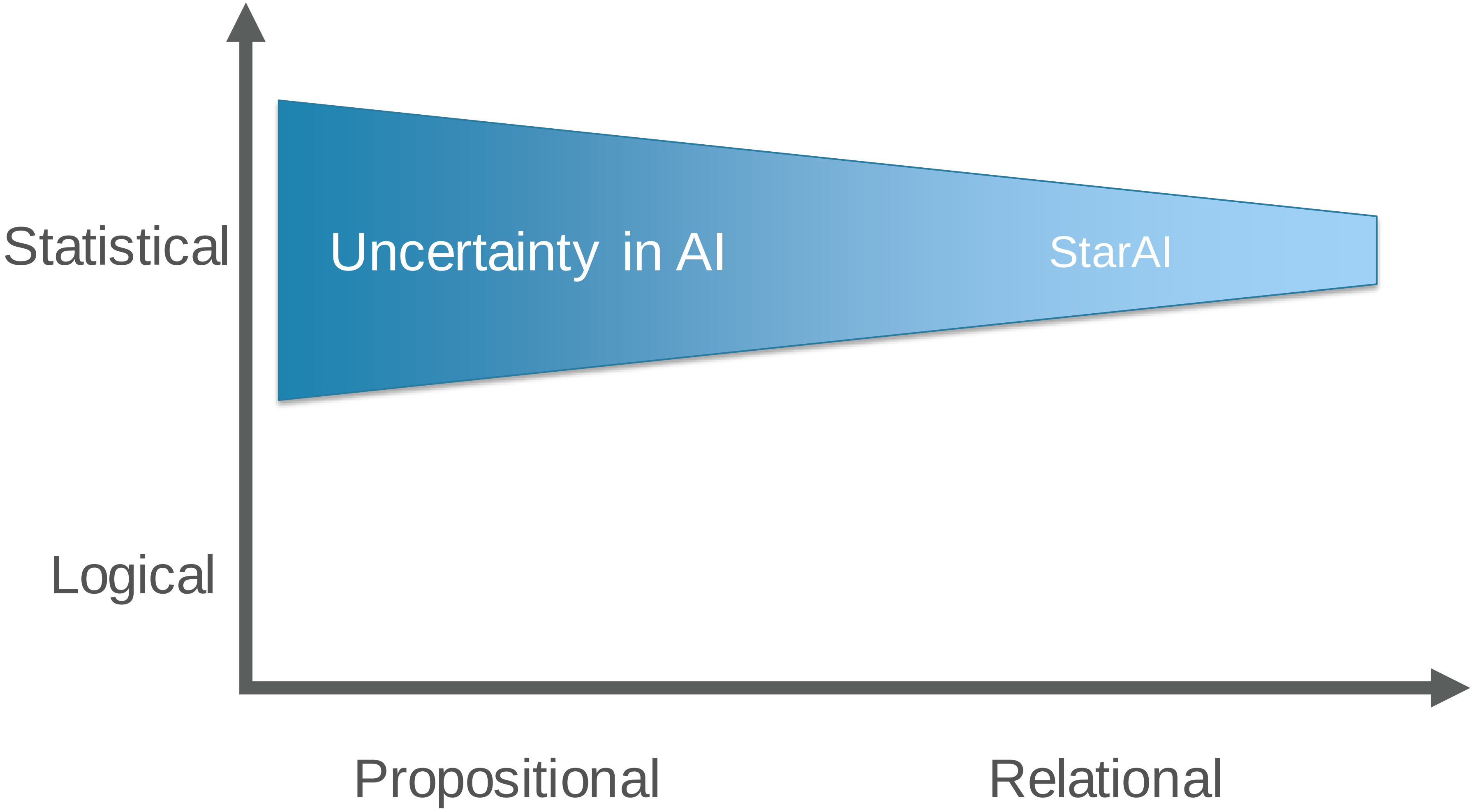
Rain ⇒ Cloudy

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

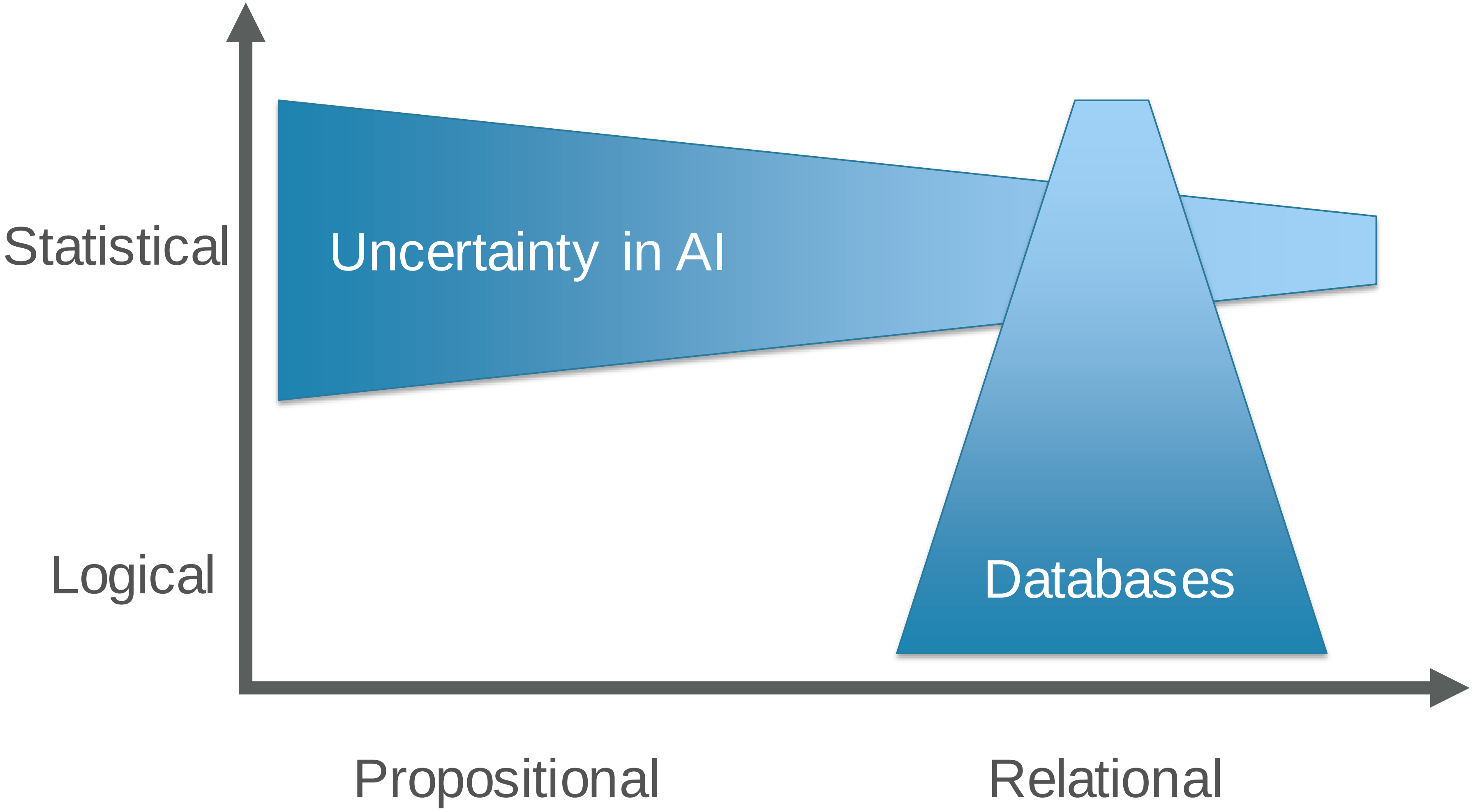
Propositional

Relational

Summary

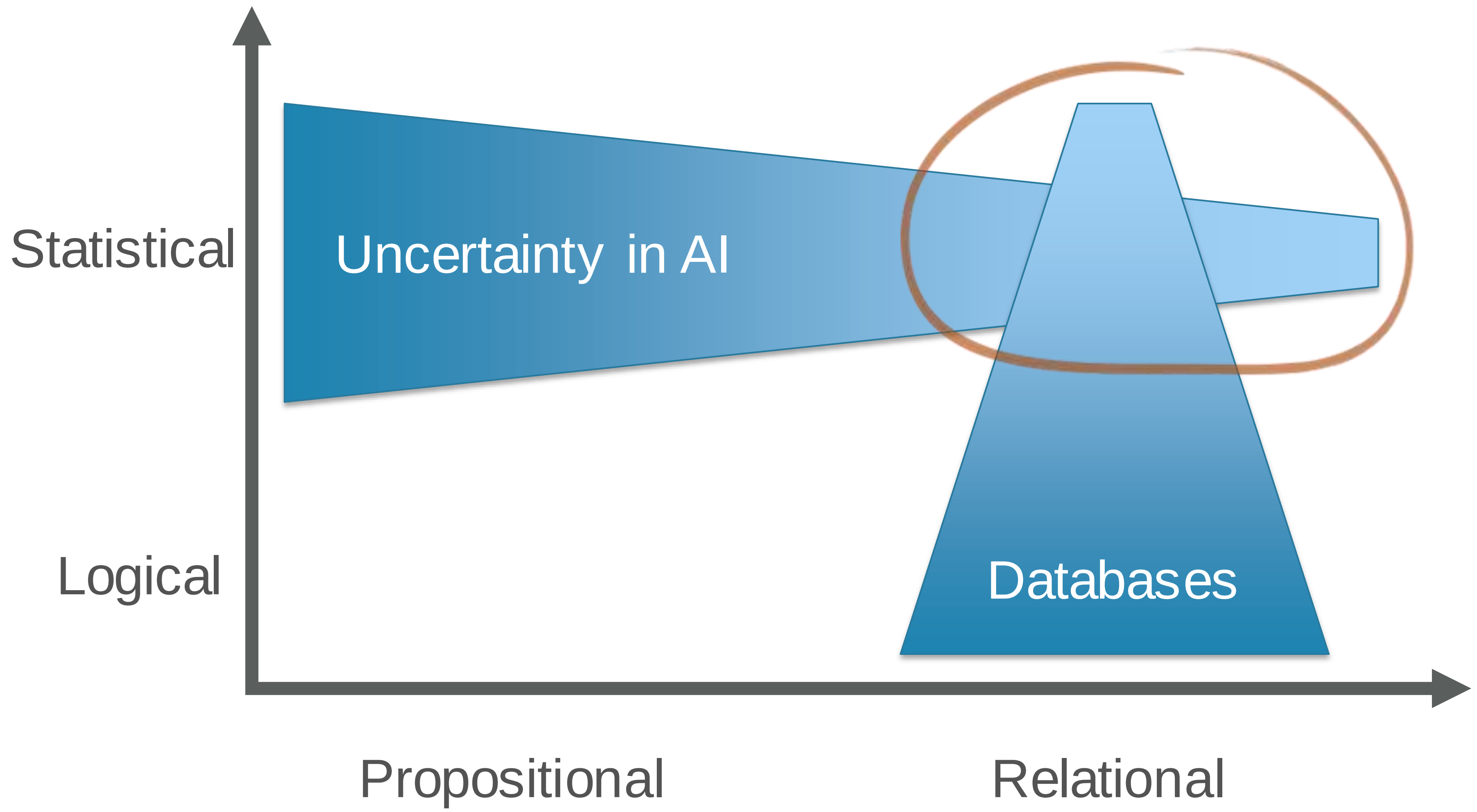

Statistical
Logical
Uncertainty in AI
StarAI
Propositional
Relational

Summary



Statistical
Logical
Uncertainty in AI
Databases
Propositional
Relational

Summary



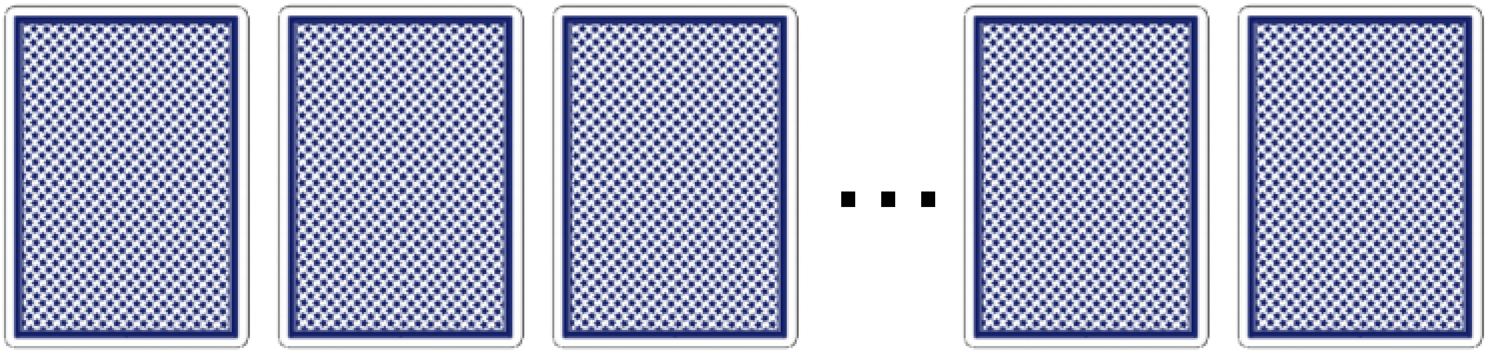
Statistical
Logical
Uncertainty in AI
Databases
Propositional
Relational

Lifted Inference

- Main idea: exploit high level relational representation to speed up reasoning

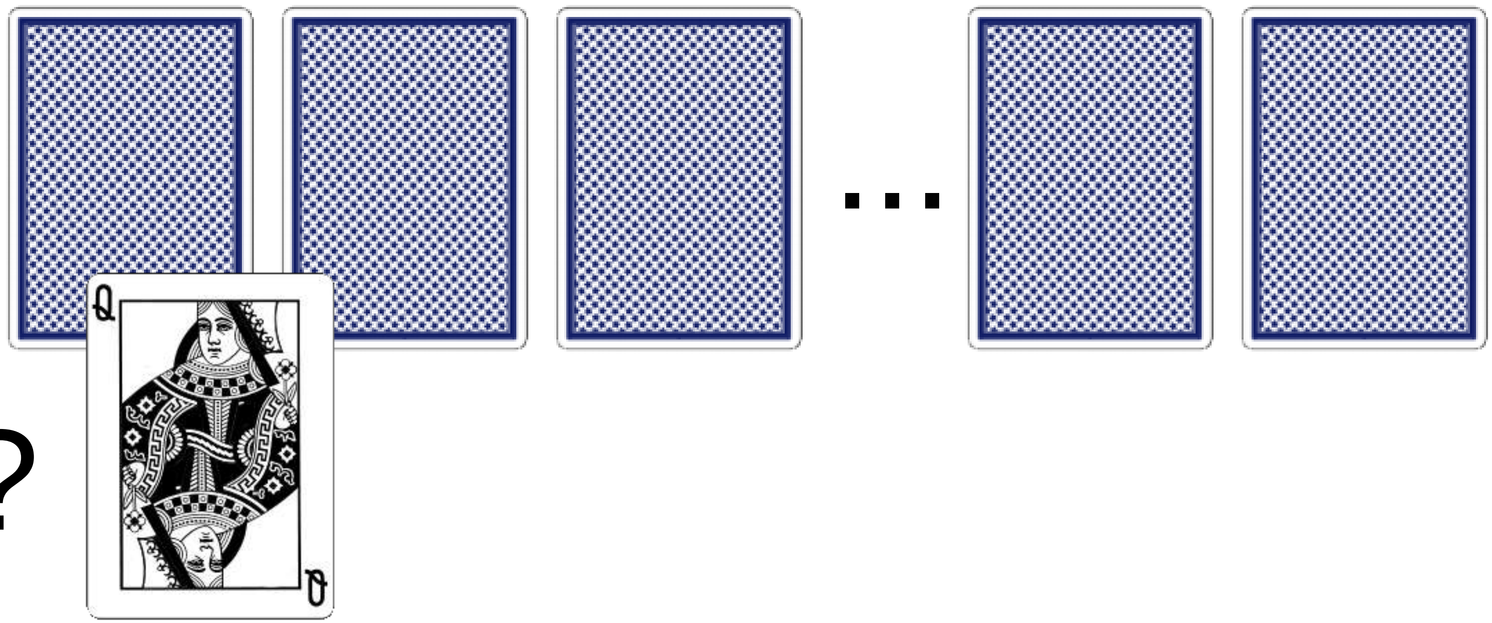
- Let’s see an example…

A Simple Reasoning Problem



- 52 playing cards
- Let us ask some simple questions

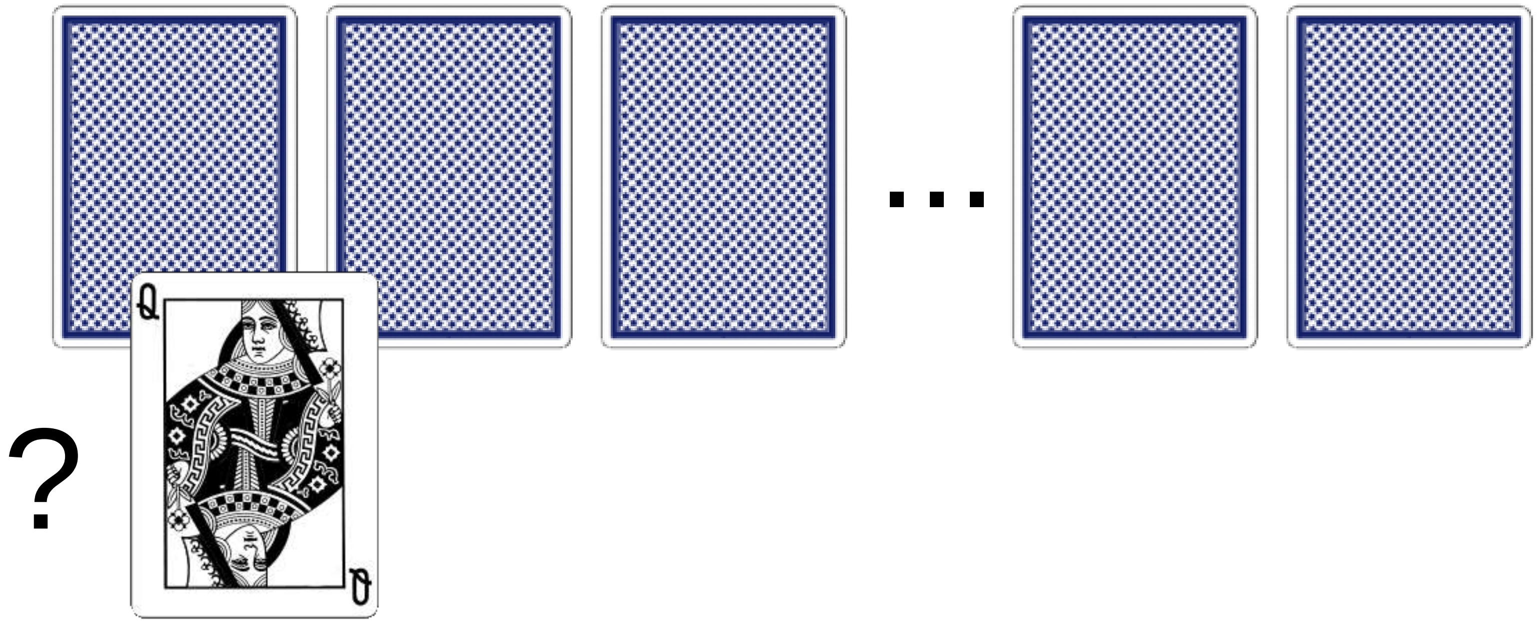
A Simple Reasoning Problem



?

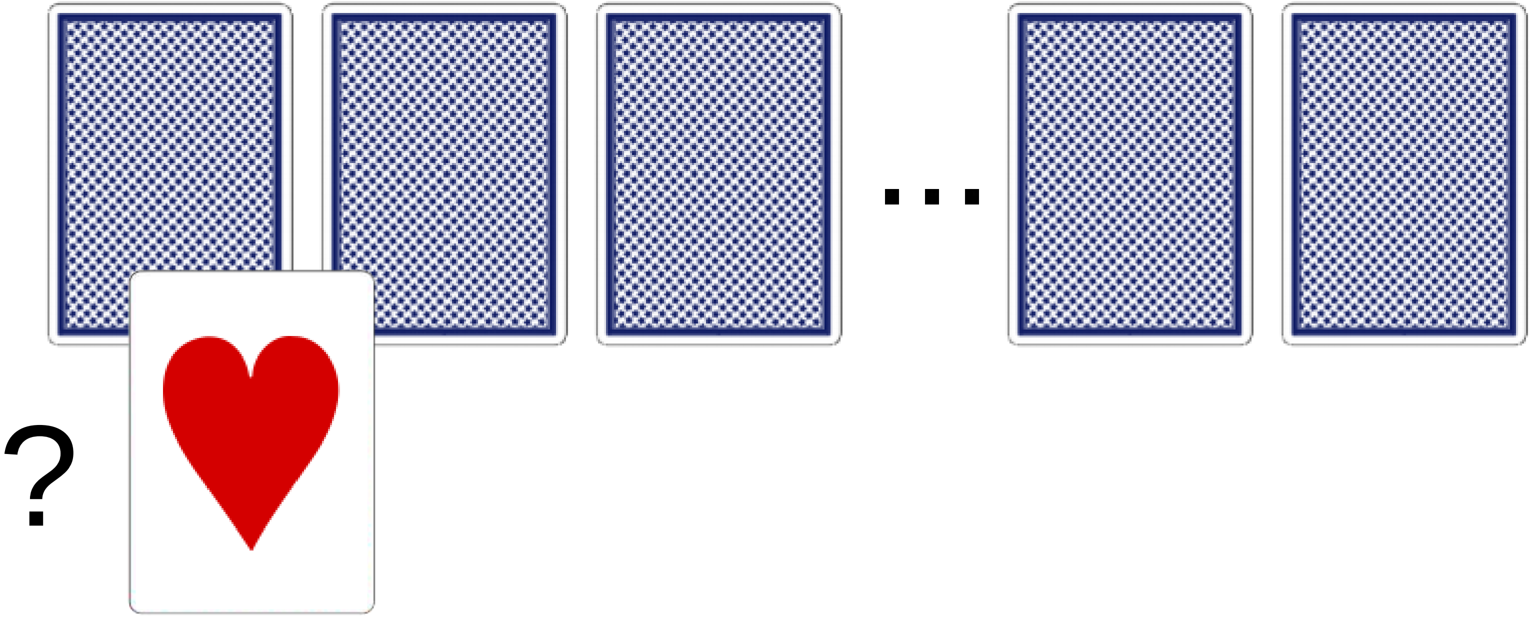
Probability that Card1 is Q?

A Simple Reasoning Problem



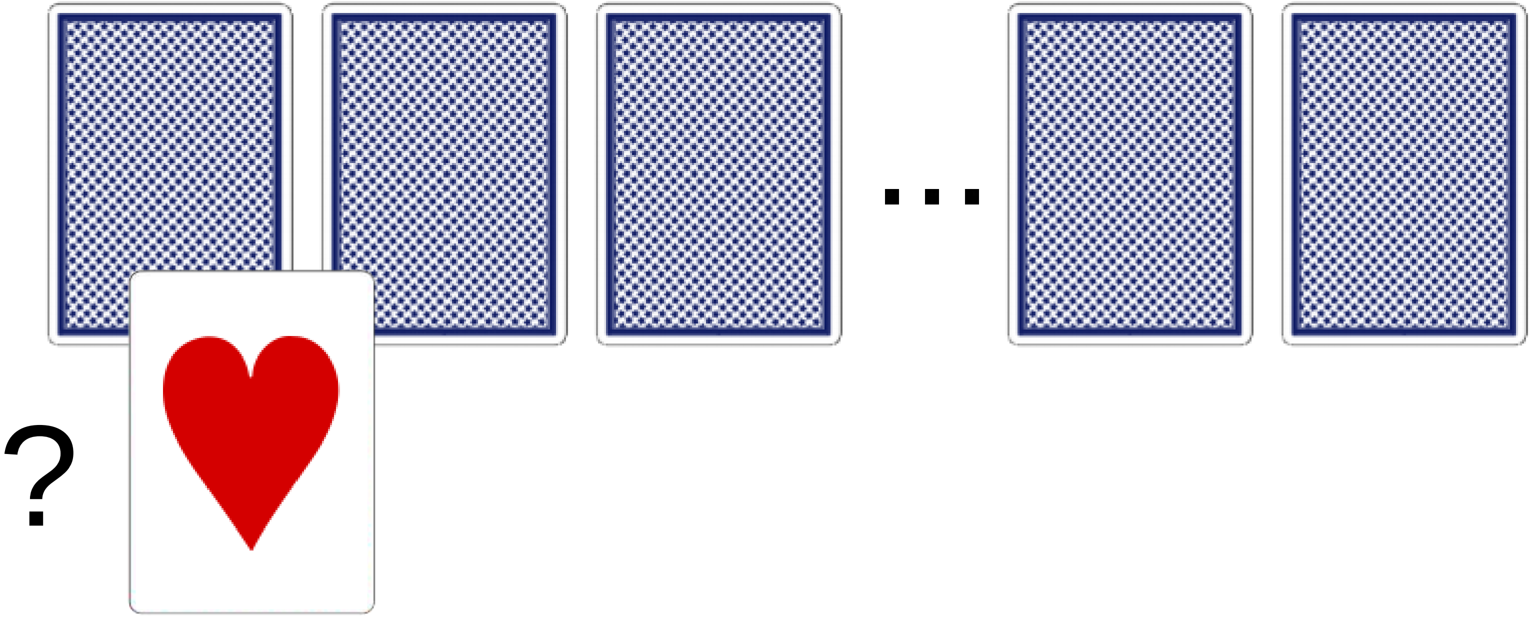
Probability that Card1 is Q? 1/13

A Simple Reasoning Problem



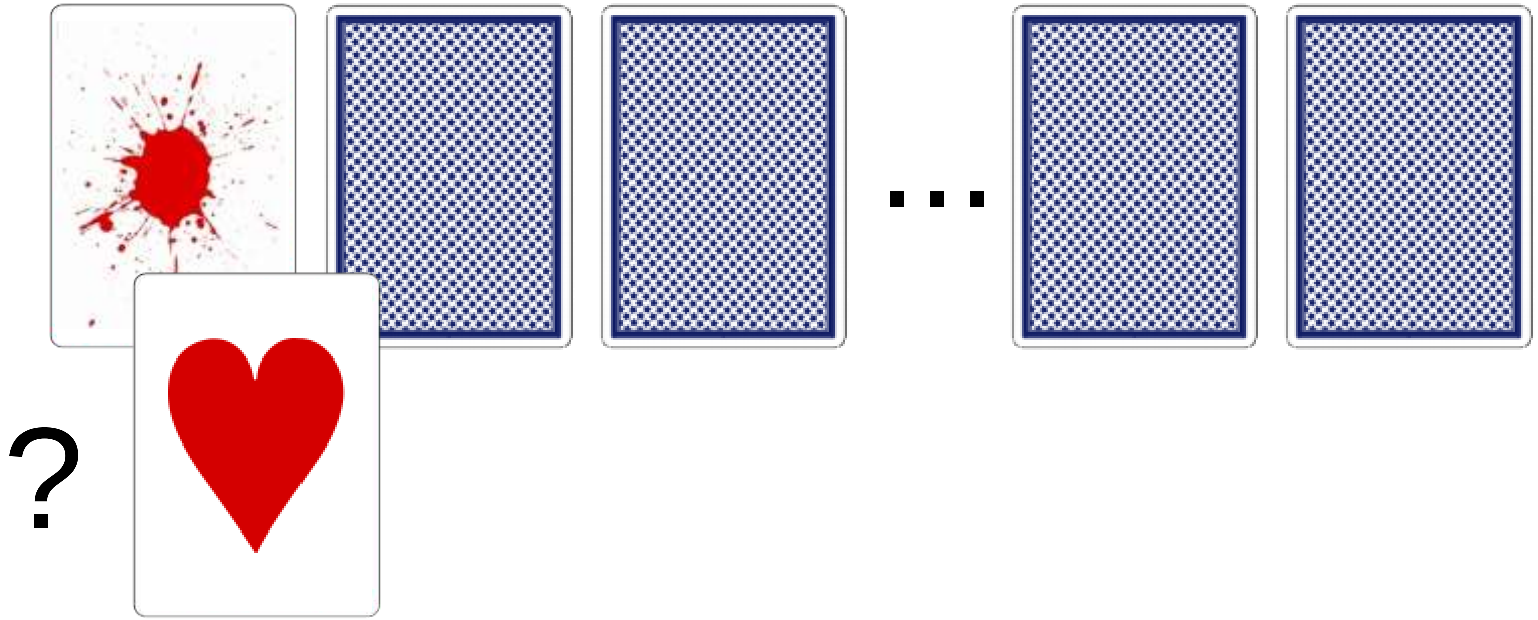
Probability that Card1 is Hearts?

A Simple Reasoning Problem



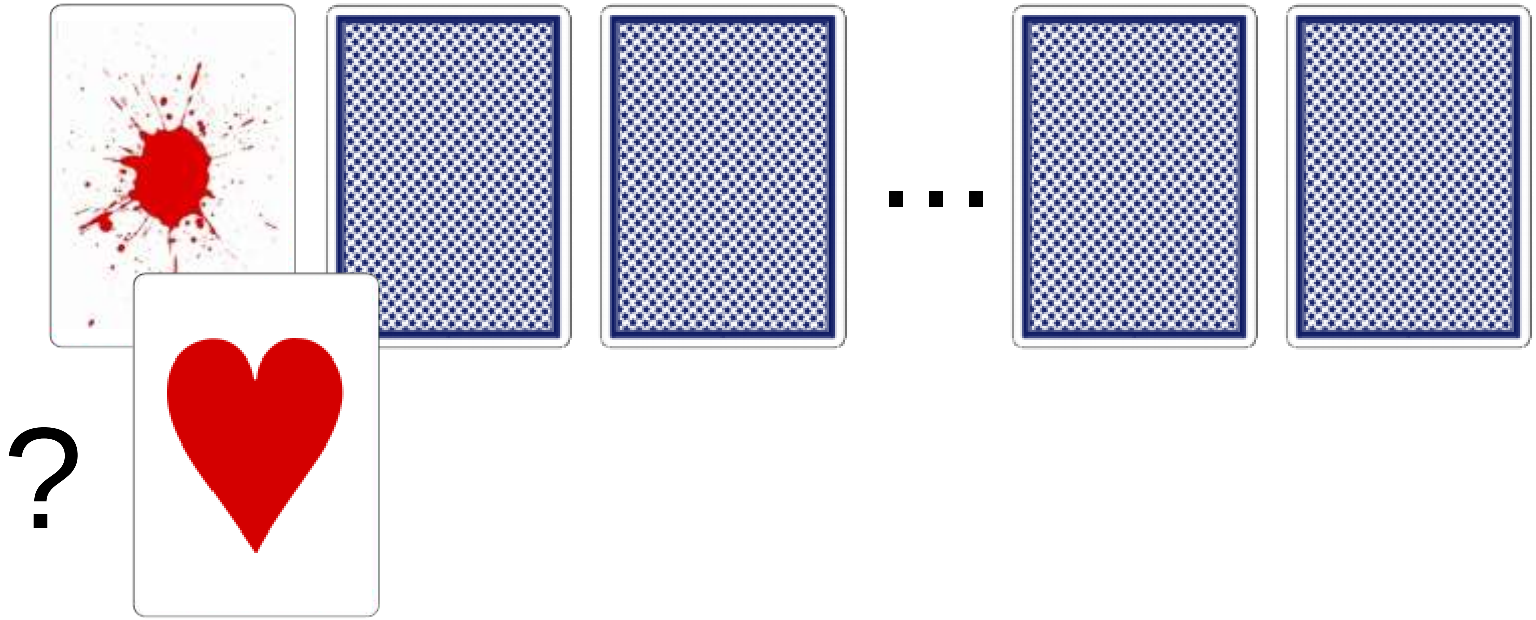
Probability that Card1 is Hearts? 1/4

A Simple Reasoning Problem



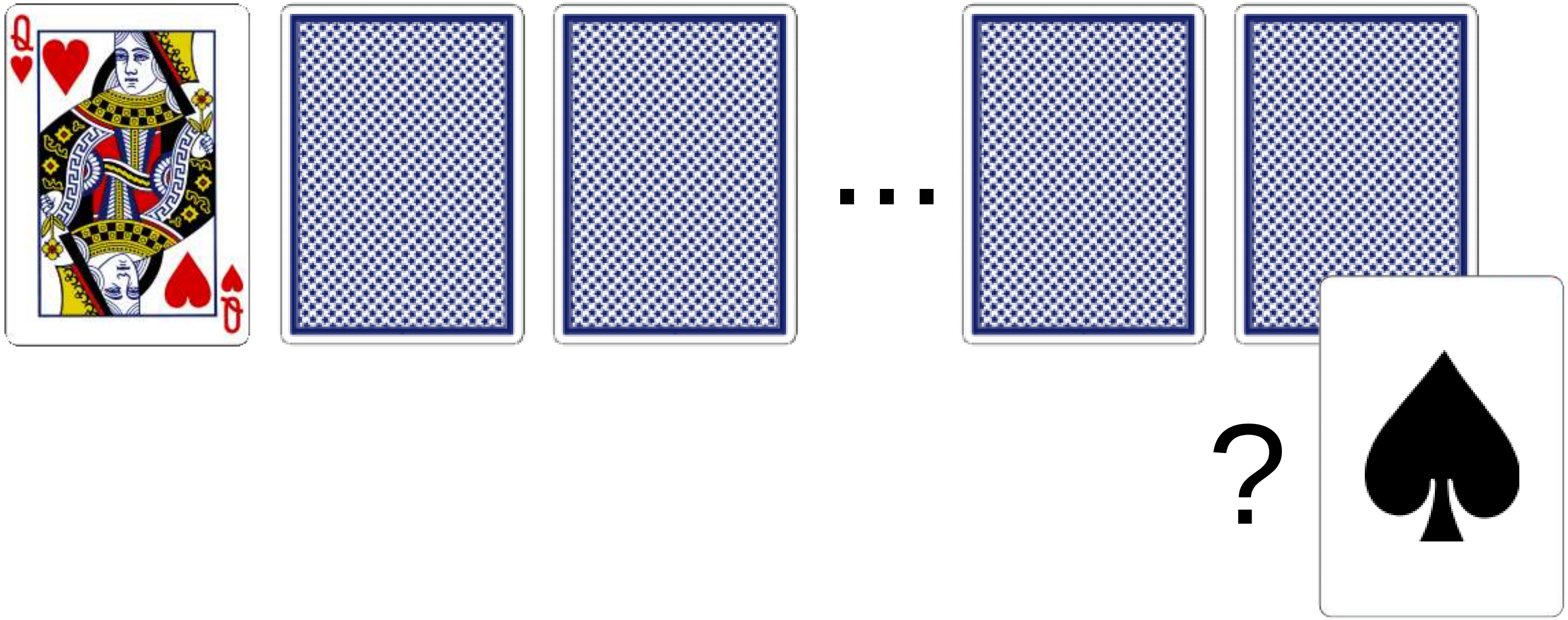
Probability that Card1 is Hearts given that Card1 is red?

A Simple Reasoning Problem



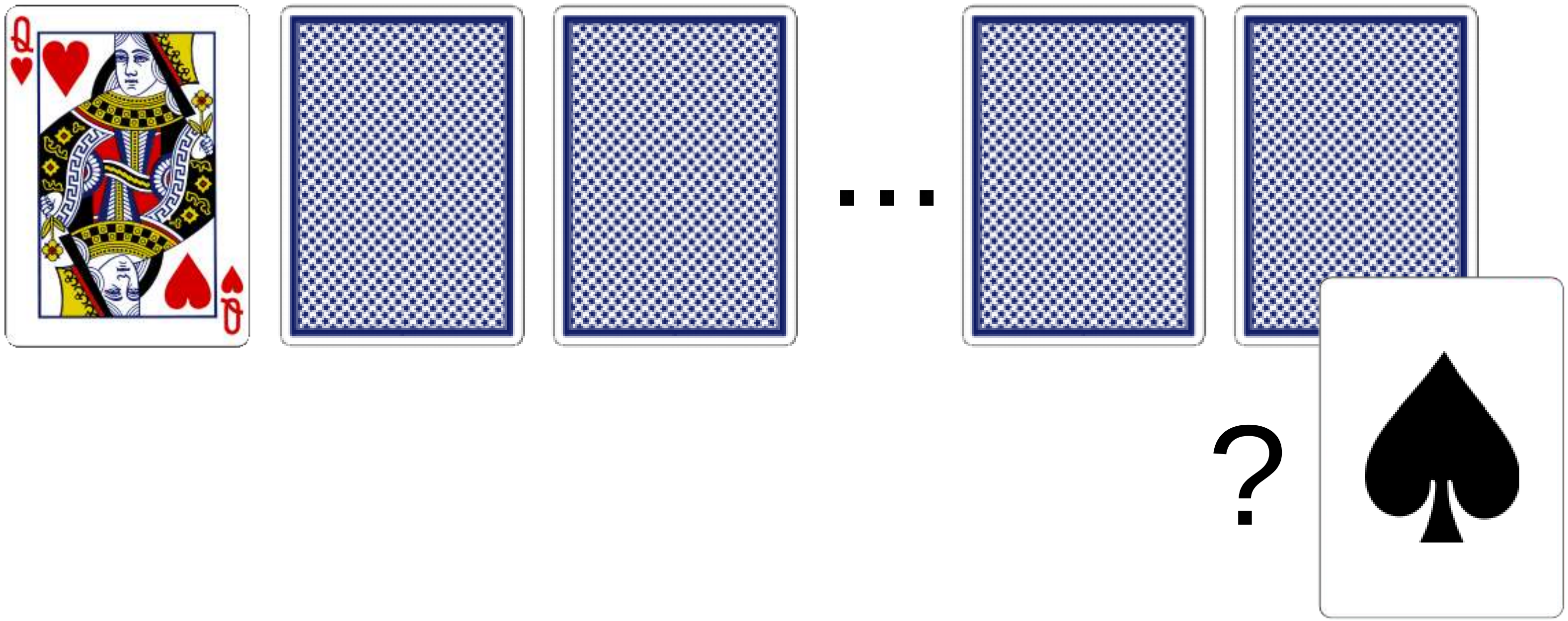
Probability that Card1 is Hearts given that Card1 is red? 1/2

A Simple Reasoning Problem



Probability that Card52 is Spades given that Card1 is QH?

A Simple Reasoning Problem



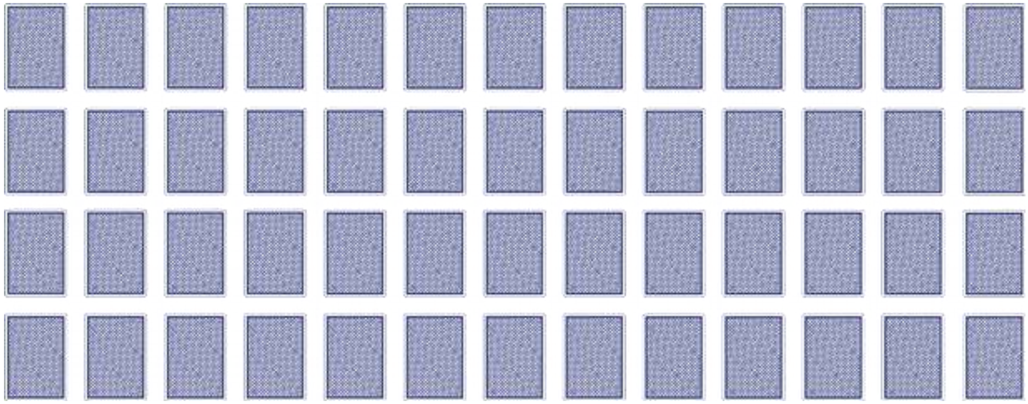
Probability that Card52 is Spades given that Card1 is QH?

13/51

Automated Reasoning

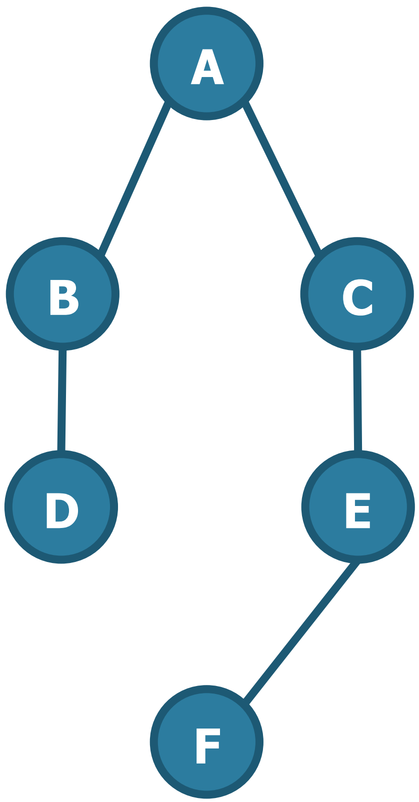
Let us automate this:

1. Probabilistic graphical model (e.g., factor graph)

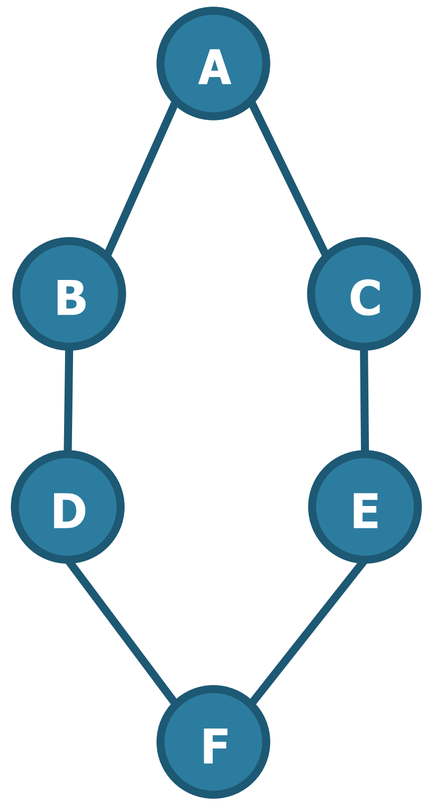


2. Probabilistic inference algorithm
   (e.g., variable elimination or junction tree)

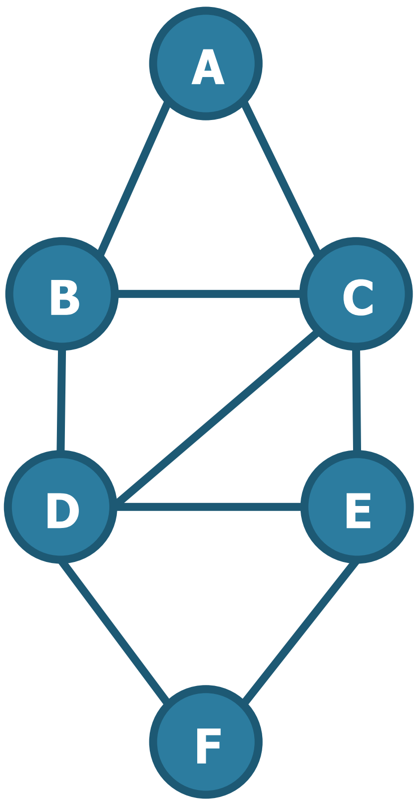
Reasoning in Propositional Models



Tree



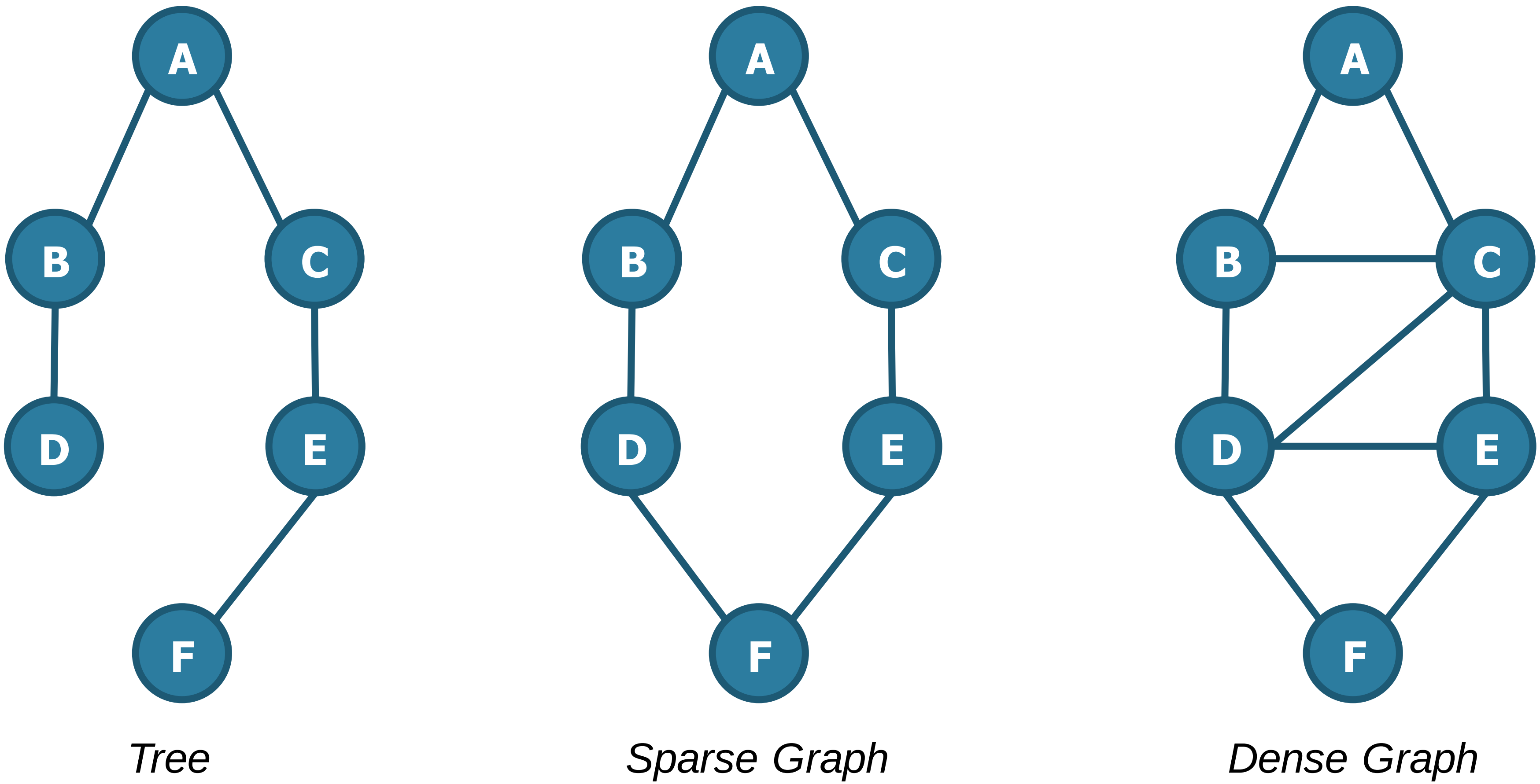
Sparse Graph



Dense Graph

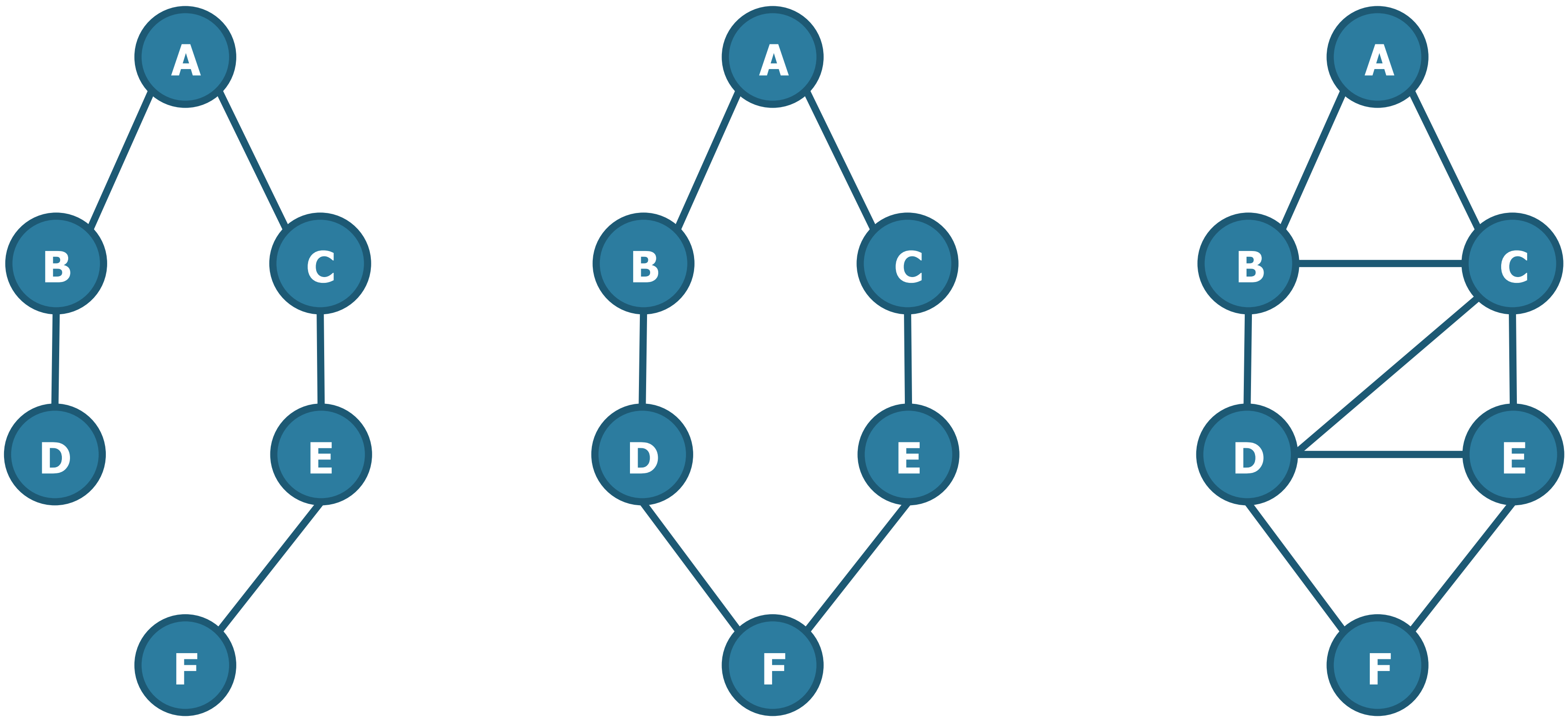
A key result: **Treewidth**   Why?

Reasoning in Propositional Models



A key result: **Treewidth**   Why?   **Conditional Independence!**

Reasoning in Propositional Models

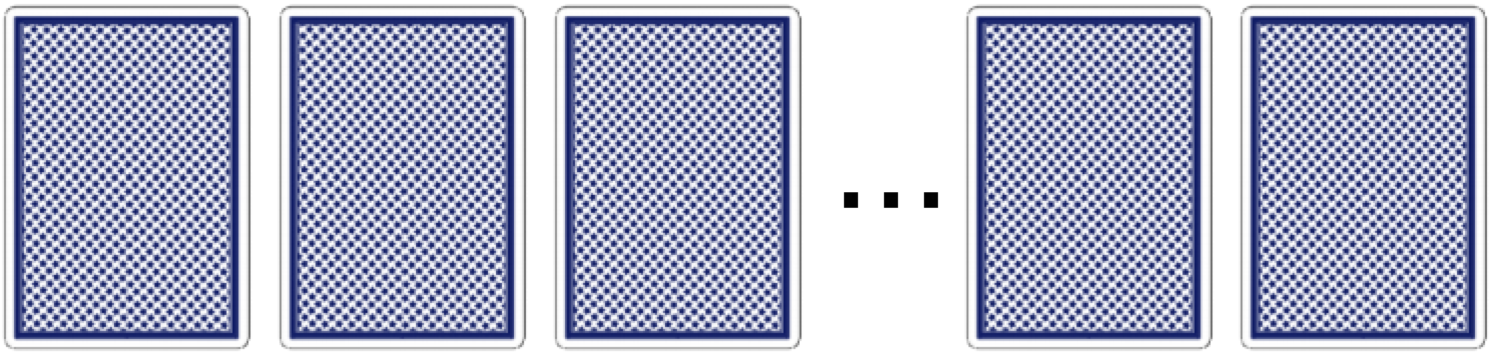


Tree *Sparse Graph* *Dense Graph*

A key result: **Treewidth** Why? **Conditional Independence!**

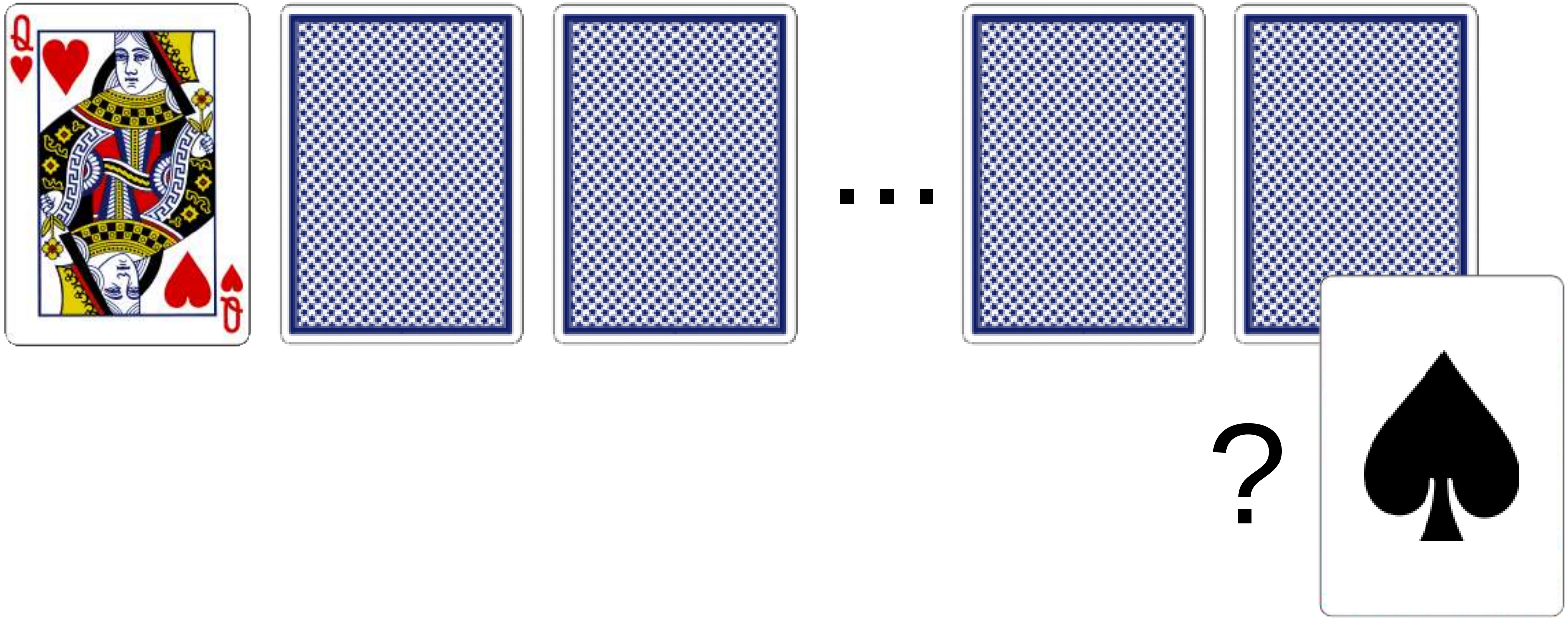
$P(A|C,E) = P(A|C)$ $P(A|B,E,F) = P(A|B,E)$ $P(A|B,E,F) \neq P(A|B,E)$

Is There Conditional Independence?



$$P(\text{Card52} \mid \text{Card1}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

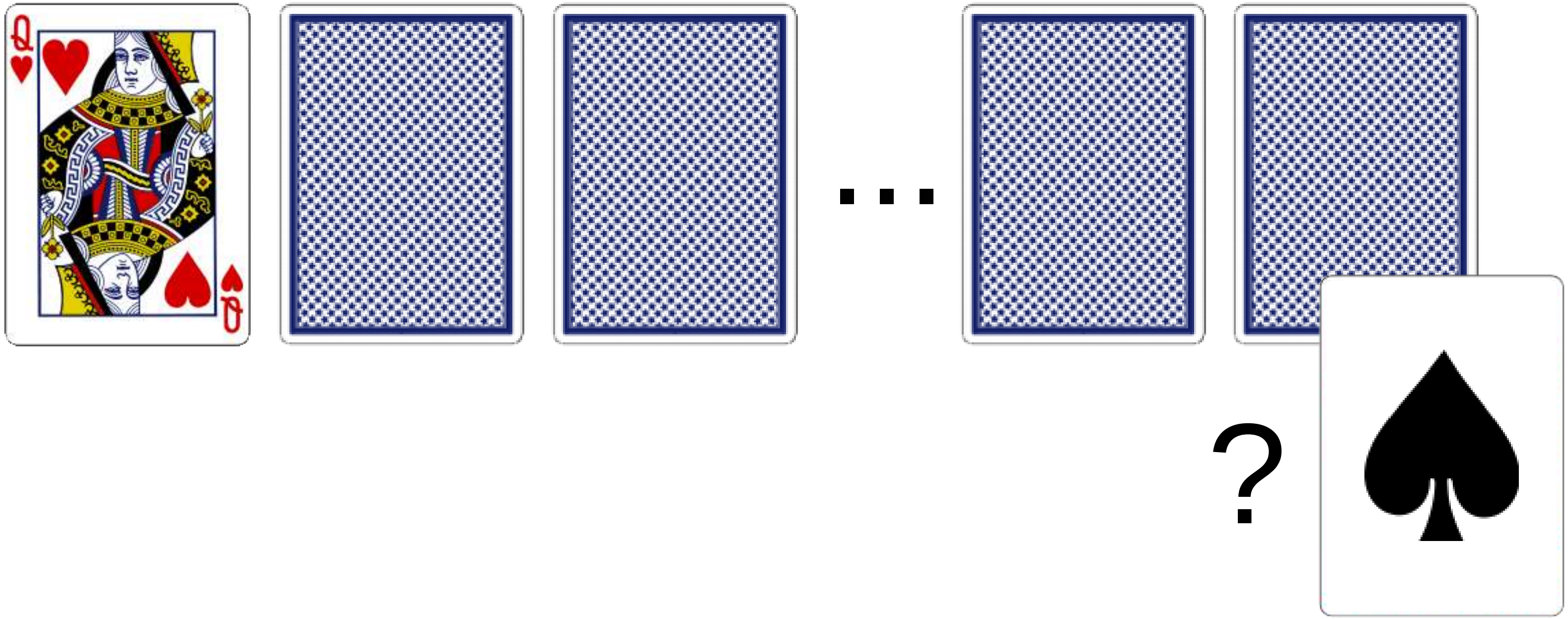
Is There Conditional Independence?



$$P(\text{Card52} \mid \text{Card1}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

$$? \stackrel{?}{=} ?$$

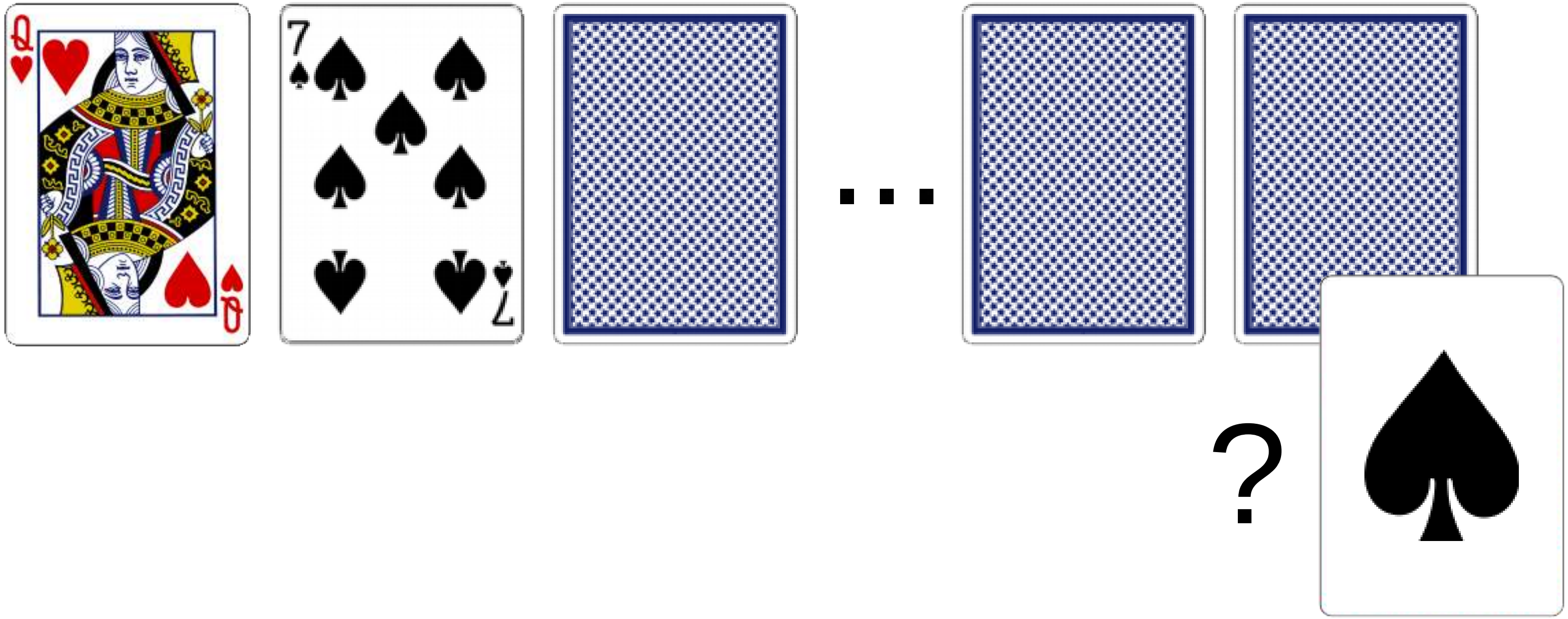
Is There Conditional Independence?



$$P(\text{Card52} \mid \text{Card1}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

$$13/51 \stackrel{?}{=} ?$$

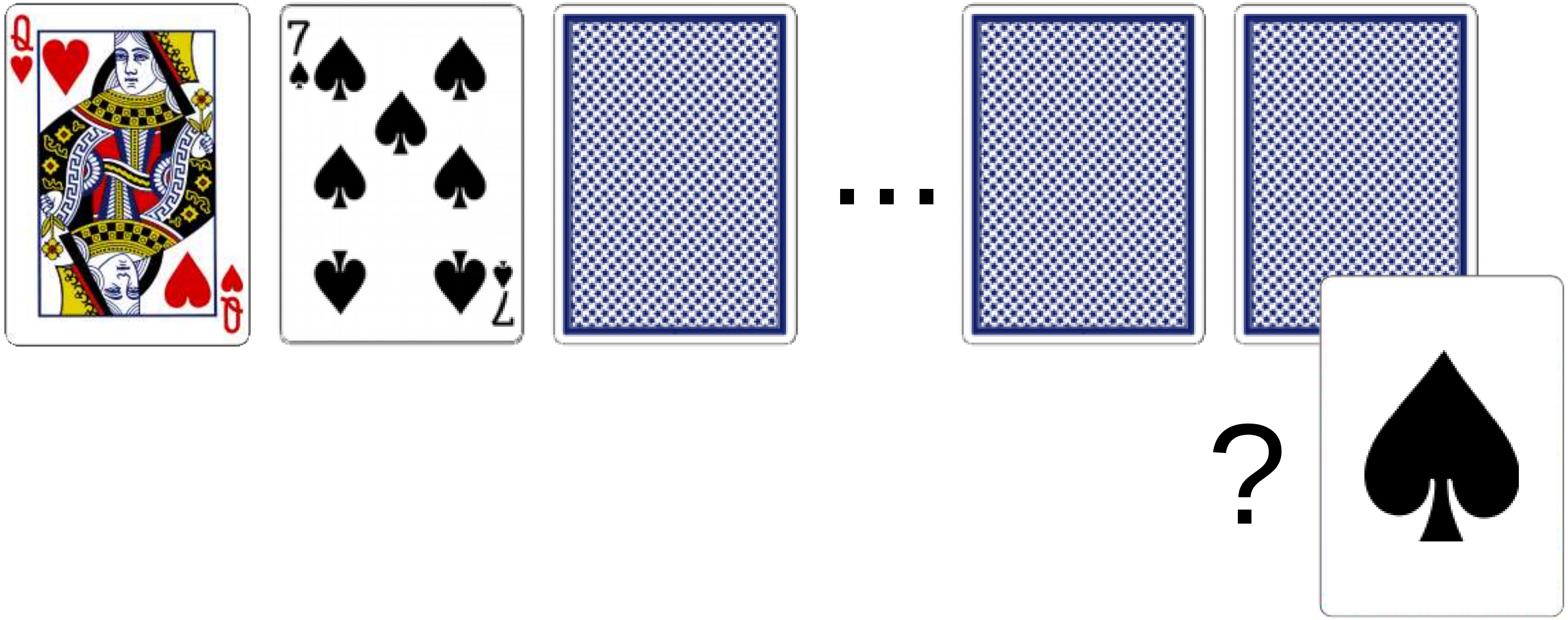
Is There Conditional Independence?



$$P(\text{Card52} \mid \text{Card1}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

$$13/51 \stackrel{?}{=} ?$$

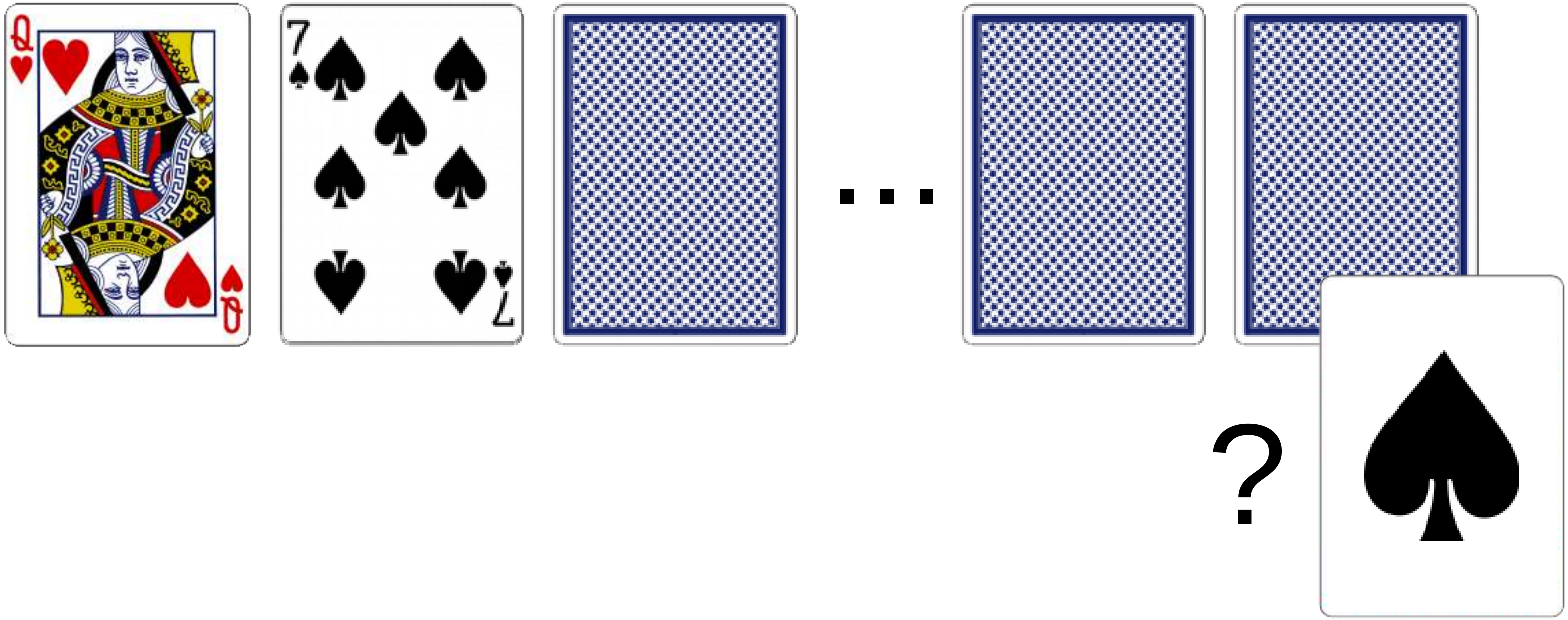
Is There Conditional Independence?



$$P(\text{Card52} \mid \text{Card1}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

$$13/51 \neq 12/50$$

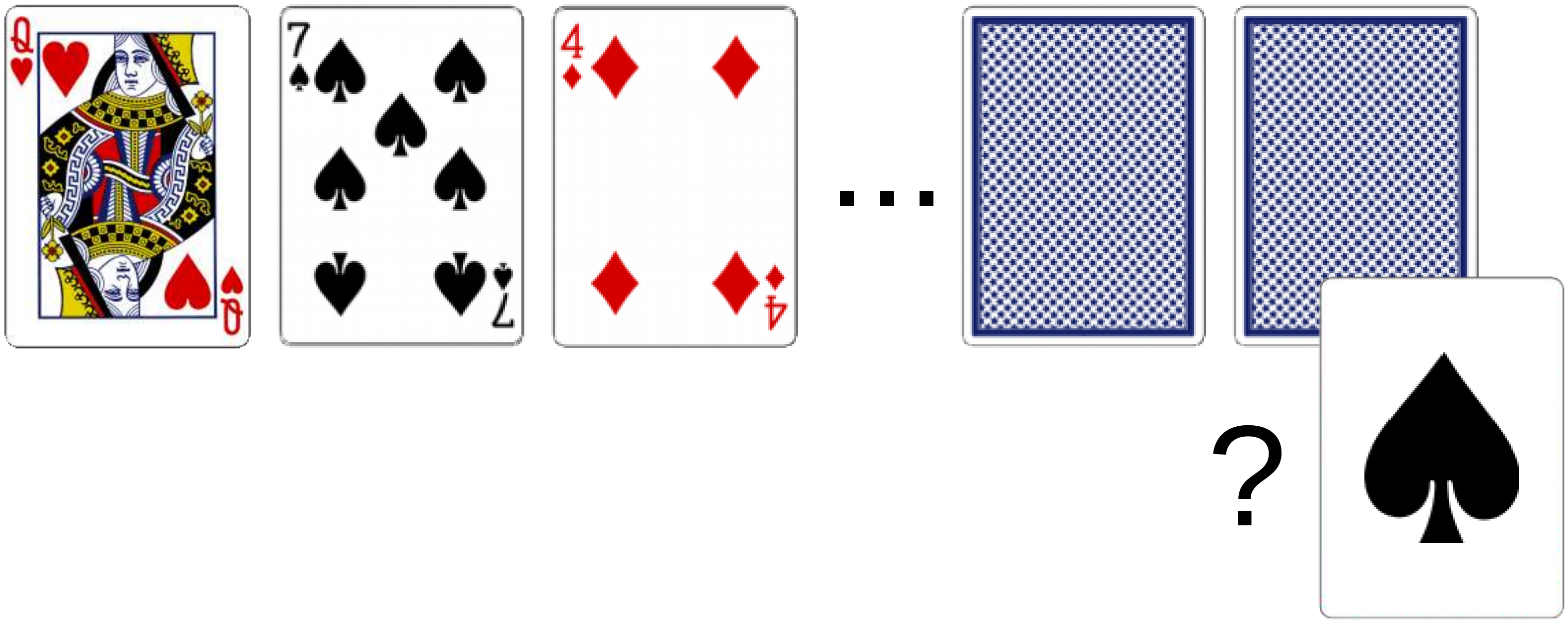
Is There Conditional Independence?



$P(\text{Card52} \mid \text{Card1}) \neq P(\text{Card52} \mid \text{Card1, Card2})$

$13/51 \neq 12/50$

Is There Conditional Independence?

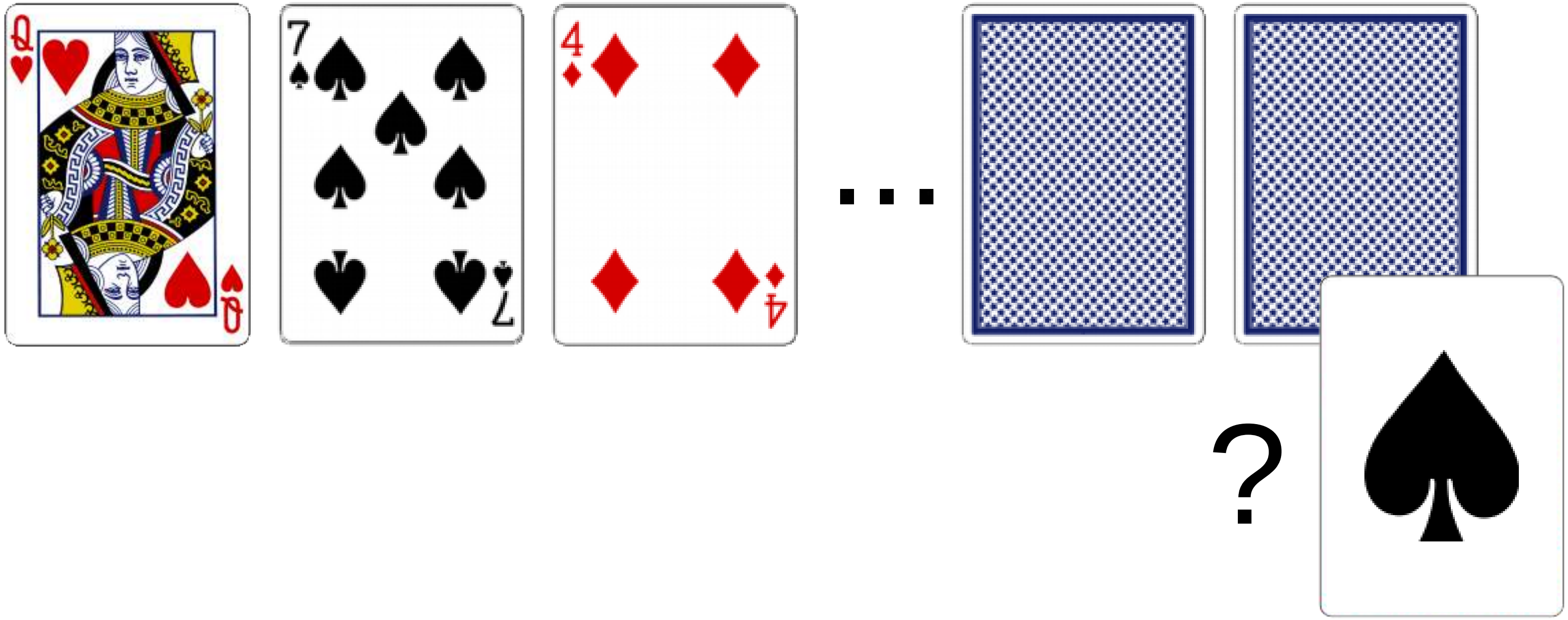


$$P(\text{Card52} \mid \text{Card1}) \neq P(\text{Card52} \mid \text{Card1}, \text{Card2})$$

$$13/51 \neq 12/50$$

$$P(\text{Card52} \mid \text{Card1}, \text{Card2}) \stackrel{?}{=} P(\text{Card52} \mid \text{Card1}, \text{Card2}, \text{Card3})$$

Is There Conditional Independence?



$P(\text{Card52} \mid \text{Card1}) \neq P(\text{Card52} \mid \text{Card1, Card2})$

$13/51 \neq 12/50$

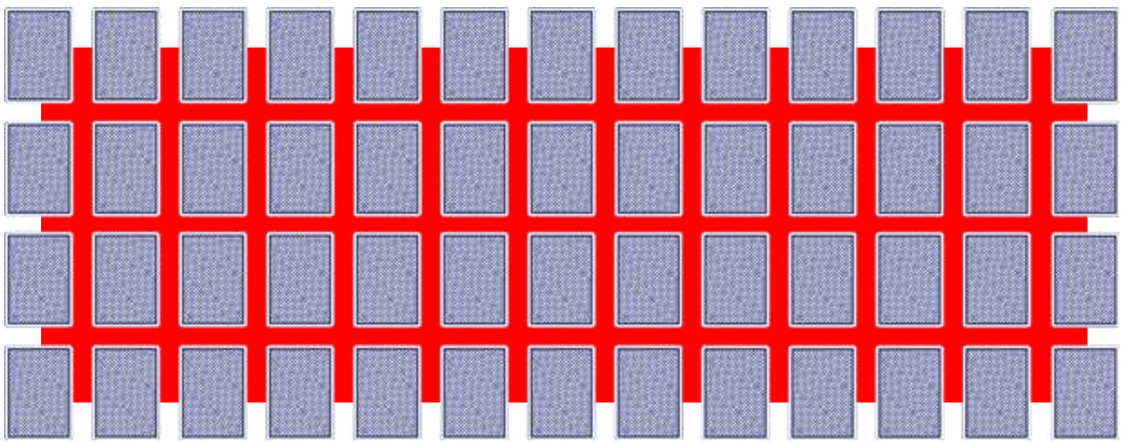
$P(\text{Card52} \mid \text{Card1, Card2}) \neq P(\text{Card52} \mid \text{Card1, Card2, Card3})$

$12/50 \neq 12/49$

Automated Reasoning

Let us automate this:

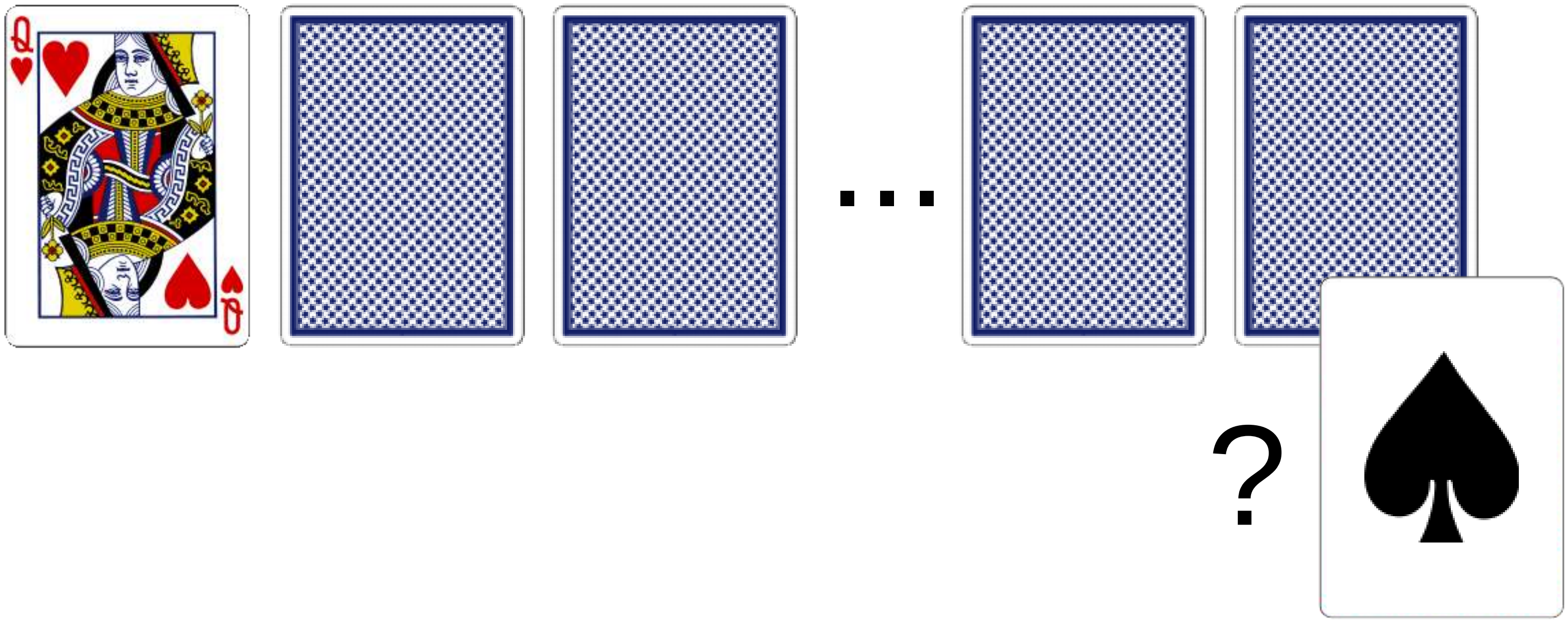
1. Probabilistic graphical model (e.g., factor graph) is fully connected!



(artist's impression)

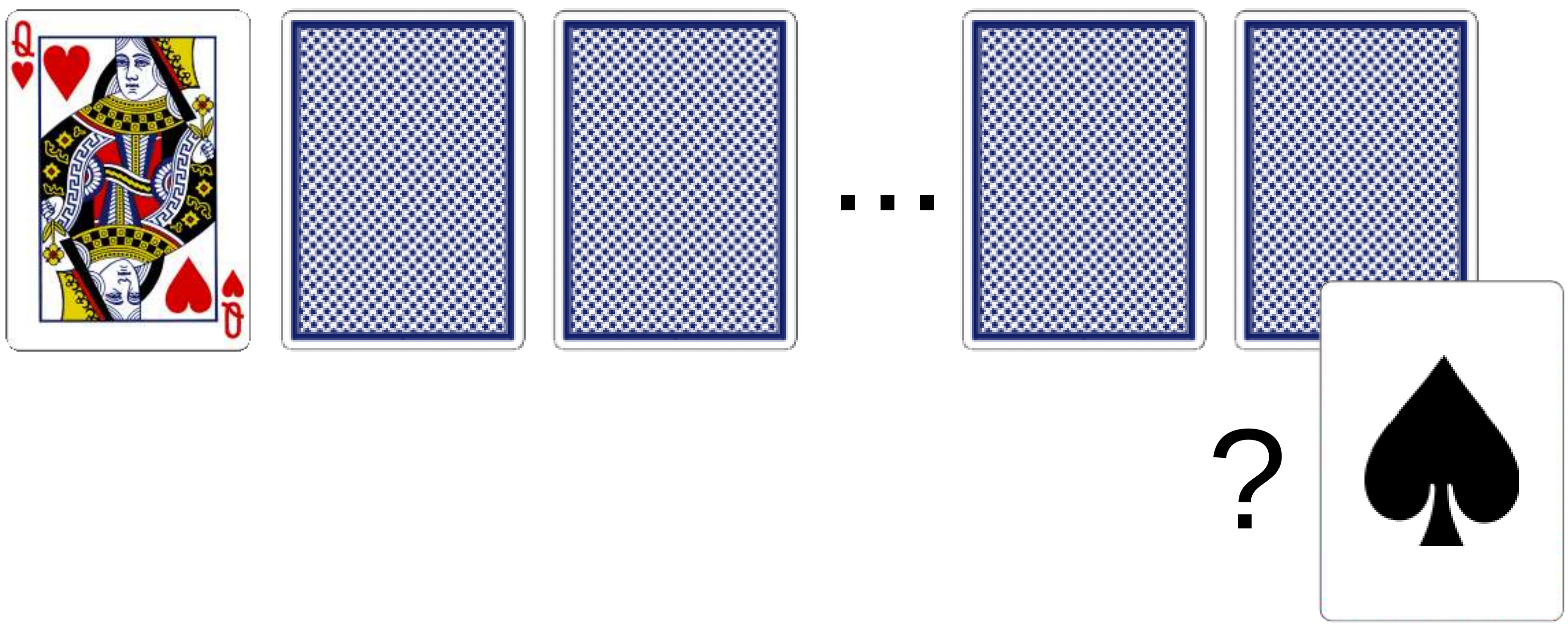
2. Probabilistic inference algorithm (e.g., variable elimination or junction tree) builds a table with 13^{52} rows

What's Going On Here?



Probability that Card52 is Spades given that Card1 is QH?

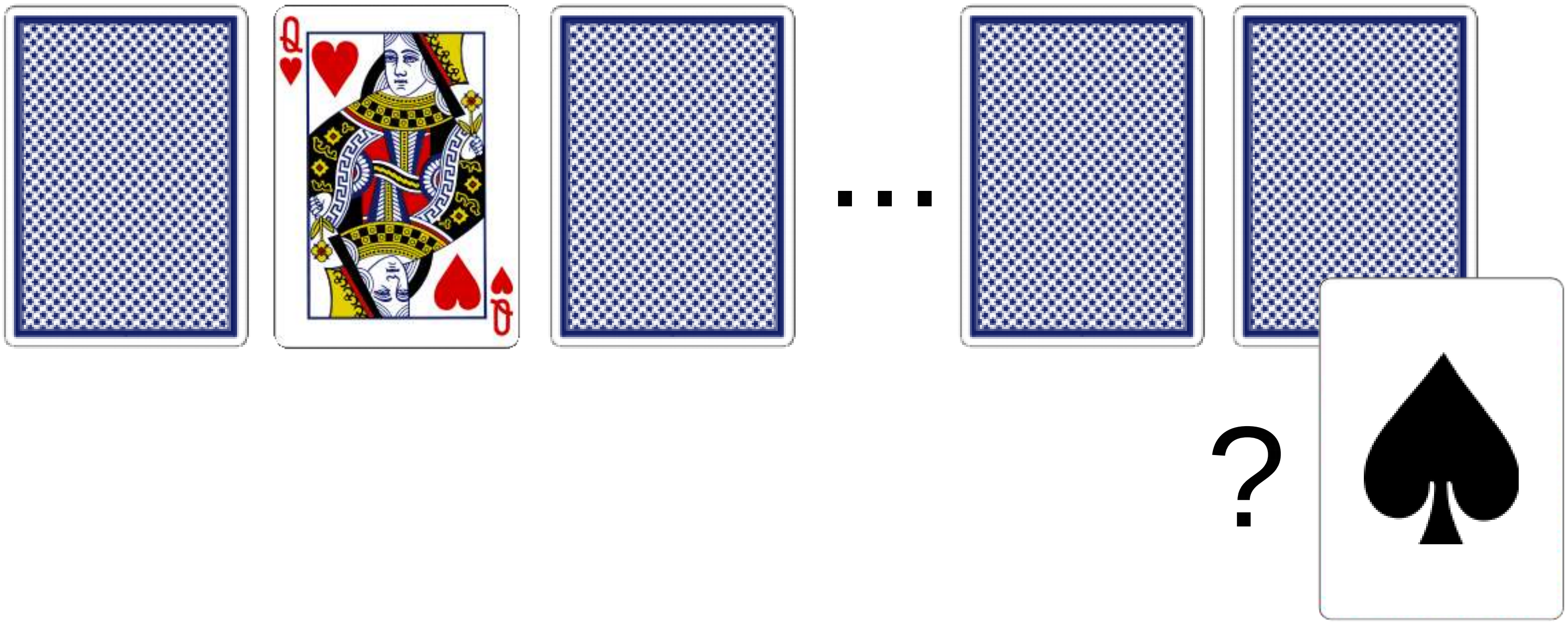
What's Going On Here?



Probability that Card52 is Spades given that Card1 is QH?

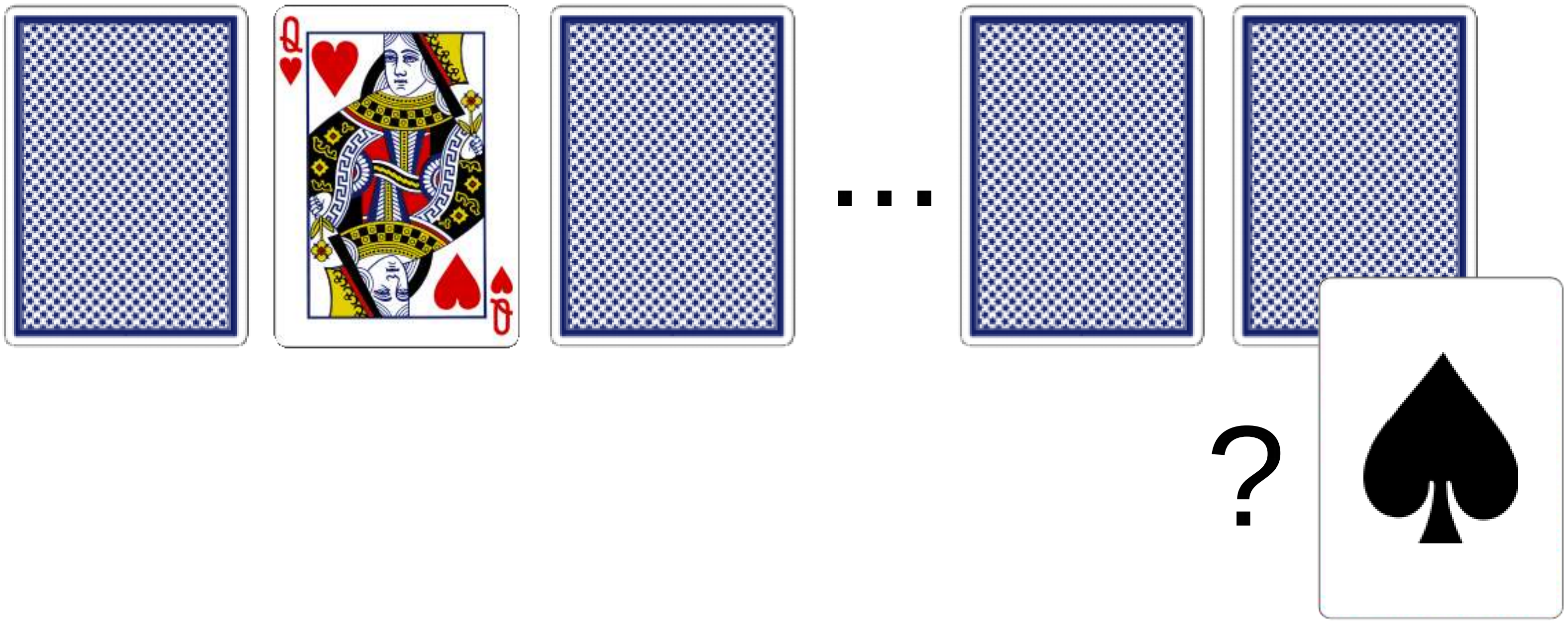
13/51

What's Going On Here?



Probability that Card52 is Spades given that Card2 is QH?

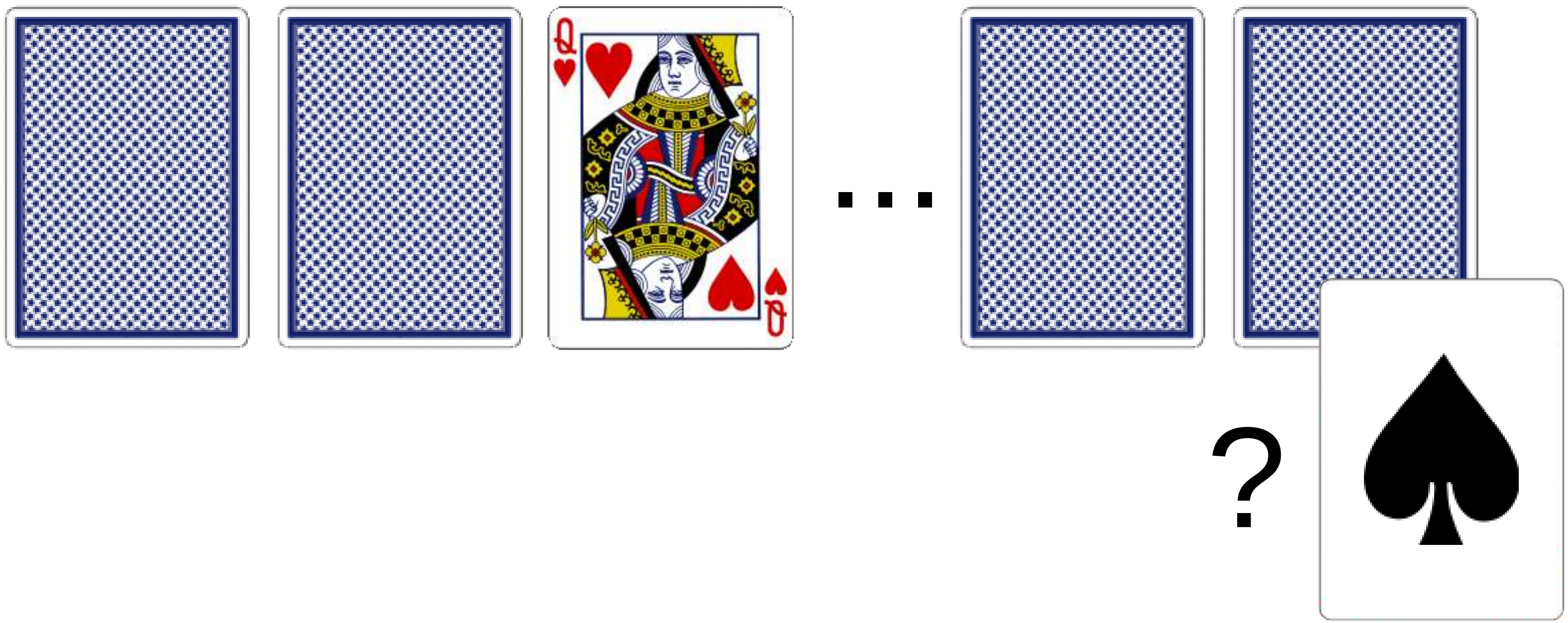
What's Going On Here?



Probability that Card52 is Spades given that Card2 is QH?

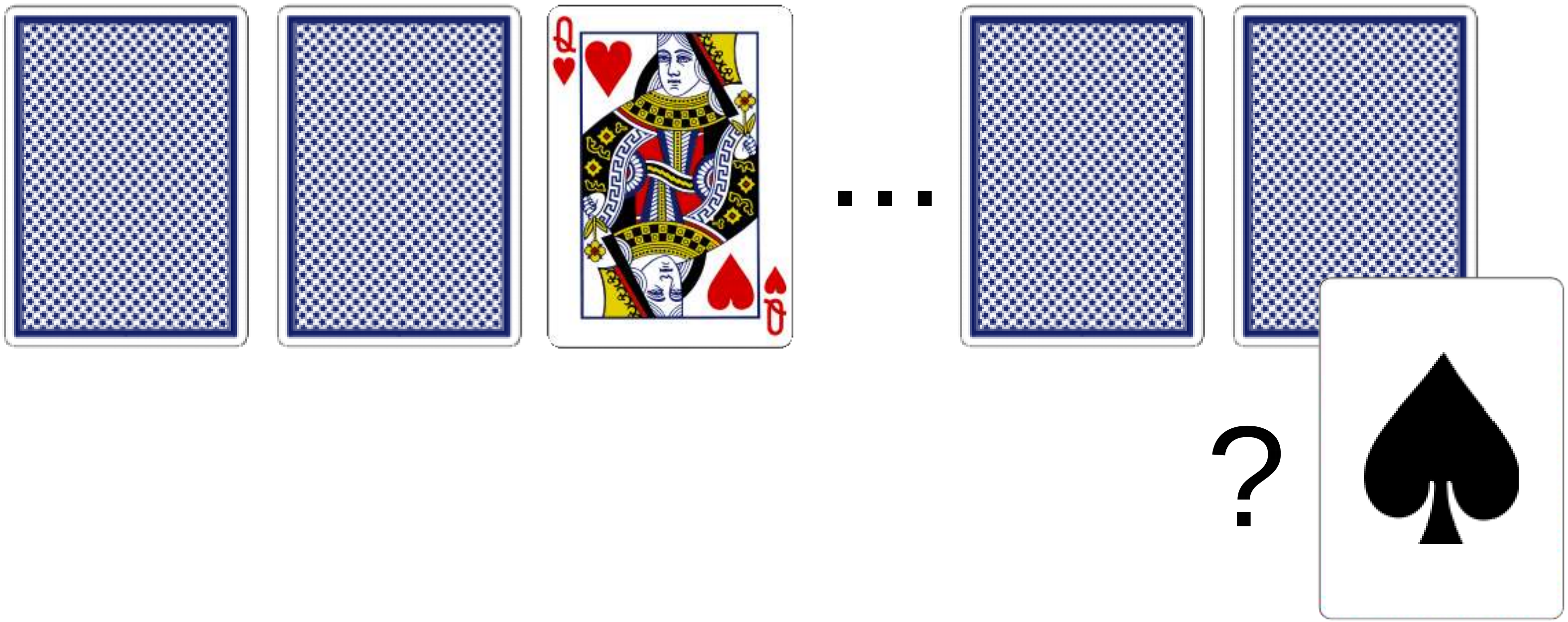
13/51

What's Going On Here?



Probability that Card52 is Spades given that Card3 is QH?

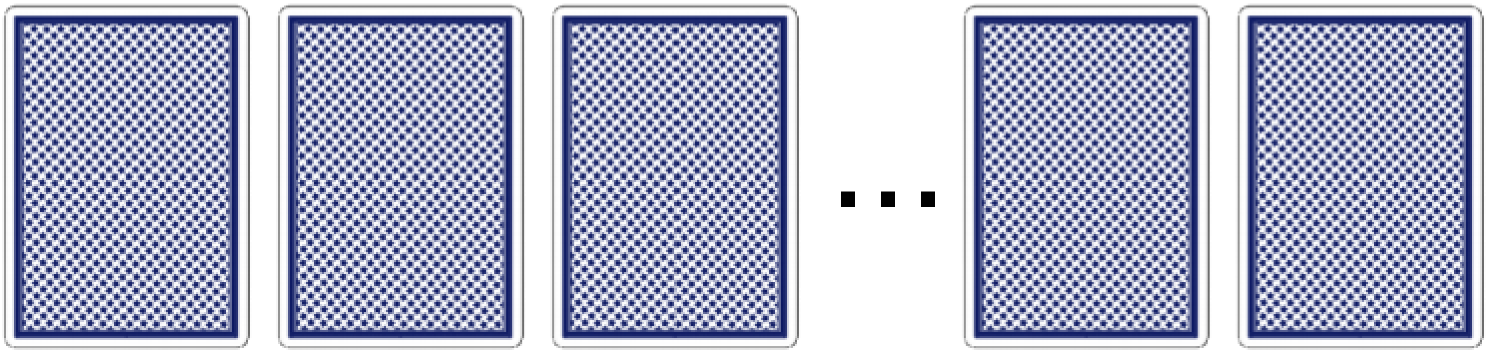
What's Going On Here?



Probability that Card52 is Spades given that Card3 is QH?

13/51

Tractable Probabilistic Inference

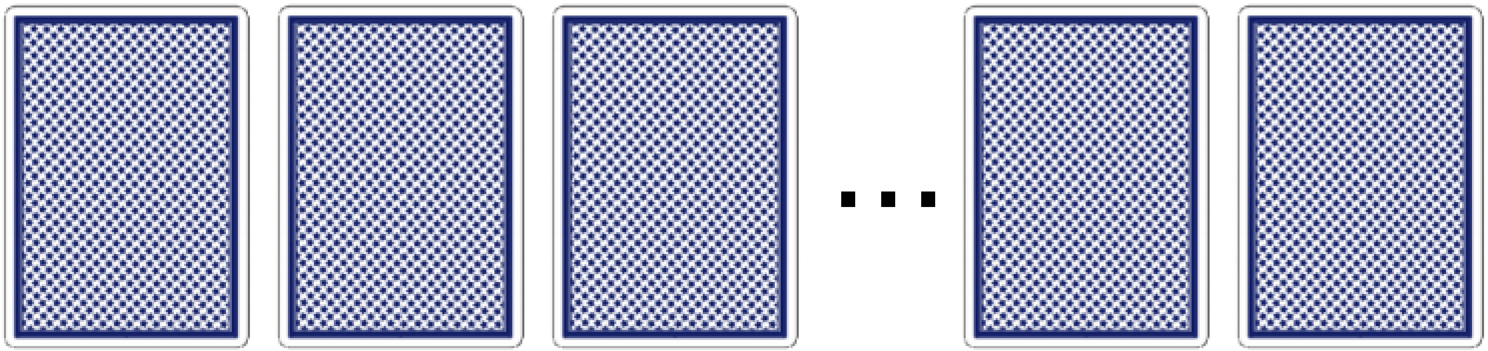


Which property makes inference tractable?

- Traditional belief: Independence (conditional/contextual)
- What's going on here?

[Niepert'14]

Tractable Probabilistic Inference



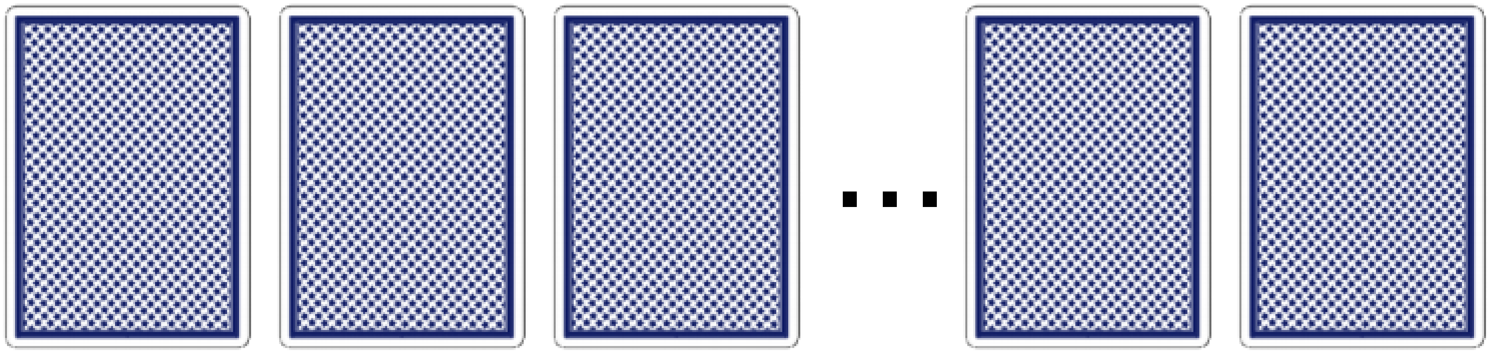
Which property makes inference tractable?

- Traditional belief: Independence (conditional/contextual)
- What's going on here?
  - High-level reasoning
  - Symmetry
  - Exchangeability

⇒ **Lifted Inference**

[Niepert'14]

Tractable Probabilistic Inference



Which property makes inference tractable?

- Traditional belief: Independence (conditional/contextual)
- What's going on here?
  - High-level reasoning
  - Symmetry
  - Exchangeability

⇒ **Lifted Inference**

See AAAI talk on Tuesday!

[Niepert'14]

Automated Reasoning

Let us automate this:

- **Relational** model

$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \land Card(p,c') \Rightarrow c = c'$$

- **Lifted** probabilistic inference algorithm

Other Examples of Lifted Inference

- First-order resolution

$$\forall x,\ \text{Human}(x) \Rightarrow \text{Mortal}(x)$$
$$\forall x,\ \text{Greek}(x) \Rightarrow \text{Human}(x)$$

implies

$$\forall x,\ \text{Greek}(x) \Rightarrow \text{Mortal}(x)$$

Other Examples of Lifted Inference

- First-order resolution
- Reasoning about populations

  We are investigating a rare disease. The disease is more rare in women, presenting only in **one in every two billion women** and **one in every billion men**. Then, assuming there are **3.4 billion men** and **3.6 billion women** in the world, the probability that **more than five people** have the disease is

$$1 - \sum_{n=0}^{5} \sum_{f=0}^{n} \binom{3.6 \cdot 10^9}{f} \left(1 - 0.5 \cdot 10^{-9}\right)^{3.6 \cdot 10^9 - f} \left(0.5 \cdot 10^{-9}\right)^{f}$$

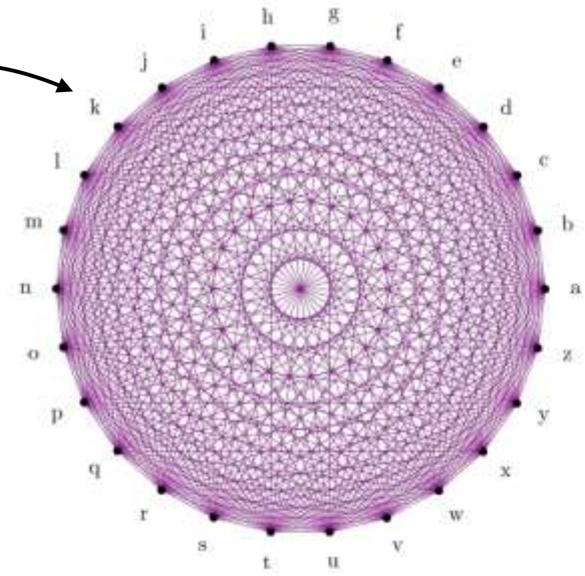
$$\times \binom{3.4 \cdot 10^9}{(n-f)} \left(1 - 10^{-9}\right)^{3.4 \cdot 10^9 - (n-f)} \left(10^{-9}\right)^{(n-f)}$$

Lifted Inference in SRL

- Statistical relational model (e.g., MLN)

3.14 FacultyPage(x) ∧ Linked(x,y) ⇒ CoursePage(y)

- As a probabilistic graphical model:
  - 26 pages; 728 variables; 676 factors
  - 1000 pages; 1,002,000 variables; 1,000,000 factors



- Highly intractable?
  - **Lifted inference** in milliseconds!

Summary of Motivation

- Relational data is everywhere:
  - Databases in industry
  - Databases in sciences
  - Knowledge bases
- Lifted inference:
  - Use relational structure during reasoning
  - Very efficient where traditional methods break

This tutorial: Lifted Inference in Relational Models

Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

What Everyone Should Know about Databases

- Database = several relations (a.k.a. tables)
- SQL Query = FO Formula
- Boolean Query = FO Sentence

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

Query: First Order Formula

$Q(z) = \exists x\,(\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

Find friends of smokers in 2009

Query answer: $Q(D)$ =

Z
Bob
Carol

Conjunctive Queries CQ = FO(∃, ∧)
Union of Conjunctive Queries UCQ = FO(∃, ∧, ∨)

What Everyone Should Know about Databases

Database: relations (= tables)

D =

Smoker

X	Y
Alice	2009
Alice	2010
Bob	2009
Carol	2010

Friend

X	Z
Alice	Bob
Alice	Carol
Bob	Carol
Carol	Bob

Query: First Order Formula

$Q(z) = \exists x\, (\text{Smoker}(x, \text{'2009'}) \wedge \text{Friend}(x,z))$

Find friends of smokers in 2009

Query answer: $Q(D)$ =

Z
Bob
Carol

Conjunctive Queries CQ = FO(∃, ∧)

Union of Conjunctive Queries UCQ = FO(∃, ∧, ∨)

Boolean Query: FO Sentence

$Q = \exists x\, (\text{Smoker}(x, \text{'2009'}) \wedge \text{Friend}(x, \text{'Bob'}))$

Query answer: $Q(D)$ = TRUE

What Everyone Should Know about Databases

Declarative Query	→	Query Plan
"what"	→	*"how"*

What Everyone Should Know about Databases

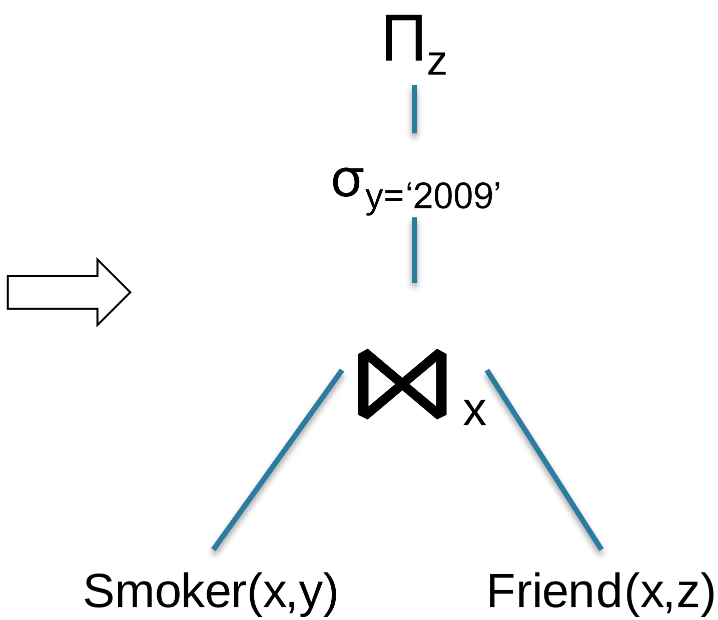
Declarative Query → Query Plan
"what" → *"how"*

$Q(z) = \exists x\,(\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

What Everyone Should Know about Databases

Declarative Query	→	Query Plan
"what"	→	*"how"*

$Q(z) = \exists x\ (\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$

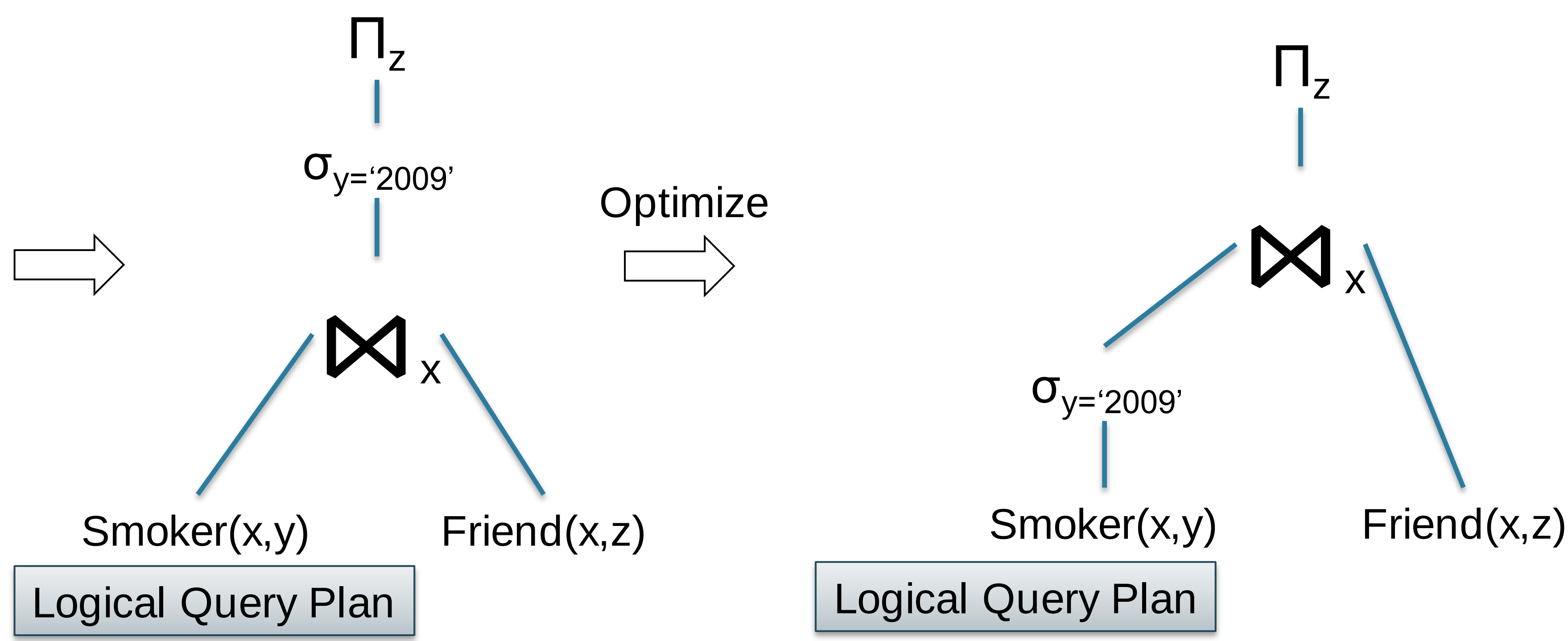


Logical Query Plan

What Everyone Should Know about Databases

Declarative Query → Query Plan
"what" → *"how"*

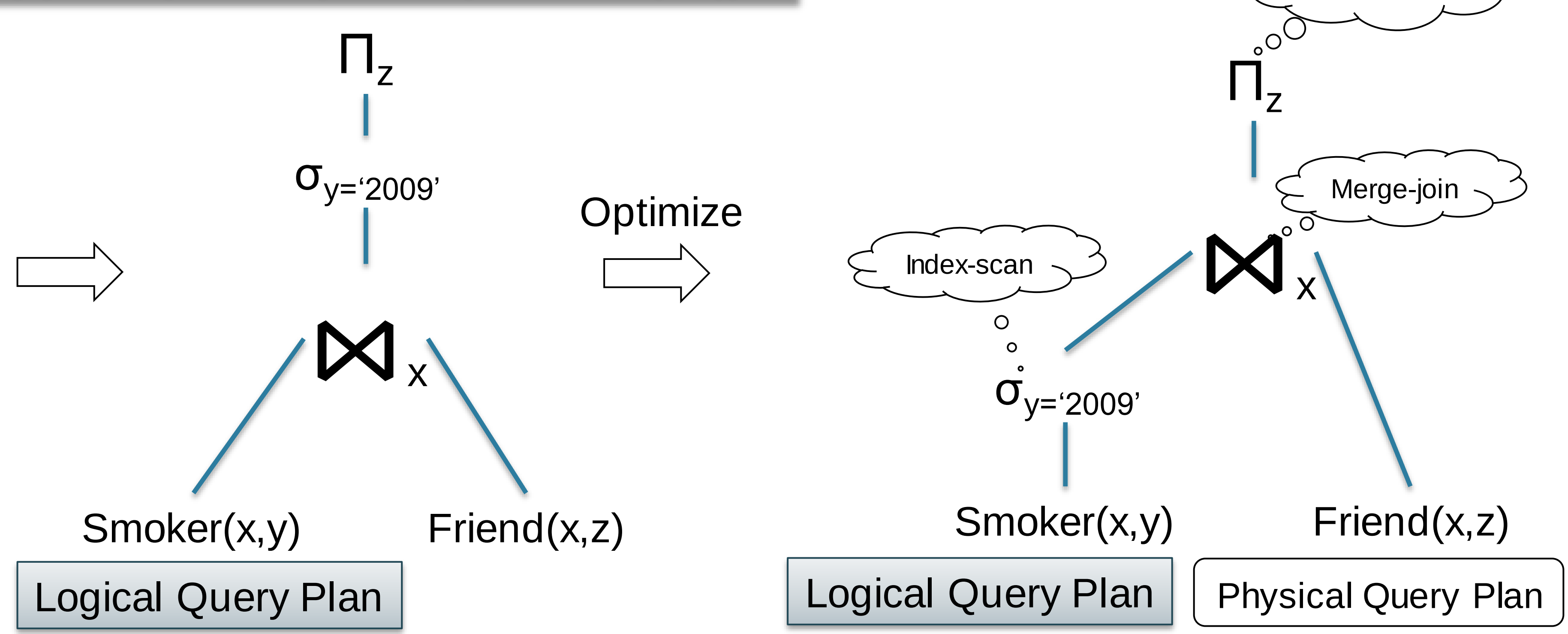
$Q(z) = \exists x\,(\mathrm{Smoker}(x,\text{'2009'}) \land \mathrm{Friend}(x,z))$



What Everyone Should Know about Databases

Declarative Query → Query Plan
"what" → *"how"*

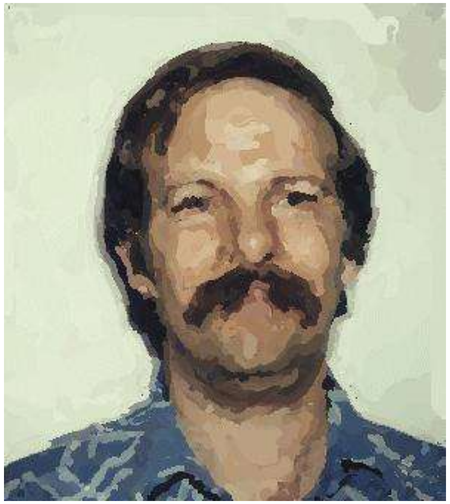
$Q(z) = \exists x\ (\text{Smoker}(x,\text{'2009'}) \wedge \text{Friend}(x,z))$



What Every Researcher Should Know about Databases

Problem: compute Q(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award



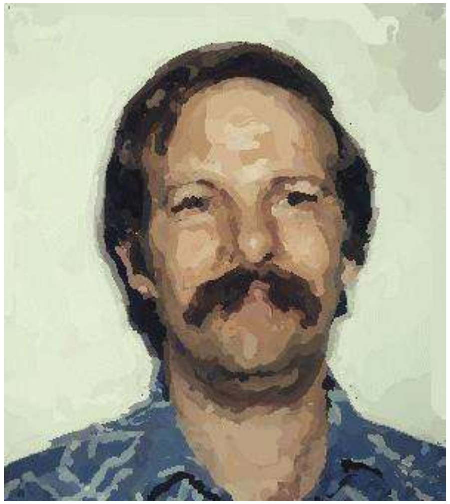
[Vardi'82]

What Every Researcher Should Know about Databases

Problem: compute Q(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award

- Data complexity:
  fix Q, complexity = f(D)

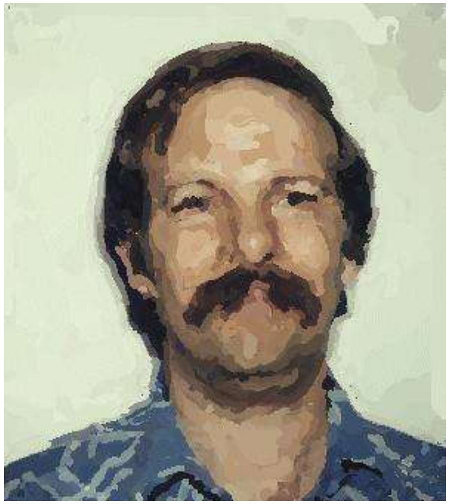


[Vardi'82]

What Every Researcher Should Know about Databases

Problem: compute Q(D)

Moshe Vardi [Vardi'82]
2008 ACM SIGMOD Contribution Award



- Data complexity:
  fix Q, complexity = f(D)

  Query complexity: (expression complexity)
  fix D, complexity = f(Q)

- Combined complexity:
  complexity = f(D,Q)

[Vardi'82]

Probabilistic Databases

- A probabilistic database = relational database where each tuple has an associated probability
- Semantics = probability distribution over possible worlds (deterministic databases)
- In this talk: tuples are independent events

[Suciu'11]

Example

Probabilistic database D:

Friend

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

Example

Probabilistic database D:

Friend

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

Possible worlds semantics:

$p_1p_2p_3$

x	y
A	B
A	C
B	C

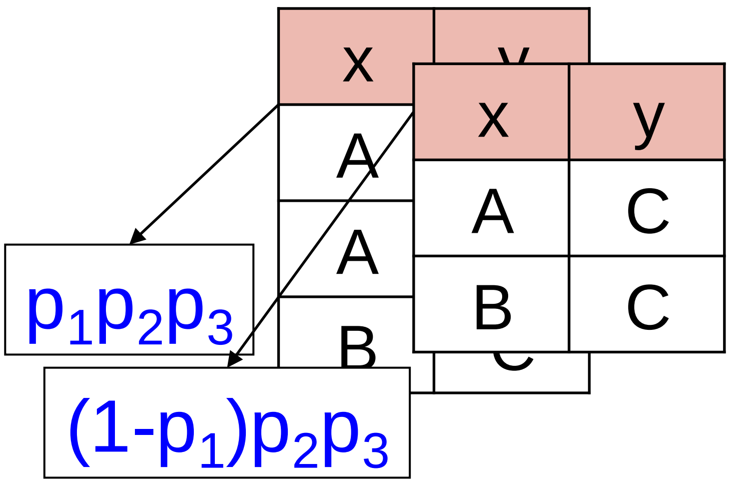
Example

Probabilistic database D:

Friend

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

Possible worlds semantics:



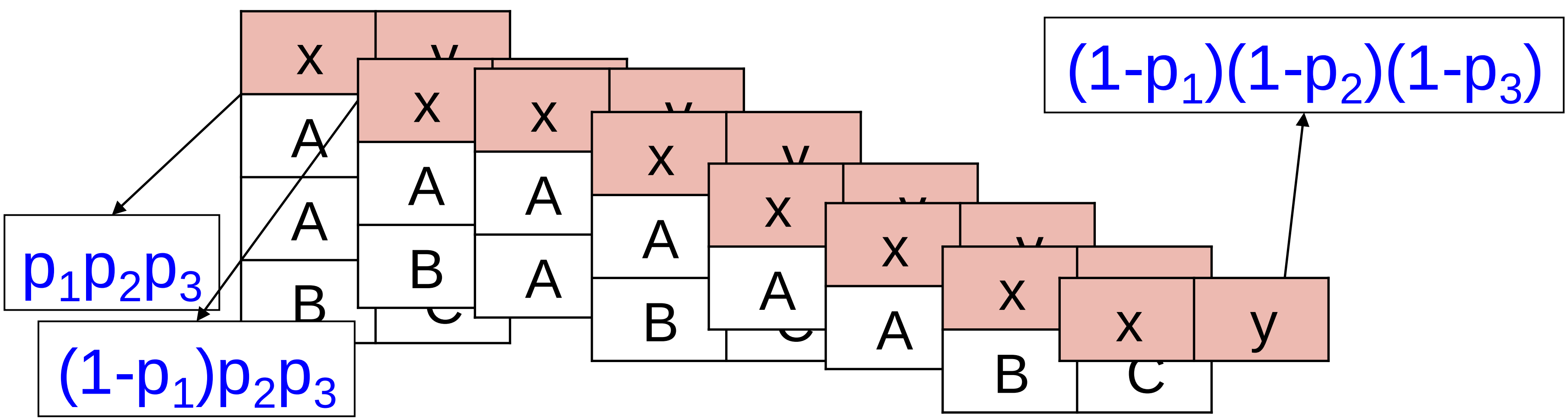
Example

Probabilistic database D:

Friend

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

Possible worlds semantics:



Query Semantics

Fix a Boolean query Q

Fix a probabilistic database D:

P (Q | D) = marginal probability of Q
on possible words of D

$Q = \exists x \exists y\ \mathrm{Smoker}(x) \wedge \mathrm{Friend}(x,y)$

An Example

$P(Q \mid D) =$

Friend

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

Smoker

x	P
A	p_1
B	p_2
C	p_3

$Q = \exists x \exists y\ Smoker(x) \land Friend(x,y)$

An Example

$P(Q \mid D) = 1-(1-q_1)*(1-q_2)$

Friend

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

Smoker

x	P
A	p_1
B	p_2
C	p_3

$Q = \exists x \exists y\ \text{Smoker}(x) \wedge \text{Friend}(x,y)$

An Example

$$P(Q \mid D) = \quad p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$$

Friend

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

Smoker

x	P
A	p_1
B	p_2
C	p_3

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

$$P(Q \mid D) = p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$$
$$1-(1-q_3)*(1-q_4)*(1-q_5)$$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

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

$Q = \exists x \exists y\ \mathrm{Smoker}(x) \land \mathrm{Friend}(x,y)$

An Example

$$P(Q \mid D) = p_1 * [\ 1-(1-q_1)*(1-q_2)\ ]$$
$$p_2 * [\ 1-(1-q_3)*(1-q_4)*(1-q_5)\ ]$$

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

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

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

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

Smoker

x	P
A	p_1
B	p_2
C	p_3

Friend

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

$Q = \exists x \exists y\ Smoker(x) \wedge Friend(x,y)$

An Example

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

One can compute $P(Q \mid D)$ in PTIME in the size of the database D

Friend

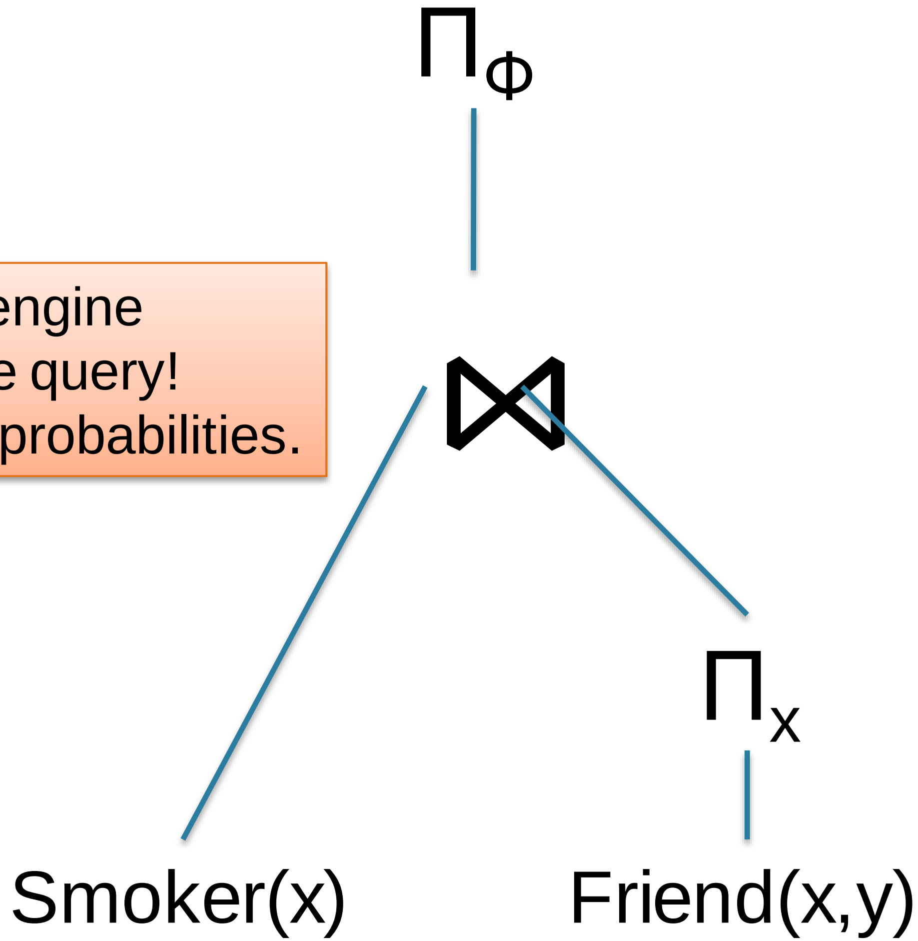
x	y	P
A	D	q_1
A	E	q_2
B	F	q_3
B	G	q_4
B	H	q_5

Smoker

x	P
A	p_1
B	p_2
C	p_3

$Q = \exists x \exists y\ \mathrm{Smoker}(x) \wedge \mathrm{Friend}(x,y)$

An Example



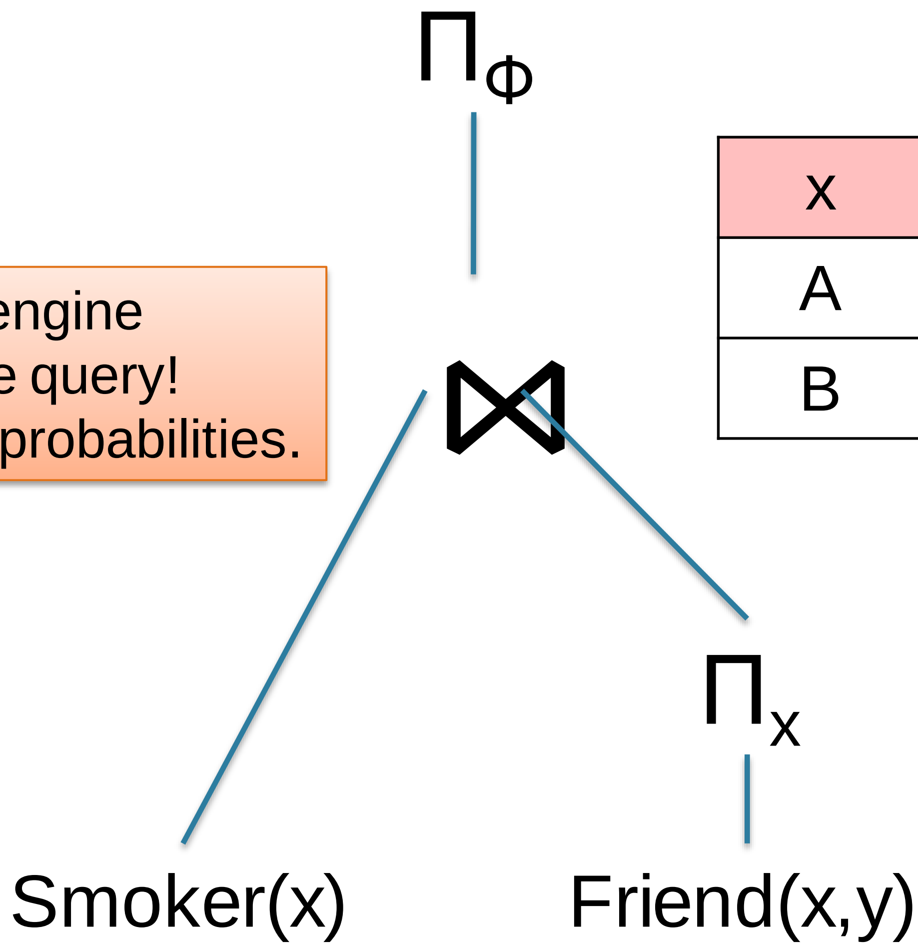
Use the SQL engine
to compute the query!
Aggregate on probabilities.

x	P
A	p_1
B	p_2
C	p_3

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

An Example

$Q = \exists x \exists y \; Smoker(x) \wedge Friend(x,y)$



Use the SQL engine
to compute the query!
Aggregate on probabilities.

x	P
A	$1-(1-q_1)(1-q_2)$
B	$1-(1-q_4)(1-q_5)\,(1-q_6)$

x	P
A	p_1
B	p_2
C	p_3

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

$Q = \exists x \exists y \ \text{Smoker}(x) \wedge \text{Friend}(x,y)$

An Example

$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}*$
$\{1-p_2[1-(1-q_4)(1-q_5)\ (1-q_6)]\}$

Π_Φ

x	P
A	$1-(1-q_1)(1-q_2)$
B	$1-(1-q_4)(1-q_5)\ (1-q_6)$

Use the SQL engine
to compute the query!
Aggregate on probabilities.

$\bowtie$

Π_x

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

x	P
A	p_1
B	p_2
C	p_3

Smoker(x)

Friend(x,y)

Problem Statement

Given: probabilistic database D, query Q

Compute: P(Q | D)

Data complexity: fix Q, complexity = f(|D|)

Approaches to Compute $P(Q \mid D)$

- Propositional inference:
  - Ground the query $Q \rightarrow F_{Q,D}$, compute $P(F_{Q,D})$
  - This is Weighted Model Counting (later…)
  - Works for every query Q
  - But: may be exponential in $|D|$ (data complexity)

- Lifted inference:
  - Compute a query plan for Q, execute plan on D
  - Always polynomial time in $|D|$ (data complexity)
  - But: does not work for all queries Q

[Olteanu'08, Jha'13, Dalvi'04, Dalvi'12]

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  ∀-Rule: $P(\forall z\ Q) = \prod_{C \in Domain} P(Q[C/z])$
  ∃-Rule: $P(\exists z\ Q) = 1 - \prod_{C \in Domain} (1 - P(Q[C/z]))$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  ∀-Rule: $P(\forall z\, Q) = \prod_{C \in Domain} P(Q[C/z])$
  ∃-Rule: $P(\exists z\, Q) = 1 - \prod_{C \in Domain} (1 - P(Q[C/z]))$

- Inclusion/Exclusion formula:
  $P(Q_1 \vee Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \wedge Q_2)$
  $P(Q_1 \wedge Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$

The Lifted Inference Rules

- If Q_1, Q_2 are independent:
  AND-rule: $P(Q_1 \wedge Q_2) = P(Q_1)P(Q_2)$
  OR-rule: $P(Q_1 \vee Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

- If $Q[C_1/x]$, $Q[C_2/x]$, … are independent
  ∀-Rule: $P(\forall z\ Q) = \Pi_{C\ \in Domain}\ P(Q[C/z])$
  ∃-Rule: $P(\exists z\ Q) = 1 - \Pi_{C\ \in Domain}\ (1 - P(Q[C/z])$

- Inclusion/Exclusion formula:
  $P(Q_1 \vee Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \wedge Q_2)$
  $P(Q_1 \wedge Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$

- Negation: $P(\neg Q) = 1 - P(Q)$

Example

$\boxed{Q = \forall x \forall y\, (\mathrm{Smoker}(x) \vee \mathrm{Friend}(x,y))}$ $= \forall x\, (\mathrm{Smoker}(x) \vee \forall y\, \mathrm{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \mathrm{Domain}} P(\mathrm{Smoker}(A) \vee \forall y\, \mathrm{Friend}(A,y))$

Check independence:

$\mathrm{Smoker}(\mathrm{Alice}) \vee \forall y\, \mathrm{Friend}(\mathrm{Alice},y)$

$\mathrm{Smoker}(\mathrm{Bob}) \vee \forall y\, \mathrm{Friend}(\mathrm{Bob},y)$

Example

$\boxed{Q = \forall x \forall y\, (\mathrm{Smoker}(x) \lor \mathrm{Friend}(x,y))}$ $= \forall x\, (\mathrm{Smoker}(x) \lor \forall y\, \mathrm{Friend}(x,y))$

∀-Rule

$P(Q) = \prod_{A \in \mathrm{Domain}} P(\mathrm{Smoker}(A) \lor \forall y\, \mathrm{Friend}(A,y))$

Check independence:
$\mathrm{Smoker}(\mathrm{Alice}) \lor \forall y\, \mathrm{Friend}(\mathrm{Alice},y)$
$\mathrm{Smoker}(\mathrm{Bob}) \lor \forall y\, \mathrm{Friend}(\mathrm{Bob},y)$

$P(Q) = \prod_{A \in \mathrm{Domain}} (1 - P(\mathrm{Smoker}(A))) \times (1 - P(\forall y\, \mathrm{Friend}(A,y)))$

∨-Rule

Example

$Q = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y))$

$= \forall x\, (\text{Smoker}(x) \vee \forall y\, \text{Friend}(x,y))$

∀-Rule

$P(Q) = \Pi_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\, \text{Friend}(A,y))$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$P(Q) = \Pi_{A \in \text{Domain}} (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\, \text{Friend}(A,y)))$

∨-Rule

$P(Q) = \Pi_{A \in \text{Domain}} (1 - P(\text{Smoker}(A))) \times (1 - \Pi_{B \in \text{Domain}} P(\text{Friend}(A,B)))$

∀-Rule

Example

$\boxed{Q = \forall x \forall y\, (\text{Smoker}(x) \vee \text{Friend}(x,y))}$ $= \forall x\, (\text{Smoker}(x) \vee \forall y\, \text{Friend}(x,y))$

∀-Rule

$$P(Q) = \prod_{A \in \text{Domain}} P(\text{Smoker}(A) \vee \forall y\, \text{Friend}(A,y))$$

Check independence:
Smoker(Alice) ∨ ∀y Friend(Alice,y)
Smoker(Bob) ∨ ∀y Friend(Bob,y)

$$P(Q) = \prod_{A \in \text{Domain}} (1 - P(\text{Smoker}(A))) \times (1 - P(\forall y\, \text{Friend}(A,y)))$$

∨-Rule

$$P(Q) = \prod_{A \in \text{Domain}} (1 - P(\text{Smoker}(A))) \times (1 - \prod_{B \in \text{Domain}} P(\text{Friend}(A,B)))$$

Lookup the probabilities in the database

∀-Rule

Example

$Q = \forall x \forall y\, (Smoker(x) \vee Friend(x,y))$ $= \forall x\, (Smoker(x) \vee \forall y\, Friend(x,y))$

∀-Rule

$P(Q) = \prod_{A \in Domain} P(Smoker(A) \vee \forall y\, Friend(A,y))$

Check independence:
$Smoker(Alice) \vee \forall y\, Friend(Alice,y)$
$Smoker(Bob) \vee \forall y\, Friend(Bob,y)$

$P(Q) = \prod_{A \in Domain} (1 - P(Smoker(A))) \times (1 - P(\forall y\, Friend(A,y)))$

∨-Rule

$P(Q) = \prod_{A \in Domain} (1 - P(Smoker(A))) \times (1 - \prod_{B \in Domain} P(Friend(A,B)))$

Lookup the probabilities in the database

∀-Rule

Runtime = $O(n^2)$.

Discussion: CNF vs. DNF

Databases		KR/AI	
Conjunctive Queries CQ	FO(∃, ∧)	Positive Clause	FO(∀, ∨)
Union of Conjunctive Queries UCQ	FO(∃, ∧, ∨) = ∃ Positive-DNF	Positive FO	FO(∀, ∧, ∨) = ∀ Positive-CNF
UCQ with "safe negation" UCQ¬	∃ DNF	First Order CNF	∀ CNF

$Q = \exists x, \exists y,\ Smoker(x) \wedge Friend(x,y)$

$Q = \forall x \forall y\ (Smoker(x) \vee Friend(x,y))$

By duality we can reduce one problem to the other:

$$\exists x, \exists y,\ Smoker(x) \wedge Friend(x,y) = \neg \forall x, \forall y,\ (\neg Smoker(x) \vee \neg Friend(x,y))$$

Discussion

Lifted Inference Sometimes Fails

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

No rule applies here!

The ∀-rule does not apply, because $H_0[\mathrm{Alice}/x]$ and $H_0[\mathrm{Bob}/x]$ are dependent:

$$H_0[\mathrm{Alice}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Alice}) \lor \mathrm{Friend}(\mathrm{Alice},y) \lor \mathrm{Jogger}(y))$$
$$H_0[\mathrm{Bob}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Bob}) \lor \mathrm{Friend}(\mathrm{Bob},y) \lor \mathrm{Jogger}(y))$$

Dependent

Discussion

Lifted Inference Sometimes Fails

$H_0 = \forall x \forall y\, (\mathrm{Smoker}(x) \vee \mathrm{Friend}(x,y) \vee \mathrm{Jogger}(y))$

No rule applies here!

The ∀-rule does not apply, because $H_0[\mathrm{Alice}/x]$ and $H_0[\mathrm{Bob}/x]$ are dependent:

$$H_0[\mathrm{Alice}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Alice}) \vee \mathrm{Friend}(\mathrm{Alice},y) \vee \mathrm{Jogger}(y))$$
$$H_0[\mathrm{Bob}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Bob}) \vee \mathrm{Friend}(\mathrm{Bob},y) \vee \mathrm{Jogger}(y))$$

Dependent

Theorem. [Dalvi'04] Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

Proof: later…

Discussion

Lifted Inference Sometimes Fails

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

No rule applies here!

The ∀-rule does not apply, because $H_0[\mathrm{Alice}/x]$ and $H_0[\mathrm{Bob}/x]$ are dependent:

$$H_0[\mathrm{Alice}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Alice}) \lor \mathrm{Friend}(\mathrm{Alice},y) \lor \mathrm{Jogger}(y))$$
$$H_0[\mathrm{Bob}/x] = \forall y\, (\mathrm{Smoker}(\mathrm{Bob}) \lor \mathrm{Friend}(\mathrm{Bob},y) \lor \mathrm{Jogger}(y))$$

Dependent

Theorem. [Dalvi'04] Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

Proof: later…

Consequence: assuming PTIME ≠ #P, H_0 is not liftable!

Summary

- Database D = relations
- Query Q = FO
- Query plans, query optimization
- Data complexity: fix Q, complexity f(D)
- Probabilistic DB's = independent tuples
- Lifted inference: simple, but fails sometimes

Next: Weighted Model Counting = Unified framework for inference

Later: Are rules complete? Yes! (sort of): Power of Lifted Inference

Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

Weighted Model Counting

- Model = solution to a propositional logic formula Δ
- Model counting = #SAT

Δ = (Rain ⇒ Cloudy)

Rain	Cloudy	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

+

#SAT = 3

Weighted Model Counting

- Model = solution to a propositional logic formula Δ
- Model counting = #SAT
- Weighted model counting (WMC)
  - Weights for assignments to variables
  - Model weight is product of variable weights w(.)

Δ = (Rain ⇒ Cloudy)

Rain

w(R)	w(¬R)
1	2

Cloudy

w(C)	w(¬C)
3	5

Rain	Cloudy	Model?	Weight
T	T	Yes	1 * 3 = 3
T	F	No	0
F	T	Yes	2 * 3 = 6
F	F	Yes	2 * 5 = 10

+

#SAT = 3

Weighted Model Counting

- Model = solution to a propositional logic formula Δ
- Model counting = #SAT
- Weighted model counting (WMC)
  - Weights for assignments to variables
  - Model weight is product of variable weights w(.)

Δ = (Rain ⇒ Cloudy)

Rain		Cloudy	
w(R)	w(¬R)	w(C)	w(¬C)
1	2	3	5

Rain	Cloudy	Model?	Weight
T	T	Yes	1 * 3 = 3
T	F	No	0
F	T	Yes	2 * 3 = 6
F	F	Yes	2 * 5 = 10

#SAT = 3 **WMC = 19**

Weighted Model Counting @ UAI

- Assembly language for non-lifted inference
- Reductions to WMC for inference in
  - Bayesian networks [Chavira'05, Sang'05 , Chavira'08]
  - Factor graphs [Choi'13]
  - Relational Bayesian networks [Chavira'06]
  - Probabilistic logic programs [Fierens'11, Fierens'13]
  - Probabilistic databases [Olteanu'08, Jha'13]
- State-of-the-art solvers
  - Knowledge compilation (WMC → d-DNNF → AC)

    Winner of the UAI'08 exact inference competition!
  - DPLL search

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d)
⇒ Cloudy(d))

Days = {Monday}

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d)
⇒ Cloudy(d))

Days = {Monday}

Rain(M)	Cloudy(M)	Model?
T	T	Yes
T	F	No
F	T	Yes
F	F	Yes

\+

#SAT = 3

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d) ⇒ Cloudy(d))

Days = {Monday, **Tuesday**}

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?
T	T	T	T	Yes
T	F	T	T	No
F	T	T	T	Yes
F	F	T	T	Yes
T	T	T	F	No
T	F	T	F	No
F	T	T	F	No
F	F	T	F	No
T	T	F	T	Yes
T	F	F	T	No
F	T	F	T	Yes
F	F	F	T	Yes
T	T	F	F	Yes
T	F	F	F	No
F	T	F	F	Yes
F	F	F	F	Yes

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

$\Delta = \forall d\ (\mathrm{Rain}(d) \Rightarrow \mathrm{Cloudy}(d))$

Days = {Monday **Tuesday**}

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?
T	T	T	T	Yes
T	F	T	T	No
F	T	T	T	Yes
F	F	T	T	Yes
T	T	T	F	No
T	F	T	F	No
F	T	T	F	No
F	F	T	F	No
T	T	F	T	Yes
T	F	F	T	No
F	T	F	T	Yes
F	F	F	T	Yes
T	T	F	F	Yes
T	F	F	F	No
F	T	F	F	Yes
F	F	F	F	Yes

+

#SAT = 9

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d) ⇒ Cloudy(d))

Days = {Monday Tuesday}

Rain

d	w(R(d))	w(¬R(d))
M	1	2
T	4	1

Cloudy

d	w(C(d))	w(¬C(d))
M	3	5
T	6	2

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?
T	T	T	T	Yes
T	F	T	T	No
F	T	T	T	Yes
F	F	T	T	Yes
T	T	T	F	No
T	F	T	F	No
F	T	T	F	No
F	F	T	F	No
T	T	F	T	Yes
T	F	F	T	No
F	T	F	T	Yes
F	F	F	T	Yes
T	T	F	F	Yes
T	F	F	F	No
F	T	F	F	Yes
F	F	F	F	Yes

+

#SAT = 9

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d) ⇒ Cloudy(d))

Days = {Monday Tuesday}

Rain

d	w(R(d))	w(¬R(d))
M	1	2
T	4	1

Cloudy

d	w(C(d))	w(¬C(d))
M	3	5
T	6	2

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?	Weight
T	T	T	T	Yes	1 * 3 * 4 * 6 = 72
T	F	T	T	No	0
F	T	T	T	Yes	2 * 3 * 4 * 6 = 144
F	F	T	T	Yes	2 * 5 * 4 * 6 = 240
T	T	T	F	No	0
T	F	T	F	No	0
F	T	T	F	No	0
F	F	T	F	No	0
T	T	F	T	Yes	1 * 3 * 1 * 6 = 18
T	F	F	T	No	0
F	T	F	T	Yes	2 * 3 * 1 * 6 = 36
F	F	F	T	Yes	2 * 5 * 1 * 6 = 60
T	T	F	F	Yes	1 * 3 * 1 * 2 = 6
T	F	F	F	No	0
F	T	F	F	Yes	2 * 3 * 1 * 2 = 12
F	F	F	F	Yes	2 * 5 * 1 * 2 = 20

+

#SAT = 9

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting

Model = solution to first-order logic formula Δ

Δ = ∀d (Rain(d) ⇒ Cloudy(d))

Days = {Monday, **Tuesday**}

Rain

d	w(R(d))	w(¬R(d))
M	1	2
T	4	1

Cloudy

d	w(C(d))	w(¬C(d))
M	3	5
T	6	2

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?	Weight
T	T	T	T	Yes	1 * 3 * 4 * 6 = 72
T	F	T	T	No	0
F	T	T	T	Yes	2 * 3 * 4 * 6 = 144
F	F	T	T	Yes	2 * 5 * 4 * 6 = 240
T	T	T	F	No	0
T	F	T	F	No	0
F	T	T	F	No	0
F	F	T	F	No	0
T	T	F	T	Yes	1 * 3 * 1 * 6 = 18
T	F	F	T	No	0
F	T	F	T	Yes	2 * 3 * 1 * 6 = 36
F	F	F	T	Yes	2 * 5 * 1 * 6 = 60
T	T	F	F	Yes	1 * 3 * 1 * 2 = 6
T	F	F	F	No	0
F	T	F	F	Yes	2 * 3 * 1 * 2 = 12
F	F	F	F	Yes	2 * 5 * 1 * 2 = 20

\+ ———— \+ ————

#SAT = 9 **WFOMC = 608**

[V.d.Broeck'11a, Gogate'11]

Weighted First-Order Model Counting @ UAI

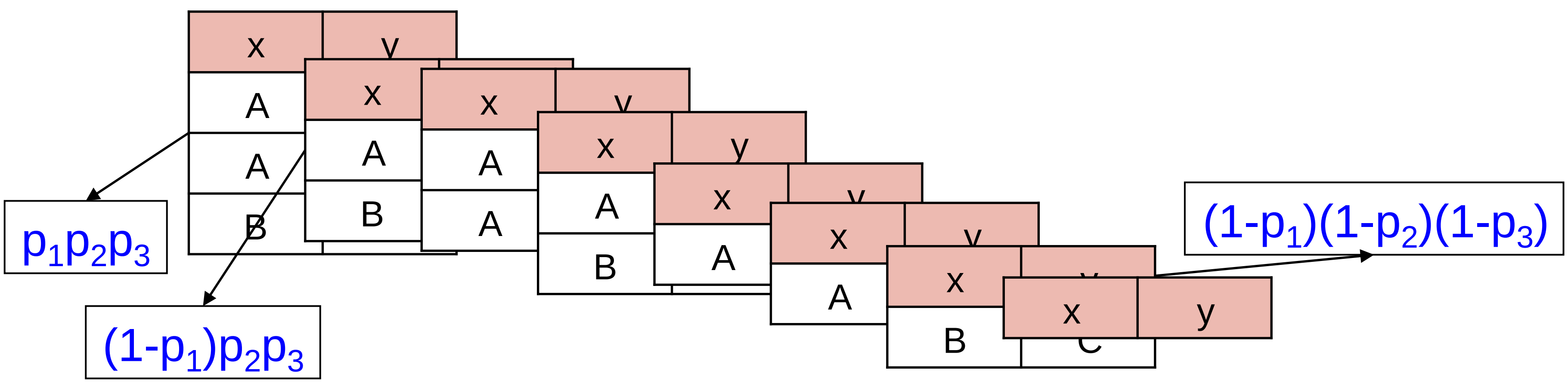
- Assembly language for lifted inference
- Reduction to WFOMC for lifted inference in
  - Markov logic networks [V.d.Broeck'11a,Gogate'11]
  - Parfactor graphs [V.d.Broeck'13a]
  - Probabilistic logic programs [V.d.Broeck'14]
  - Probabilistic databases [Gribkoff'14]

From Probabilities to Weights

Friend

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

→

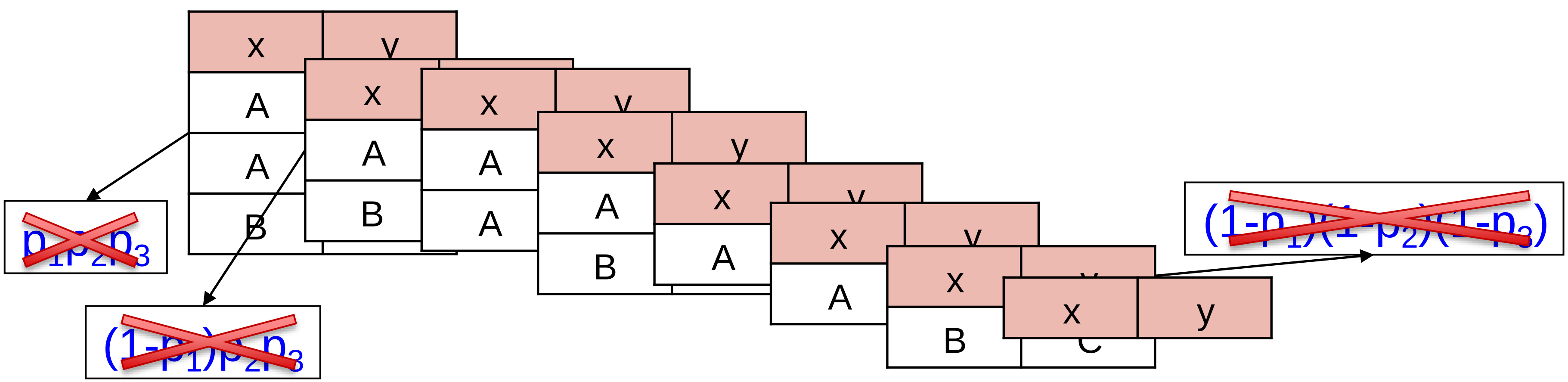


From Probabilities to Weights

Friend

x	y	P
A	B	$\cancel{p_1}$
A	C	$\cancel{p_2}$
B	C	$\cancel{p_3}$

→



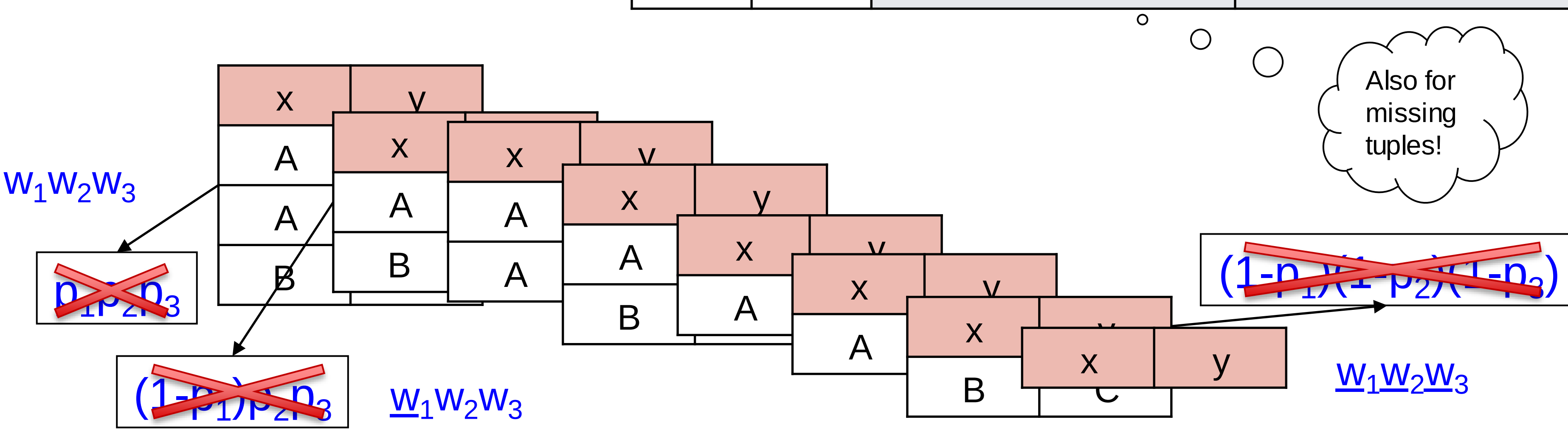
From Probabilities to Weights

Friend

x	y	P
A	B	$\cancel{p_1}$
A	C	$\cancel{p_2}$
B	C	$\cancel{p_3}$

→

x	y	w(Friend(x,y))	w(¬Friend(x,y))
A	B	$w_1 = p_1$	$\underline{w}_1 = 1-p_1$
A	C	$w_2 = p_2$	$\underline{w}_2 = 1-p_2$
B	C	$w_3 = p_3$	$\underline{w}_3 = 1-p_3$
A	A	$w_4 = 0$	$\underline{w}_4 = 1$
A	C	$w_5 = 0$	$\underline{w}_5 = 1$
	…	…	



Discussion

- Simple idea: replace p, $1\text{-}p$ by w, $\underline{w}$
- Query computation becomes WFOMC

- To obtain a probability space, divide the weight of each world by Z = sum of weights of all worlds:

  $Z = (w_1+\underline{w}_1)\ (w_2+\underline{w}_2)\ (w_3+\underline{w}_3)\ \ldots$

- Why weights instead of probabilities?
  They can describe complex correlations (next)

Markov Logic

Capture knowledge through constraints (a.k.a. “features”):

Hard constraint

∞ Smoker(x) $\Rightarrow$ Person(x)

Soft constraint,
weight = exp(3.73)

3.75 Smoker(x) $\wedge$ Friend(x,y) $\Rightarrow$ Smoker(y)

[Richardson’06]

Markov Logic

Capture knowledge through constraints (a.k.a. "features"):

Hard constraint

∞ Smoker(x) $\Rightarrow$ Person(x)

Soft constraint,
weight = exp(3.73)

3.75 Smoker(x) $\wedge$ Friend(x,y) $\Rightarrow$ Smoker(y)

An MLN is a set of constraints (w, $\Gamma(\mathbf{x})$), where w=weight, $\Gamma(\mathbf{x})$=FO formula

[Richardson'06]

Markov Logic

Capture knowledge through constraints (a.k.a. “features”):

Hard constraint

∞	Smoker(x) $\Rightarrow$ Person(x)

Soft constraint,
weight = exp(3.73)

3.75	Smoker(x) $\wedge$ Friend(x,y) $\Rightarrow$ Smoker(y)

An MLN is a set of constraints (w, $\Gamma(\mathbf{x})$), where w=weight, $\Gamma(\mathbf{x})$=FO formula

Weight of a world = product of exp(w), for all MLN rules (w, $\Gamma(\mathbf{x})$)
and grounding $\Gamma(\mathbf{a})$ that hold in that world

[Richardson’06]

Markov Logic

Capture knowledge through constraints (a.k.a. “features”):

Hard constraint

∞ Smoker(x) $\Rightarrow$ Person(x)

Soft constraint,
weight = exp(3.73)

3.75 Smoker(x) $\wedge$ Friend(x,y) $\Rightarrow$ Smoker(y)

An MLN is a set of constraints (w, $\Gamma(\mathbf{x})$), where w=weight, $\Gamma(\mathbf{x})$=FO formula

Weight of a world = product of exp(w), for all MLN rules (w, $\Gamma(\mathbf{x})$)
and grounding $\Gamma(\mathbf{a})$ that hold in that world

Probability of a world = Weight / Z
Z = sum of weights of all worlds (no longer a simple expression!)

[Richardson’06]

Problem Statement

Given:

MLN:

0.7 Actor(a) ⇒ ¬Director(a)
1.2 Director(a) ⇒ ¬WorkedFor(a,b)
1.4 InMovie(m,a) ∧ WorkedFor(a,b) ⇒ InMovie(m,b)

Database tables (if missing, then w = 1)

Actor:

Name	w
Brando	2.9
Cruise	3.8
Coppola	1.1

WorkedFor:

Actor	Director	w
Brando	Coppola	2.5
Coppola	Brando	0.2
Cruise	Coppola	1.7

Compute:

P(InMovie(GodFather, Brando) = ??

Discussion

- Probabilistic databases = independence
  MLN = complex correlations

- To translate weights to probabilities we need to divide by Z, which often is difficult to compute

- However, we can reduce the Z-computation problem to WFOMC (next)

$Z \rightarrow WFOMC(\Delta)$

1. **<u>Formula Δ</u>**

2. **<u>Weight function w(.)</u>**

$Z \rightarrow WFOMC(\Delta)$

1. **Formula Δ**

If all MLN constraints are hard: $\boxed{\Delta = \bigwedge_{(\infty,\Gamma(\mathbf{x}))\in MLN} (\forall \mathbf{x}\ \Gamma(\mathbf{x}))}$

2. **Weight function w(.)**

$Z \rightarrow \text{WFOMC}(\Delta)$

1. **Formula Δ**

If all MLN constraints are hard: $\boxed{\Delta = \bigwedge_{(\infty, \Gamma(\mathbf{x})) \in \text{MLN}} (\forall \mathbf{x}\ \Gamma(\mathbf{x}))}$

If $(w_i, \Gamma_i(\mathbf{x}))$ is a soft MLN constraint, then:

a) Remove $(w_i, \Gamma_i(\mathbf{x}))$ from the MLN
b) Add new probabilistic relation $F_i(\mathbf{x})$
c) Add hard constraint $(\infty, \forall \mathbf{x}\ (F_i(\mathbf{x}) \Leftrightarrow \Gamma_i(\mathbf{x})))$

2. **Weight function w(.)**

$Z \rightarrow \mathrm{WFOMC}(\Delta)$

1. Formula Δ

If all MLN constraints are hard: $\Delta = \bigwedge_{(\infty,\Gamma(\mathbf{x}))\in MLN} (\forall \mathbf{x}\ \Gamma(\mathbf{x}))$

If $(w_i, \Gamma_i(\mathbf{x}))$ is a soft MLN constraint, then:

a) Remove $(w_i, \Gamma_i(\mathbf{x}))$ from the MLN
b) Add new probabilistic relation $F_i(\mathbf{x})$
c) Add hard constraint $(\infty, \forall \mathbf{x}\ (F_i(\mathbf{x}) \Leftrightarrow \Gamma_i(\mathbf{x})))$

2. Weight function w(.)

For all constants **A**, relations F_i,
set $w(F_i(\mathbf{A})) = \exp(w_i)$, $w(\neg F_i(\mathbf{A})) = 1$

Better rewritings in
[Jha'12],[V.d.Broeck'14]

$Z \rightarrow \text{WFOMC}(\Delta)$

1. Formula Δ

If all MLN constraints are hard: $\Delta = \bigwedge_{(\infty, \Gamma(\mathbf{x})) \in MLN} (\forall \mathbf{x} \; \Gamma(\mathbf{x}))$

If $(w_i, \Gamma_i(\mathbf{x}))$ is a soft MLN constraint, then:

a) Remove $(w_i, \Gamma_i(\mathbf{x}))$ from the MLN
b) Add new probabilistic relation $F_i(\mathbf{x})$
c) Add hard constraint $(\infty, \forall \mathbf{x} \; (F_i(\mathbf{x}) \Leftrightarrow \Gamma_i(\mathbf{x})))$

2. Weight function w(.)

For all constants **A**, relations F_i,
set $w(F_i(\mathbf{A})) = \exp(w_i)$, $w(\neg F_i(\mathbf{A})) = 1$

Theorem: $Z = \text{WFOMC}(\Delta)$

Better rewritings in [Jha'12],[V.d.Broeck'14]

Example

1. **Formula Δ**

2. **Weight function w(.)**

Example

1. **Formula Δ**

∞ Smoker(x) ⇒ Person(x)

2. **Weight function w(.)**

Example

1. **<u>Formula Δ</u>**

∞ Smoker(x) ⇒ Person(x)

$\Delta = \forall x\,(\text{Smoker}(x) \Rightarrow \text{Person}(x))$

2. **<u>Weight function w(.)</u>**

Example

∞	Smoker(x) $\Rightarrow$ Person(x)
3.75	Smoker(x) $\wedge$ Friend(x,y) $\Rightarrow$ Smoker(y)

1. **Formula Δ**

$\Delta = \forall x\ (\text{Smoker}(x) \Rightarrow \text{Person}(x))$

2. **Weight function w(.)**

Example

1. **<u>Formula Δ</u>**

∞	Smoker(x) ⇒ Person(x)
3.75	Smoker(x) ∧ Friend(x,y) ⇒ Smoker(y)

$$\Delta = \forall x\,(\text{Smoker}(x) \Rightarrow \text{Person}(x))$$
$$\wedge\ \forall x \forall y\,(F(x,y) \Leftrightarrow [\text{Smoker}(x) \wedge \text{Friend}(x,y) \Rightarrow \text{Smoker}(y)])$$

2. **<u>Weight function w(.)</u>**

Example

1. Formula Δ

∞	Smoker(x) ⇒ Person(x)
3.75	Smoker(x) ∧ Friend(x,y) ⇒ Smoker(y)

$$\Delta = \forall x\,(\text{Smoker}(x) \Rightarrow \text{Person}(x))$$
$$\wedge\ \forall x \forall y\,(F(x,y) \Leftrightarrow [\text{Smoker}(x) \wedge \text{Friend}(x,y) \Rightarrow \text{Smoker}(y)])$$

2. Weight function w(.)

F

x	y	w(F(x,y))	w(¬F(x,y))
A	A	exp(3.75)	1
A	B	exp(3.75)	1
A	C	exp(3.75)	1
B	A	exp(3.75)	1
	. . .	. . .	

Note: if no tables given for Smoker, Person, etc, (i.e. no evidence) then set their w = <u>w</u> = 1

Example

1. Formula Δ

∞	Smoker(x) ⇒ Person(x)
3.75	Smoker(x) ∧ Friend(x,y) ⇒ Smoker(y)

Δ = ∀x (Smoker(x) ⇒ Person(x))
∧ ∀x∀y (F(x,y) ⇔ [Smoker(x) ∧ Friend(x,y) ⇒ Smoker(y)])

2. Weight function w(.)

F

x	y	w(F(x,y))	w(¬F(x,y))
A	A	exp(3.75)	1
A	B	exp(3.75)	1
A	C	exp(3.75)	1
B	A	exp(3.75)	1
	. . .	. . .	

Note: if no tables given for Smoker, Person, etc, (i.e. no evidence) then set their w = $\underline{w}$ = 1

Z = WFOMC(Δ)

Lessons

- Weighed Model Counting:
  - Unified framework for probabilistic inference tasks
  - Independent variables
- Weighed FO Model Counting:
  - Formula described by a concise FO sentence
  - Still independent variables
- MLN:
  - Formulas plus weights
  - Correlations!
  - Can be converted to WFOMC

Symmetric vs. Asymmetric

Symmetric WFOMC:

- In every relation R, all tuples have same weight
- Example: converting MLN “without evidence” into WFOMC leads to a symmetric weight function

Asymmetric WFOMC:

- Each relation R is given explicitly
- Example: Probabilistic Databases
- Example: MLN’s plus evidence

[Gribkoff’14]

Terminology

	MLNs	Prob. DBs
Random variable is a	Ground atom	DB Tuple
Weights w associated with	Formulas	DB Tuples
Typical query Q is a	Single atom	FO formula/SQL
Data is encoded into	Evidence (Query)	Distribution
Correlations induced by	Model formulas	Query
Model generalizes across domains?	Yes	No
Query generalizes across domains?	No	Yes
Sum of weights of worlds is 1 (normalized)?	No	Yes

Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc.

- A formal definition: **Domain-lifted inference**

> Inference runs in time polynomial
> in the number of objects in the domain.

  – Polynomial in #people, #webpages, #cards
  – Not polynomial in #predicates, #formulas, #logical variables
  – Related to data complexity in databases

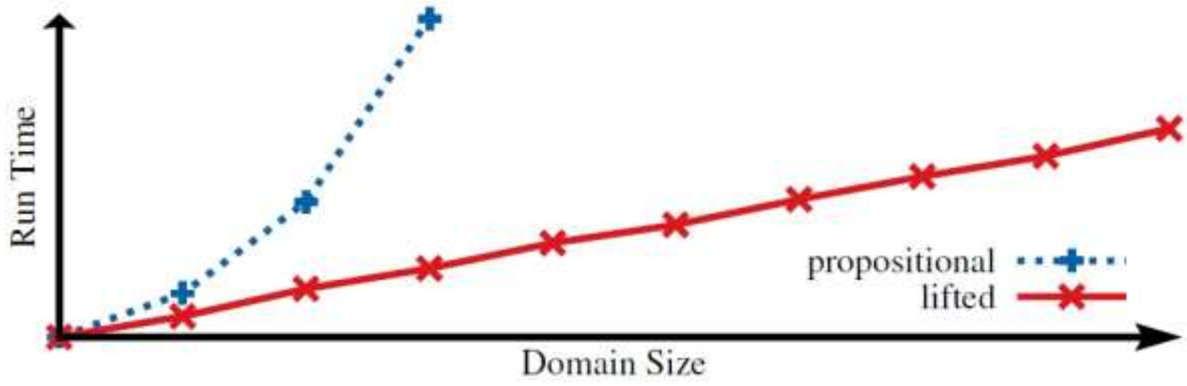
[V.d.Broeck'11b, Jaeger'12]

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc.

- A formal definition: **Domain-lifted inference**



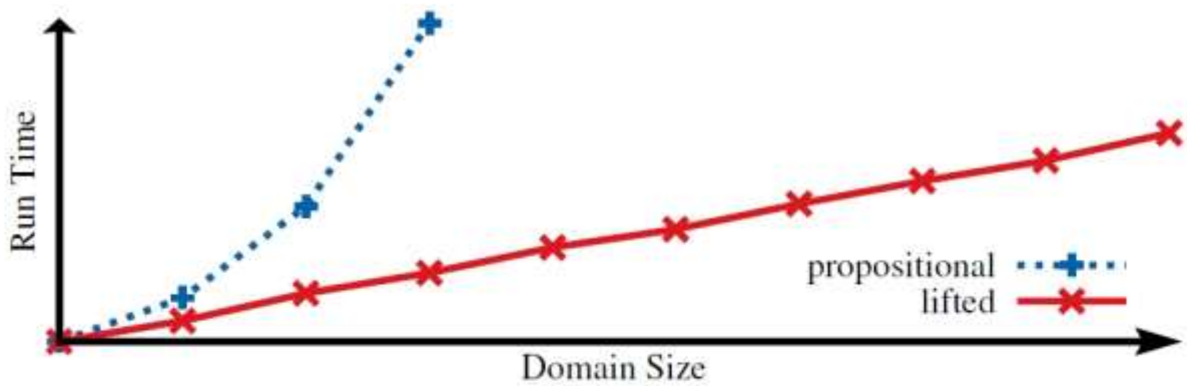
[V.d.Broeck'11b, Jaeger'12]

Defining Lifted Inference

- Informal:

  Exploit symmetries, Reason at first-order level, Reason about groups of objects, Scalable inference, High-level probabilistic reasoning, etc.

- A formal definition: **Domain-lifted inference**



- Alternative in this tutorial:

Lifted inference = ∃Query Plan = ∃FO Compilation

[V.d.Broeck'11b, Jaeger'12]

Rules for Asymmetric WFOMC

- If Δ_1, Δ_2 are independent:
  AND-rule: $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) * WMC(\Delta_2)$
  OR-rule: $WMC(\Delta_1 \vee \Delta_2) = Z - (Z_1 - WMC(\Delta_1)) * (Z_2 - WMC(\Delta_2))$

Rules for Asymmetric WFOMC

Normalization constants
(easy to compute)

- If Δ_1, Δ_2 are independent:

  AND-rule: $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) * WMC(\Delta_2)$

  OR-rule: $WMC(\Delta_1 \vee \Delta_2) = Z - (Z_1 - WMC(\Delta_1)) * (Z_2 - WMC(\Delta_2))$

Rules for Asymmetric WFOMC

Normalization constants (easy to compute)

- If Δ_1, Δ_2 are independent:
  AND-rule: $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) * WMC(\Delta_2)$
  OR-rule: $WMC(\Delta_1 \vee \Delta_2) = Z - (Z_1 - WMC(\Delta_1)) * (Z_2 - WMC(\Delta_2))$

- If $\Delta[c_1/x]$, $\Delta[c_2/x]$, … are independent
  $\forall$-Rule: $WMC(\forall z\, \Delta) = \prod_{c \in Domain} WMC(\Delta[c/z])$
  $\exists$-Rule: $WMC(\exists z\, \Delta) = Z - \prod_{c \in Domain} (Z_c - WMC(\Delta[c/z])$

Rules for Asymmetric WFOMC

Normalization constants (easy to compute)

- If Δ_1, Δ_2 are independent:
  AND-rule: $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) * WMC(\Delta_2)$
  OR-rule: $WMC(\Delta_1 \vee \Delta_2) = Z - (Z_1 - WMC(\Delta_1)) * (Z_2 - WMC(\Delta_2))$

- If $\Delta[c_1/x]$, $\Delta[c_2/x]$, … are independent
  ∀-Rule: $WMC(\forall z\, \Delta) = \prod_{c \in Domain} WMC(\Delta[c/z])$
  ∃-Rule: $WMC(\exists z\, \Delta) = Z - \prod_{c \in Domain} (Z_c - WMC(\Delta[c/z])$

- Inclusion/Exclusion formula:
  $WMC(\Delta_1 \vee \Delta_2) = WMC(\Delta_1) + WMC(\Delta_2) - WMC(\Delta_1 \wedge \Delta_2)$
  $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) + WMC(\Delta_2) - WMC(\Delta_1 \vee \Delta_2)$

Rules for Asymmetric WFOMC

Normalization constants (easy to compute)

- If Δ_1, Δ_2 are independent:
  AND-rule: $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) * WMC(\Delta_2)$
  OR-rule: $WMC(\Delta_1 \vee \Delta_2) = Z - (Z_1 - WMC(\Delta_1)) * (Z_2 - WMC(\Delta_2))$

- If $\Delta[c_1/x]$, $\Delta[c_2/x]$, … are independent
  $\forall$-Rule: $WMC(\forall z\, \Delta) = \Pi_{c \in Domain}\ WMC(\Delta[c/z])$
  $\exists$-Rule: $WMC(\exists z\, \Delta) = Z - \Pi_{c \in Domain}\ (Z_c - WMC(\Delta[c/z])$

- Inclusion/Exclusion formula:
  $WMC(\Delta_1 \vee \Delta_2) = WMC(\Delta_1) + WMC(\Delta_2) - WMC(\Delta_1 \wedge \Delta_2)$
  $WMC(\Delta_1 \wedge \Delta_2) = WMC(\Delta_1) + WMC(\Delta_2) - WMC(\Delta_1 \vee \Delta_2)$

- Negation: $WMC(\neg\Delta) = Z - WMC(\Delta)$

Symmetric WFOMC Rules

- Simplifications:

If $\Delta[c_1/x]$, $\Delta[c_2/x]$, … are independent
∀-Rule: $WMC(\forall z\, \Delta) = WMC(\Delta[c_1/z])^{|Domain|}$
∃-Rule: $WMC(\exists z\, \Delta) = Z - (Z_{c_1} - WMC(\Delta[c_1/z])^{|Domain|}$

- A powerful new inference rule: atom counting
  Only possible with symmetric weights
  Intuition: **Remove unary relations**

[V.d.Broeck'11a]

Symmetric WFOMC Rules

- Simplifications:

  If $\Delta[c_1/x]$, $\Delta[c_2/x]$, … are independent

  ∀-Rule: $WMC(\forall z\, \Delta) = WMC(\Delta[c_1/z])^{|Domain|}$

  ∃-Rule: $WMC(\exists z\, \Delta) = Z - (Z_{c_1} - WMC(\Delta[c_1/z])^{|Domain|}$

- A powerful new inference rule: atom counting

  Only possible with symmetric weights

  Intuition: **Remove unary relations**

The workhorse of Symmetric WFOMC

[V.d.Broeck'11a]

Symmetric WFOMC Rules: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

Symmetric WFOMC Rules: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

4. Δ = (Stress(Alice) ⇒ Smokes(Alice))

Domain = {Alice}

Symmetric WFOMC Rules: Example

- FO-Model Counting: w(R) = w(¬R) = 1
- Apply inference rules backwards (step 4-3-2-1)

---

4. Δ = (Stress(Alice) ⇒ Smokes(Alice))

Domain = {Alice}

WMC(¬Stress(Alice) ∨ Smokes(Alice))) =

OR-rule

= Z – WMC(Stress(Alice)) × WMC(¬Smokes(Alice))

= 4 – 1 × 1 = 3 models

Symmetric WFOMC Rules: Example

- FO-Model Counting: w(R) = w(¬R) = 1
- Apply inference rules backwards (step 4-3-2-1)

---

4. Δ = (Stress(Alice) ⇒ Smokes(Alice))

Domain = {Alice}

WMC(¬Stress(Alice) ∨ Smokes(Alice))) =

OR-rule

= Z – WMC(Stress(Alice)) × WMC(¬Smokes(Alice))

= 4 – 1 × 1 = 3 models

---

3. Δ = ∀x, (Stress(x) ⇒ Smokes(x))

Domain = {n people}

Symmetric WFOMC Rules: Example

- FO-Model Counting: $w(R) = w(\neg R) = 1$
- Apply inference rules backwards (step 4-3-2-1)

---

4. $\boxed{\Delta = (\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice}))}$ $\boxed{\text{Domain} = \{\text{Alice}\}}$

$$\begin{aligned} &\text{WMC}(\neg\text{Stress}(\text{Alice}) \vee \text{Smokes}(\text{Alice})) = \\ &= Z - \text{WMC}(\text{Stress}(\text{Alice})) \times \text{WMC}(\neg\text{Smokes}(\text{Alice})) \\ &= 4 - 1 \times 1 = 3 \text{ models} \end{aligned}$$

OR-rule

---

3. $\boxed{\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))}$ $\boxed{\text{Domain} = \{n \text{ people}\}}$

$\rightarrow 3^n$ models

∀-Rule

Symmetric WFOMC Rules: Example

3. $\boxed{\Delta = \forall x, (Stress(x) \Rightarrow Smokes(x))}$ $\boxed{\text{Domain} = \{n \text{ people}\}}$

$\rightarrow 3^n$ models

Symmetric WFOMC Rules: Example

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$ Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))$ D = {n people}

Symmetric WFOMC Rules: Example

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$ Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))$ D = {n people}

$$\begin{aligned}
\text{WMC}(\Delta) &= \text{WMC}(\neg \text{Female} \vee \forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y))) \\
&= 2 * 2^n * 2^n - (2-1) * (2^n * 2^n - \text{WMC}(\forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)))) \\
&= 2 * 4^n - (4^n - 3^n)
\end{aligned}$$

OR-Rule

Symmetric WFOMC Rules: Example

3. $\boxed{\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))}$ $\boxed{\text{Domain} = \{n \text{ people}\}}$

$\rightarrow 3^n$ models

2. $\boxed{\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))}$ $\boxed{D = \{n \text{ people}\}}$

$$\begin{aligned} \text{WMC}(\Delta) &= \text{WMC}(\neg\text{Female} \vee \forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y))) \\ &= 2 * 2^n * 2^n - (2 - 1) * (2^n * 2^n - \text{WMC}(\forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)))) \\ &= 2 * 4^n - (4^n - 3^n) \end{aligned}$$

$\rightarrow 3^n + 4^n$ models

OR-Rule

Symmetric WFOMC Rules: Example

3. $\boxed{\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))}$ $\boxed{\text{Domain} = \{n \text{ people}\}}$

$\rightarrow 3^n$ models

2. $\boxed{\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))}$ $\boxed{D = \{n \text{ people}\}}$

$$\begin{aligned}
\text{WMC}(\Delta) &= \text{WMC}(\neg \text{Female} \vee \forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y))) \\
&= 2 * 2^n * 2^n - (2 - 1) * (2^n * 2^n - \text{WMC}(\forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)))) \\
&= 2 * 4^n - (4^n - 3^n)
\end{aligned}$$

$\rightarrow 3^n + 4^n$ models

OR-Rule

1. $\boxed{\Delta = \forall x,y, (\text{ParentOf}(x,y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x,y))}$ $\boxed{D = \{n \text{ people}\}}$

Symmetric WFOMC Rules: Example

3. $\Delta = \forall x, (\text{Stress}(x) \Rightarrow \text{Smokes}(x))$ Domain = {n people}

$\rightarrow 3^n$ models

2. $\Delta = \forall y, (\text{ParentOf}(y) \wedge \text{Female} \Rightarrow \text{MotherOf}(y))$ D = {n people}

$$\begin{aligned}
\text{WMC}(\Delta) &= \text{WMC}(\neg \text{Female} \vee \forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y))) \\
&= 2 * 2^n * 2^n - (2-1) * (2^n * 2^n - \text{WMC}(\forall y, (\text{ParentOf}(y) \Rightarrow \text{MotherOf}(y)))) \\
&= 2 * 4^n - (4^n - 3^n)
\end{aligned}$$

$\rightarrow 3^n + 4^n$ models

OR-Rule

1. $\Delta = \forall x,y, (\text{ParentOf}(x,y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x,y))$ D = {n people}

$\rightarrow (3^n + 4^n)^n$ models

∀-Rule

Atom Counting: Example

Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

Atom Counting: Example

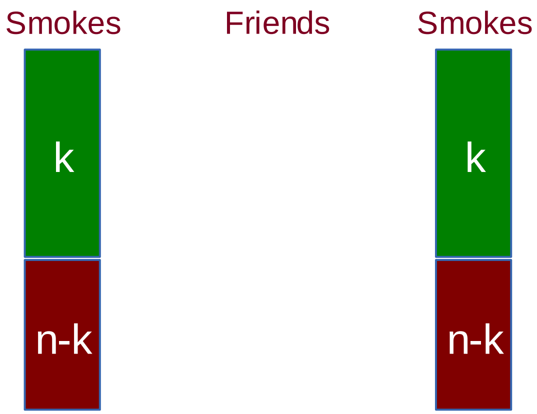
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

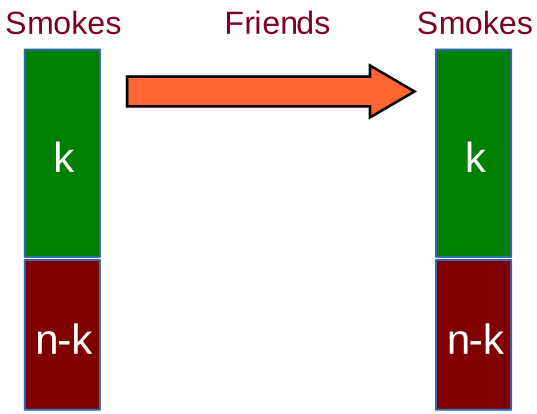
$\Delta = \forall x,y, (Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y))$

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

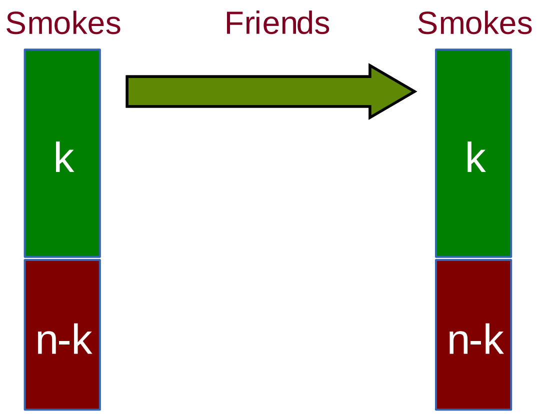
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

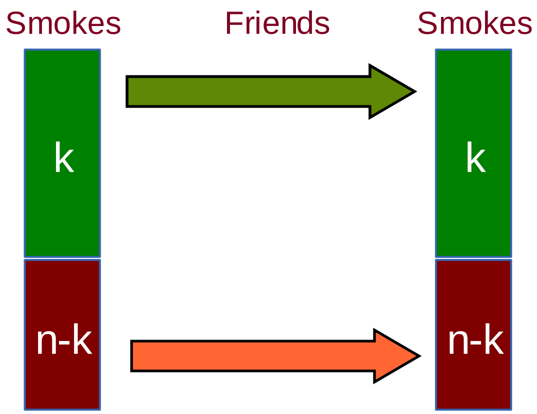
$\Delta = \forall x,y, (Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y))$

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

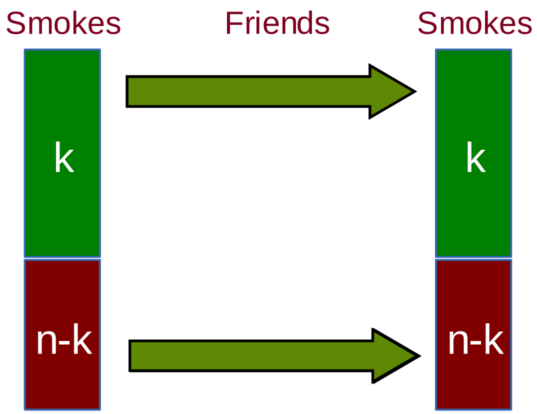
$\Delta = \forall x,y, (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

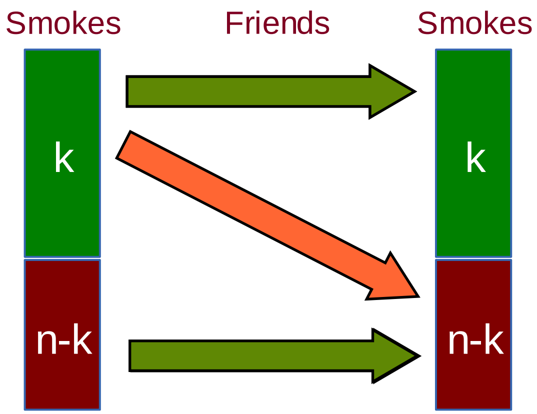
$\Delta = \forall x,y, (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

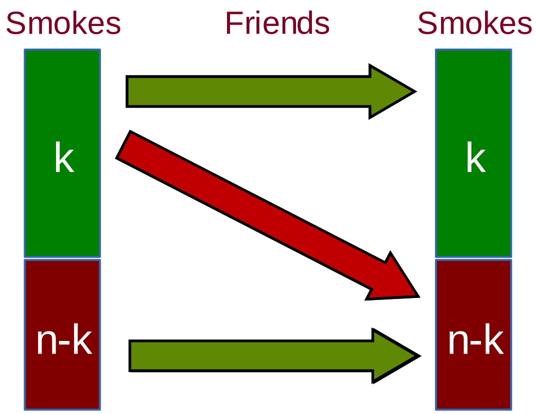
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

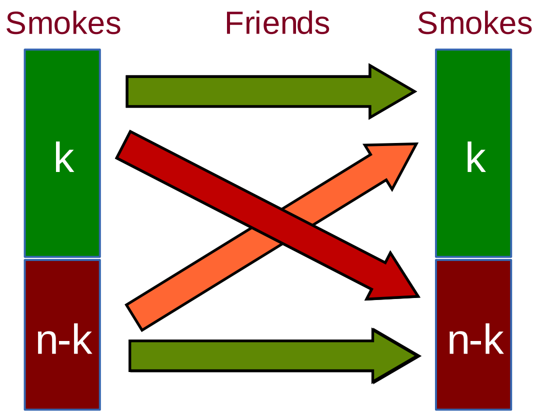
$\Delta = \forall x,y, (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

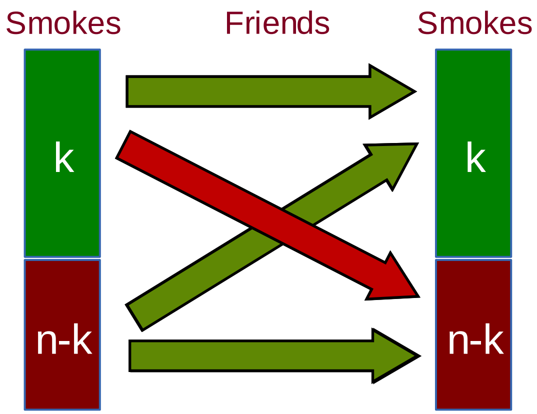
Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...



Atom Counting: Example

Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

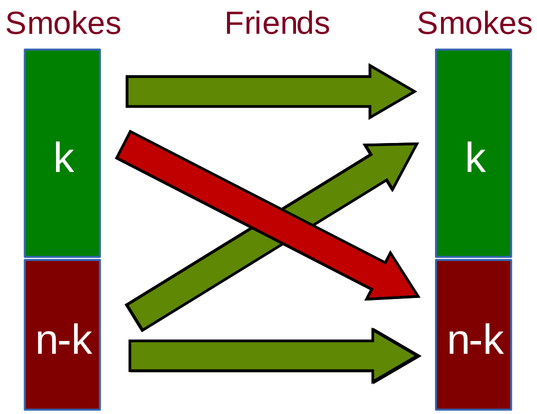
Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

$\rightarrow 2^{n^2-k(n-k)}$ models



Atom Counting: Example

Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

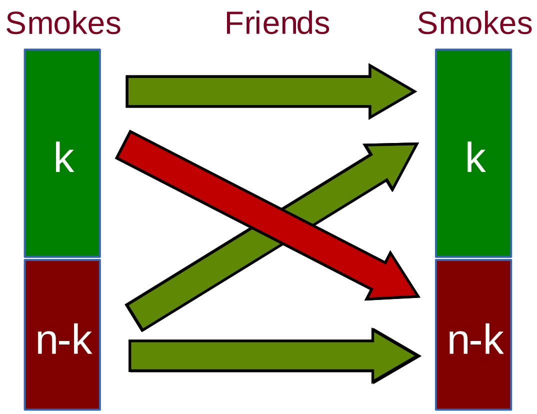
Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

$\rightarrow 2^{n^2-k(n-k)}$ models



- If we know that there are *k* smokers?

Atom Counting: Example

Δ = ∀x,y, (Smokes(x) ∧ Friends(x,y) ⇒ Smokes(y))

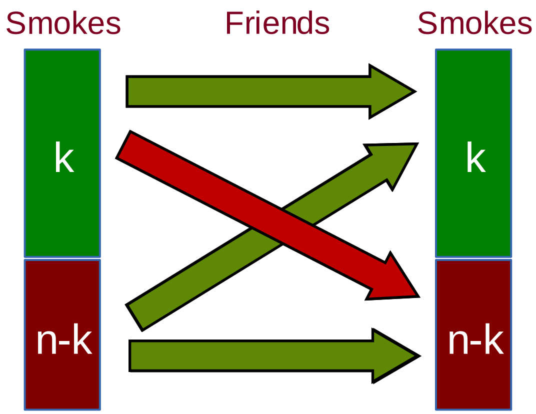
Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

$\rightarrow 2^{n^2-k(n-k)}$ models



- If we know that there are *k* smokers? $\rightarrow \binom{n}{k} 2^{n^2-k(n-k)}$ models

Atom Counting: Example

$\Delta = \forall x,y, (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

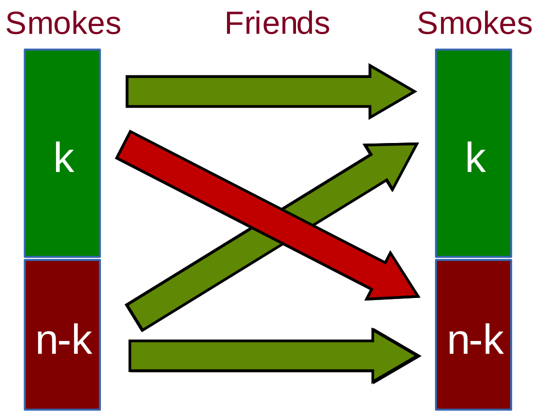
Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

$\rightarrow 2^{n^2 - k(n-k)}$ models



- If we know that there are *k* smokers? $\rightarrow \binom{n}{k} 2^{n^2 - k(n-k)}$ models

- In total…

Atom Counting: Example

$\Delta = \forall x,y,\ (\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y))$

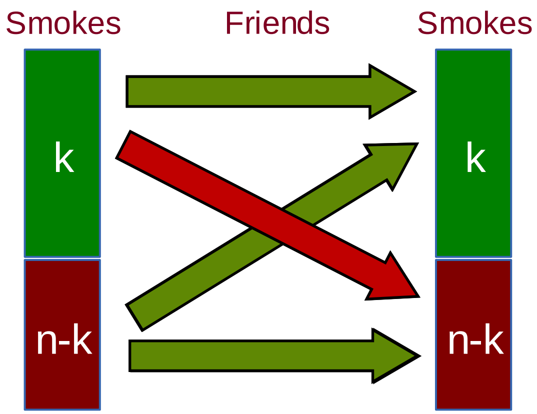
Domain = {n people}

- If we know precisely who smokes, and there are *k* smokers?

Database:

Smokes(Alice) = 1
Smokes(Bob) = 0
Smokes(Charlie) = 0
Smokes(Dave) = 1
Smokes(Eve) = 0
...

$\rightarrow 2^{n^2 - k(n-k)}$ models



- If we know that there are *k* smokers? $\rightarrow \binom{n}{k} 2^{n^2 - k(n-k)}$ models

- In total… $\rightarrow \sum_{k=0}^{n} \binom{n}{k} 2^{n^2 - k(n-k)}$ models

Augment Rules with Logical Rewritings

Augment Rules with Logical Rewritings

1. Remove constants (shattering)

$\Delta = \forall x\ (\text{Friend}(\text{Alice}, x) \lor \text{Friend}(x, \text{Bob}))$

Augment Rules with Logical Rewritings

$F_1(x)$ = Friend(Alice,x)
$F_2(x)$ = Friend(x,Bob)
F_3 = Friend(Alice, Alice)
F_4 = Friend(Alice,Bob)
F_5 = Friend(Bob,Bob)

1. Remove constants (shattering)

$\Delta = \forall x\ (\text{Friend}(\text{Alice}, x) \vee \text{Friend}(x, \text{Bob}))$

$\Rightarrow$ $\Delta = \forall x\ (F_1(x) \vee F_2(x)) \wedge (F_3 \vee F_4) \wedge (F_4 \vee F_5)$

Augment Rules with Logical Rewritings

$F_1(x)$ = Friend(Alice,x)
$F_2(x)$ = Friend(x,Bob)
F_3 = Friend(Alice, Alice)
F_4 = Friend(Alice,Bob)
F_5 = Friend(Bob,Bob)

1. Remove constants (shattering)

$\Delta = \forall x\,(\text{Friend}(\text{Alice}, x) \vee \text{Friend}(x, \text{Bob}))$

$\Rightarrow$ $\Delta = \forall x\,(F_1(x) \vee F_2(x)) \wedge (F_3 \vee F_4) \wedge (F_4 \vee F_5)$

2. "Rank" variables (= occur in the same order in each atom)

$\Delta = (\text{Friend}(x,y) \vee \text{Enemy}(x,y)) \quad \wedge \quad (\text{Friend}(x,y) \vee \text{Enemy}(y,x))$

Wrong order

Augment Rules with Logical Rewritings

$F_1(x)$ = Friend(Alice,x)
$F_2(x)$ = Friend(x,Bob)
F_3 = Friend(Alice, Alice)
F_4 = Friend(Alice,Bob)
F_5 = Friend(Bob,Bob)

1. Remove constants (shattering)

$\Delta = \forall x\, (\text{Friend}(\text{Alice}, x) \vee \text{Friend}(x, \text{Bob}))$

$\Rightarrow$ $\Delta = \forall x\, (F_1(x) \vee F_2(x)) \wedge (F_3 \vee F_4) \wedge (F_4 \vee F_5)$

---

2. "Rank" variables (= occur in the same order in each atom)

$\Delta = (\text{Friend}(x,y) \vee \text{Enemy}(x,y)) \quad \wedge \quad (\text{Friend}(x,y) \vee \text{Enemy}(y,x))$

Wrong order

$F_1(u,v)$ = Friend(u,v),u<v
$F_2(u)$ = Friend(u,u)
$F_3(u,v)$ = Friend(v,u),v<u

$E_1(u,v)$ = Friend(u,v),u<v
$E_2(u)$ = Friend(u,u)
$E_3(u,v)$ = Friend(v,u),v<u

$\Rightarrow$

$$\begin{aligned}\Delta = &(F_1(x,y) \vee E_1(x,y)) \wedge (F_1(x,y) \vee E_3(x,y)) \\ &\wedge (F_2(x) \vee E_2(x)) \\ &\wedge (F_3(x,y) \vee E_3(x,y)) \wedge (F_3(x,y) \vee E_1(x,y))\end{aligned}$$

Augment Rules with Logical Rewritings

3. Perform Resolution [Gribkoff'14]

$\Delta = \forall x \forall y\, (R(x) \lor \neg S(x,y)) \land \forall x \forall y\, (S(x,y) \lor T(y))$

Rules stuck…

Resolution: $\Delta \land \forall x \forall y\, (R(x) \lor T(y))$

Now apply I/E!

See UAI Poster on Saturday!

4. Skolemization [V.d.Broeck'14]

$\Delta = \forall p, \exists c, \mathrm{Card}(p,c)$

Mix ∀/∃ in encodings of MLNs with quantifiers and probabilistic programs

Input: Mix ∀/∃ Output: Only ∀

Skolemization: Example

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

[V.d.Boeck’14]

Skolemization: Example

$\Delta = \forall p, \exists c, Card(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, Card(p,c) \Rightarrow S(p)$

[V.d.Boeck'14]

Skolemization: Example

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

[V.d.Boeck'14]

Skolemization: Example

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\exists c, \text{Card}(p,c) = \text{true}$

$\exists c, \text{Card}(p,c) = \text{false}$

[V.d.Boeck'14]

Skolemization: Example

$\boxed{\Delta = \forall p, \exists c, \mathrm{Card}(p,c)}$

Skolemization

$\boxed{\Delta' = \forall p, \forall c, \mathrm{Card}(p,c) \Rightarrow S(p)}$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\boxed{\exists c, \mathrm{Card}(p,c) = \text{true}}$

$S(p) = \text{true}$

Also model of Δ, weight * 1

$\boxed{\exists c, \mathrm{Card}(p,c) = \text{false}}$

[V.d.Boeck'14]

Skolemization: Example

$\Delta = \forall p, \exists c, \text{Card}(p,c)$

Skolemization

$\Delta' = \forall p, \forall c, \text{Card}(p,c) \Rightarrow S(p)$

$w(S) = 1$ and $w(\neg S) = -1$

Skolem predicate

Consider one position p:

$\exists c, \text{Card}(p,c)$ = true

- $S(p)$ = true — Also model of Δ, weight * 1

$\exists c, \text{Card}(p,c)$ = false

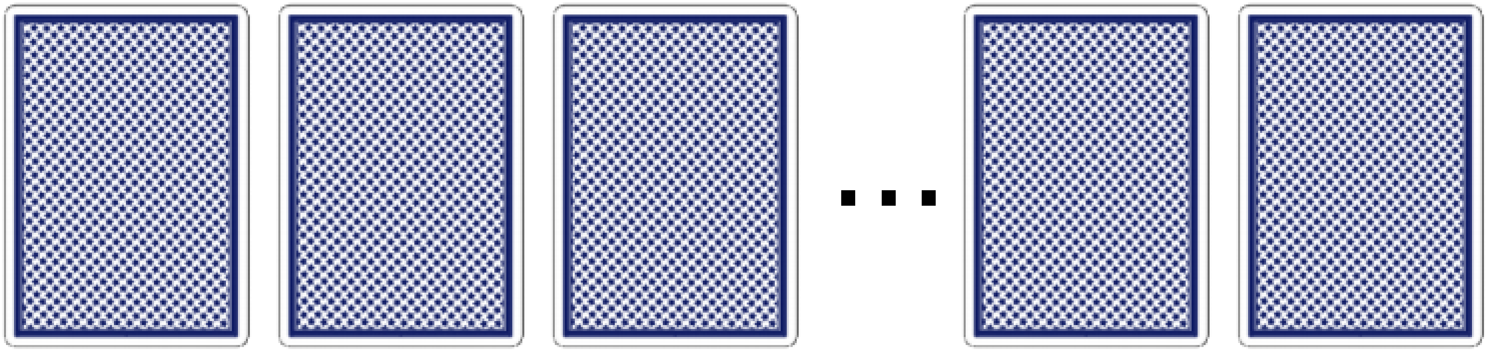
- $S(p)$ = true — No model of Δ, weight * 1
- $S(p)$ = false — No model of Δ, weight * -1

Extra models

Cancel out

[V.d.Boeck'14]

Playing Cards Revisited



Let us automate this:

– **Relational** model

$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

– **Lifted** probabilistic inference algorithm

Playing Cards Revisited

$$\forall p, \exists c, \mathrm{Card}(p,c)$$
$$\forall c, \exists p, \mathrm{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \mathrm{Card}(p,c) \land \mathrm{Card}(p,c') \Rightarrow c = c'$$

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow S_1(p)$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow S_2(c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow S_1(p)$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow S_2(c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \land \text{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

↓ Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

↓ Atom counting

Playing Cards Revisited

$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, Card(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, Card(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

Playing Cards Revisited

$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, Card(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, Card(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$\downarrow$ $\forall$-Rule

Playing Cards Revisited

$$\forall p, \exists c, \text{Card}(p,c)$$
$$\forall c, \exists p, \text{Card}(p,c)$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, \text{Card}(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, \text{Card}(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

$$\forall p, \forall c, \forall c', \text{Card}(p,c) \wedge \text{Card}(p,c') \Rightarrow c = c'$$

$\downarrow$ $\forall$-Rule

$$\forall c, \forall c', \text{Card}(c) \wedge \text{Card}(c') \Rightarrow c = c'$$

Playing Cards Revisited

$$\forall p, \exists c, Card(p,c)$$
$$\forall c, \exists p, Card(p,c)$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$\downarrow$ Skolemization

$$\forall p, \forall c, Card(p,c) \Rightarrow \cancel{S_1(p)}$$
$$\forall c, \forall p, Card(p,c) \Rightarrow \cancel{S_2(c)}$$
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$w(S_1) = 1$ and $w(\neg S_1) = -1$

$w(S_2) = 1$ and $w(\neg S_2) = -1$

$\downarrow$ Atom counting

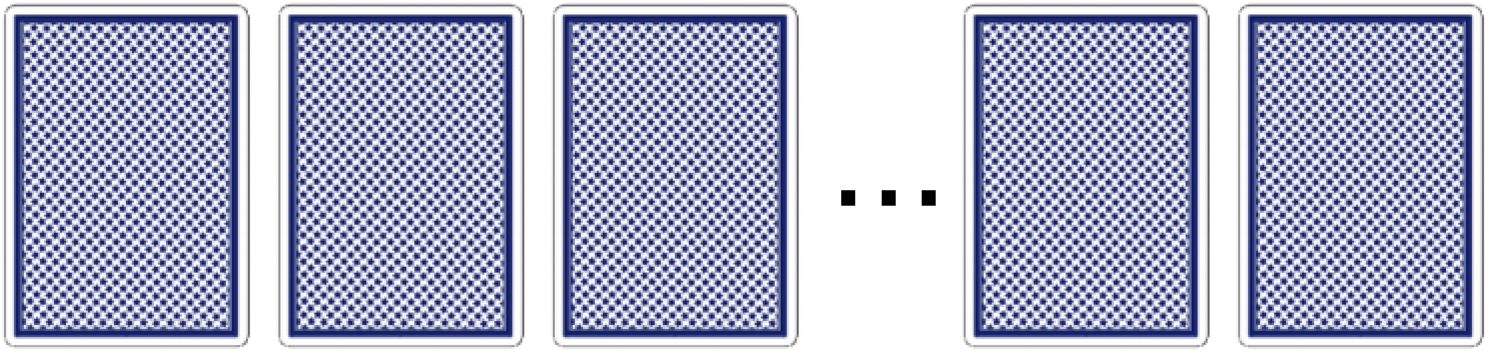
$$\forall p, \forall c, \forall c', Card(p,c) \wedge Card(p,c') \Rightarrow c = c'$$

$\downarrow$ $\forall$-Rule

$$\forall c, \forall c', Card(c) \wedge Card(c') \Rightarrow c = c'$$

$\downarrow$ …

Playing Cards Revisited



Let us automate this:

– **Lifted** probabilistic inference algorithm

$$\#SAT = \sum_{k=0}^{n} \binom{n}{k} \sum_{l=0}^{n} \binom{n}{l} (l+1)^k (-1)^{2n-k-l} = n!$$

Computed in time polynomial in n

Summary Lifted Inference

- By definition: PTIME data complexity
  Also: ∃ FO compilation = ∃ Query Plan
- However: only works for "liftable" queries
- The rules:
  - AND/OR-rules, ∀/∃-rules, I/E (inclusion/exclusion), Atom Counting
  - Deceptively simple: the only surprising rules are I/E and atom counting

Next: will show that lifted inference is provably more powerful than grounded inference

Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

Two Questions

- Q1: Are the lifted rules complete?
  - We know that they get stuck on some queries
  - Do we need to add more rules?

- Q2: Are lifted rules stronger than grounded?
  - Some lifted rules easily correspond to operations on grounded formulas (e.g. Independent-AND)
  - Can we simulate every lifted inference directly on the grounded formula?

Two Questions

- Q1: Are the lifted rules complete?
  - We know that they get stuck on some queries
  - Do we need to add more rules?

Complete for Positive CNF-FO, for UCQ

- Q2: Are lifted rules stronger than grounded?
  - Some lifted rules easily correspond to operations on grounded formulas (e.g. Independent-AND)
  - Can we simulate every lifted inference directly on the grounded formula?

Symmetric: yes (grounded inference ignores symmetries)
Asymmetric: Strictly stronger than Decision-DNNF & DPLL-based algorithms

1. Are the Lifted Rules Complete?

We use complexity classes

- Inference rules: PTIME data complexity
- Some queries: #P-hard data complexity

Dichotomy Theorem for Positive CNF-FO:

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

Implies lifted rules are complete for Positive CNF-FO

Will show in two steps: **Small** and **Big Dichotomy Theorem**

NP v.s. #P

- SAT = Satisfiability Problem
- SAT is NP-complete [Cook'71]
- NP = <u>*decision problems*</u>
  polynomial-time, nondeterministic TM

- #SAT = model counting
- #SAT is #P-complete [Valiant'79]
- #P = <u>*numerical functions*</u>
  polynomial-time, nondeterministic TM,
  answer = #accepting computations

Note: it would be wrong to say "#SAT is NP-complete"

A Simple Propositional Formula that is Hard

A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

Where E = the edge set of a bipartite graph

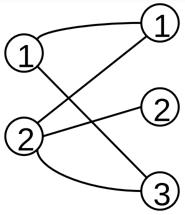
A Simple Propositional Formula that is Hard

A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

Where E = the edge set of a bipartite graph

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$

E: 

A Simple Propositional Formula that is Hard

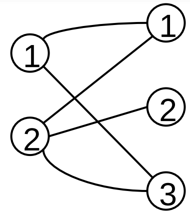
A **Positive, Partitioned 2CNF** Formula is a formula of the form:

$$F = \bigwedge_{(i,j) \in E} (x_i \vee y_j)$$

Where E = the edge set of a bipartite graph

$$F = (x_1 \vee y_1) \wedge (x_2 \vee y_1) \wedge (x_2 \vee y_3) \wedge (x_1 \vee y_3) \wedge (x_2 \vee y_2)$$

E:



Theorem [Provan'83] #SAT for PP2CNF is #P-hard

A Query That is #P-Hard

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

Theorem. Computing $P(H_0 \mid D)$ is #P-hard in $|D|$

[Dalvi'04]

Proof: Reduction from PP2CNF. Given a PP2CNF F defined by edge relation E, set:

$P(\text{Friend}(a,b)) = 1$ if $(a,b) \in E$

$P(\text{Friend}(a,b)) = 0$ if $(a,b) \notin E$

Then the grounding of H_0 is: $\bigwedge_{(i,j) \in E} (\text{Smoker}(i) \vee \text{Jogger}(j)) = F$

Hence, $P(H_0 \mid D) = P(F)$

Lesson: no lifted inference rules will ever compute H_0

Hierarchical Clause

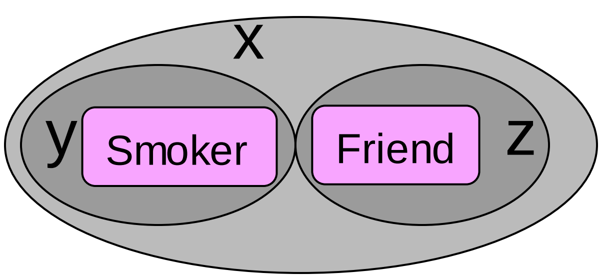
at(x) = set of atoms containing the variable x

Definition A clause Q is hierarchical if forall variables x, y:
$at(x) \supseteq at(y)$ or $at(x) \supseteq at(y)$ or $at(x) \cap at(y) = \emptyset$

Hierarchical

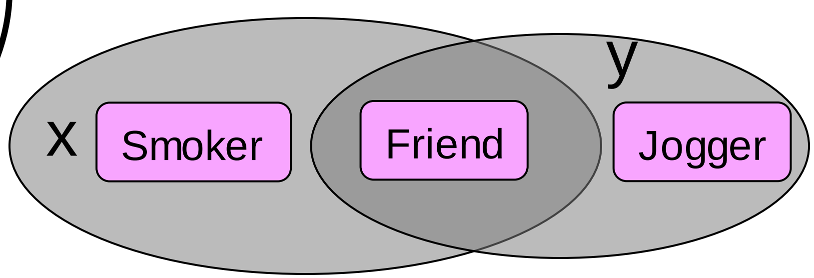
$Q = (Smoker(x,y) \vee Friend(x,z))$

$= \forall x[\forall y\ Smoker(x,y)] \vee [\forall z\ Friend(x,z)]$



Non-hierarchical

$H_0 = Smoker(x) \vee Friend(x,y) \vee Jogger(y)$



Small Dichotomy Theorem

Definition A clause Q is hierarchical if forall variables x, y:
$at(x) \supseteq at(y)$ or $at(x) \supseteq at(y)$ or $at(x) \cap at(y) = \emptyset$

Let Q be a single clause, w/o repeating relation symbols

Theorem [Dalvi'04] Dichotomy:

- If Q is hierarchical, then Q is liftable (PTIME data complexity)
- If Q is not hierarchical, Q is #P-hard

And, moreover, the OR-rule and ∀-rule are complete.

Note: checking "Q is hierarchical" is in AC^0 (expression complexity)

Proof

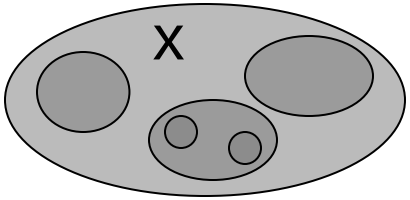
Hierarchical → PTIME

Proof

Hierarchical → PTIME

Case 1:

Q=



∀-Rule:

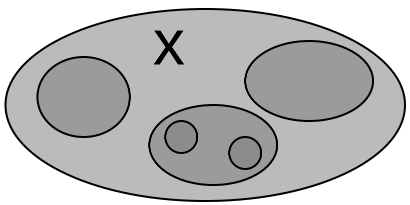
$P(\forall x\ Q) = \Pi_a P(Q[a/x])$

Proof

Hierarchical → PTIME

Case 1:

Q=

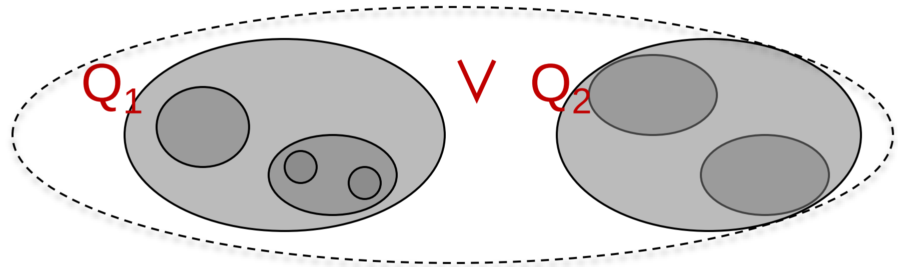


∀-Rule:

$$P(\forall x\ Q) = \Pi_a P(Q[a/x])$$

Case 2:

Q=



∨−Rule:

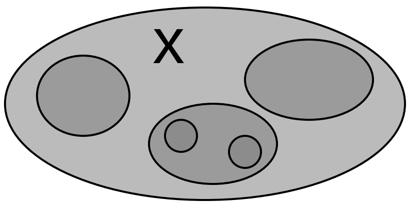
$$P(Q) = 1 - (1 - P(Q_1))(1 - P(Q_2))$$

Proof

Hierarchical → PTIME

Case 1:

Q=

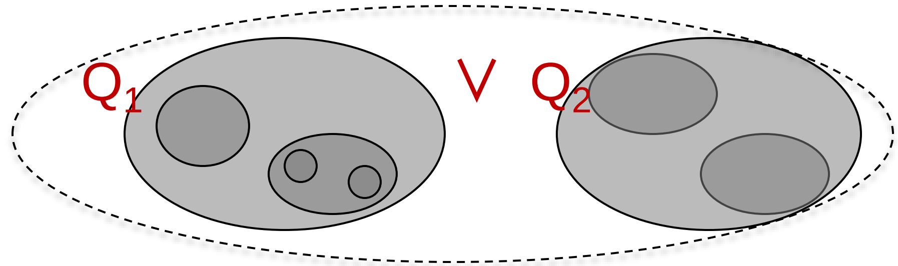


∀-Rule:

$P(\forall x\, Q) = \Pi_a P(Q[a/x])$

Case 2:

Q=

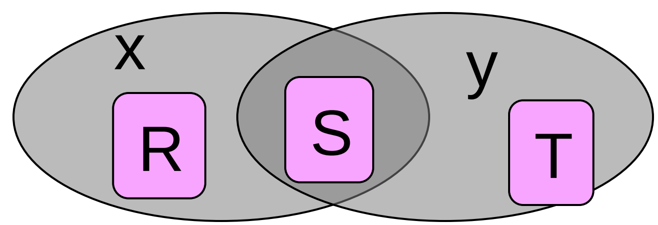


∨−Rule:

$P(Q) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

Non-hierarchical → #P-hard

Reduction from H_0:



$Q = \ldots R(x, \ldots) \vee S(x,y,\ldots) \vee T(y,\ldots), \ldots$

The Big Dichotomy Theorem

- For Positive CNF-FO the rules are *not* complete as stated!
- Instead we will revise inclusion/exclusion
- After the revision, the rules are complete
- We start with some non-liftable queries…

The Non-liftable Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

The Non-liftable Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

$H_2 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee T(y_2)]$

The Non-liftable Queries H_k

$H_0 = R(x) \lor S(x,y) \lor T(y)$

$H_1 = [R(x_0) \lor S(x_0,y_0)] \land [S(x_1,y_1) \lor T(y_1)]$

$H_2 = [R(x_0) \lor S_1(x_0,y_0)] \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \land [S_2(x_2,y_2) \lor T(y_2)]$

$H_3 = [R(x_0) \lor S_1(x_0,y_0)] \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \land [S_2(x_2,y_2) \lor S_3(x_2,y_2)] \land [S_3(x_3,y_3) \lor T(y_3)]$

. . .

The Non-liftable Queries H_k

$H_0 = R(x) \vee S(x,y) \vee T(y)$

$H_1 = [R(x_0) \vee S(x_0,y_0)] \wedge [S(x_1,y_1) \vee T(y_1)]$

$H_2 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee T(y_2)]$

$H_3 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \wedge [S_3(x_3,y_3) \vee T(y_3)]$

. . .

Theorem. [Dalvi'12] For every k, the query H_k is #P-hard

So far, not very interesting…

$H_3 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \wedge [S_3(x_3,y_3) \vee T(y_3)]$

The Query Q_W

Q_W is a Boolean combination of clauses in H_3

$Q_W =$

$[\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_2 \forall y_2 (S_2(x_2,y_2) \vee S_3(x_2,y_2))]$ /* Q_1 */

$\vee\ [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3)]$ /* Q_2 */

$\vee\ [\forall x_1 \forall y_1 (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))]$ /* Q_3 */

$H_3 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \wedge [S_3(x_3,y_3) \vee T(y_3)]$

Q_W is a Boolean combination of clauses in H_3

The Query Q_W

$Q_W =$

$[\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_2 \forall y_2 (S_2(x_2,y_2) \vee S_3(x_2,y_2))]$ /* Q_1 */

$\vee\ [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))]$ /* Q_2 */

$\vee\ [\forall x_1 \forall y_1 (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))]$ /* Q_3 */

Q_W is liftable BUT we need to use cancellations!

$H_3 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \wedge [S_3(x_3,y_3) \vee T(y_3)]$

Q_W is a Boolean combination of clauses in H_3

The Query Q_W

$$
\begin{aligned}
Q_W = \; & [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_2 \forall y_2 (S_2(x_2,y_2) \vee S_3(x_2,y_2))] \quad /^*\ Q_1\ ^*/ \\
\vee \; & [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))] \quad /^*\ Q_2\ ^*/ \\
\vee \; & [\forall x_1 \forall y_1 (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))] \quad /^*\ Q_3\ ^*/
\end{aligned}
$$

Q_W is liftable BUT we need to use cancellations!

Liftable

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \wedge Q_2) - P(Q_2 \wedge Q_3) - P(Q_1 \wedge Q_3) \\
& + P(Q_1 \wedge Q_2 \wedge Q_3)
\end{aligned}
$$

$= H_3$ (hard !)

Also $= H_3$

$H_3 = [R(x_0) \vee S_1(x_0,y_0)] \wedge [S_1(x_1,y_1) \vee S_2(x_1,y_1)] \wedge [S_2(x_2,y_2) \vee S_3(x_2,y_2)] \wedge [S_3(x_3,y_3) \vee T(y_3)]$

Q_W is a Boolean combination of clauses in H_3

The Query Q_W

$$
\begin{aligned}
Q_W = \; & [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_2 \forall y_2 (S_2(x_2,y_2) \vee S_3(x_2,y_2))] \quad /^* \; Q_1 \; ^*/ \\
\vee \; & [\forall x_0 \forall y_0 (R(x_0) \vee S_1(x_0,y_0)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))] \quad /^* \; Q_2 \; ^*/ \\
\vee \; & [\forall x_1 \forall y_1 (S_1(x_1,y_1) \vee S_2(x_1,y_1)) \wedge \forall x_3 \forall y_3 (S_3(x_3,y_3) \vee T(y_3))] \quad /^* \; Q_3 \; ^*/
\end{aligned}
$$

Q_W is liftable BUT we need to use cancellations!

Liftable

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \wedge Q_2) - P(Q_2 \wedge Q_3) - P(Q_1 \wedge Q_3) \\
& + P(Q_1 \wedge Q_2 \wedge Q_3)
\end{aligned}
$$

$= H_3$ (hard !)

Also $= H_3$

The two hard queries cancel out, and what remains is Liftable

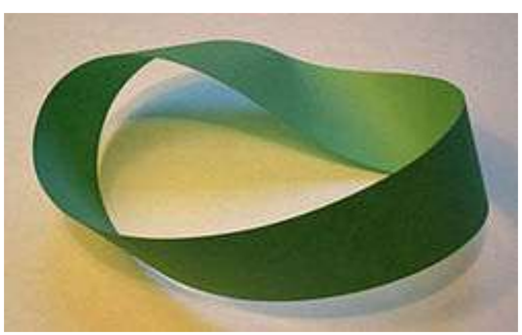
Cancellations?

- Cancellations in the inclusion/exclusion formula are critical! If we fail to do them, then the rules get stuck

- The mathematical concept that explains which terms cancel out is the Mobius' function (next)

August Ferdinand Möbius
1790-1868

- Möbius strip
- Möbius function μ in number theory
- Generalized to lattices [Stanley'97]
- And to lifted inference!



The Lattice of a Query

Definition. The lattice of $Q = Q_1 \wedge Q_2 \wedge \ldots$ is:

- Elements are terms of inclusion/exclusion;
- Order is logical implication

$\hat{1}$

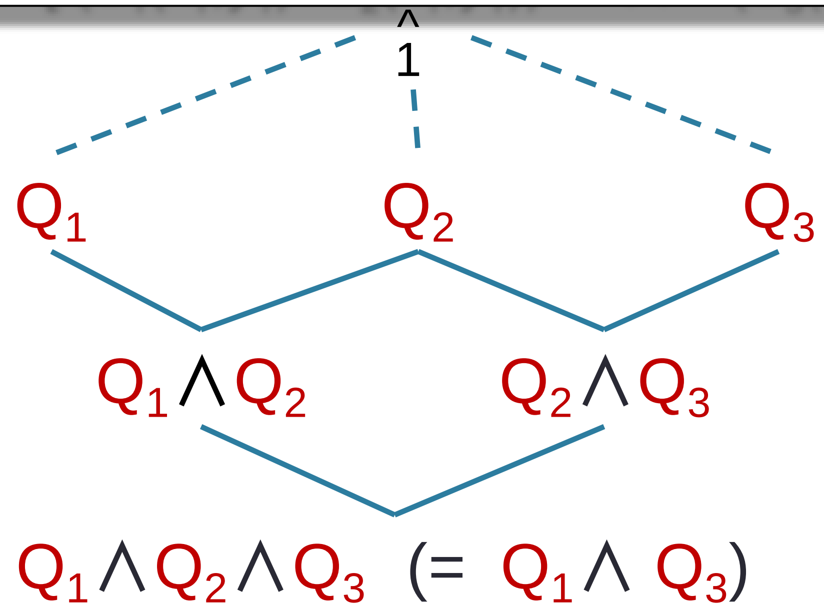
$\hat{1}$

The Lattice of a Query

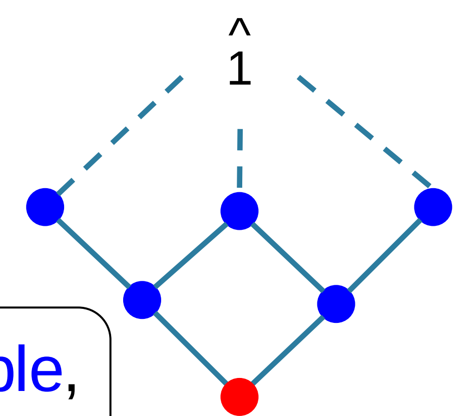
Definition. The lattice of $Q = Q_1 \wedge Q_2 \wedge \ldots$ is:

- Elements are terms of inclusion/exclusion;
- Order is logical implication

$$
\begin{aligned}
Q_W = \\
& [\,(R(x_0) \vee S_1(x_0,y_0)) && \wedge && (S_2(x_2,y_2) \vee S_3(x_2,y_2))] && /^{*}\ Q_1\ ^{*}/ \\
\vee\ & [\,(R(x_0) \vee S_1(x_0,y_0)) && \wedge && (S_3(x_3,y_3) \vee T(y_3)] && /^{*}\ Q_2\ ^{*}/ \\
\vee\ & [\,(S_1(x_1,y_1) \vee S_2(x_1,y_1)) && \wedge && (S_3(x_3,y_3) \vee T(y_3))] && /^{*}\ Q_3\ ^{*}/
\end{aligned}
$$



Nodes • Liftable,
Nodes • #P hard.

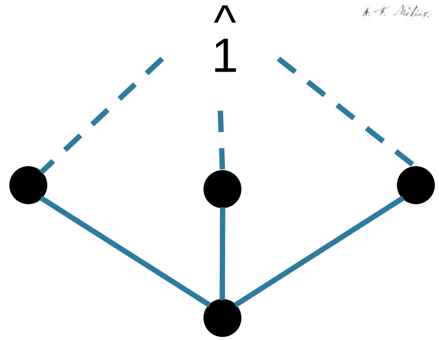


The Möbius' Function

Def. The Möbius function:

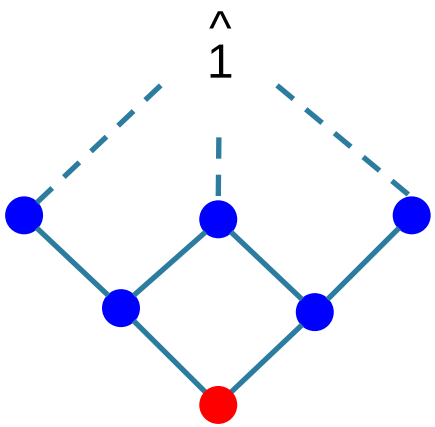
$\mu(\hat{1},\hat{1}) = 1$

$\mu(u,\hat{1}) = -\Sigma_{u < v \leq \hat{1}}\ \mu(v,\hat{1})$



Möbius' Inversion Formula:

$$P(Q) = -\Sigma_{Q_i < \hat{1}}\ \mu(Q_i,\hat{1})\ P(Q_i)$$

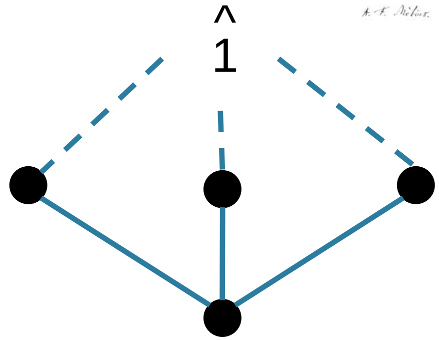


The Möbius' Function

Def. The Möbius function:

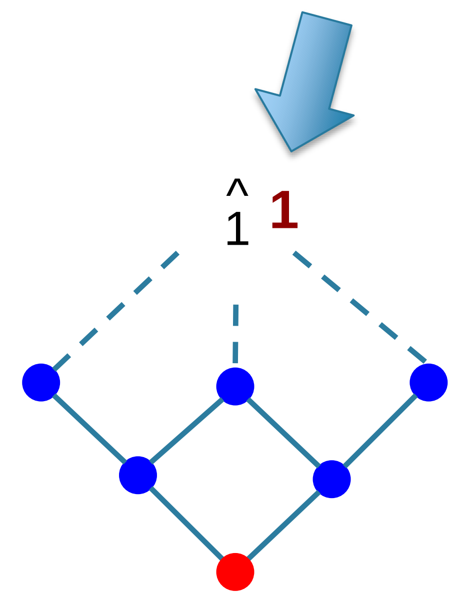
$\mu(\hat{1},\hat{1}) = 1$

$\mu(u,\hat{1}) = -\sum_{u < v \leq \hat{1}} \mu(v,\hat{1})$



Möbius' Inversion Formula:

$$P(Q) = -\sum_{Q_i < \hat{1}} \mu(Q_i,\hat{1})\, P(Q_i)$$

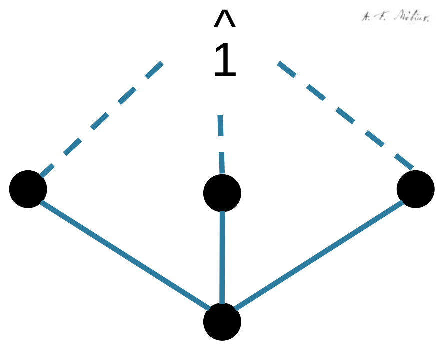


The Möbius' Function

Def. The Möbius function:

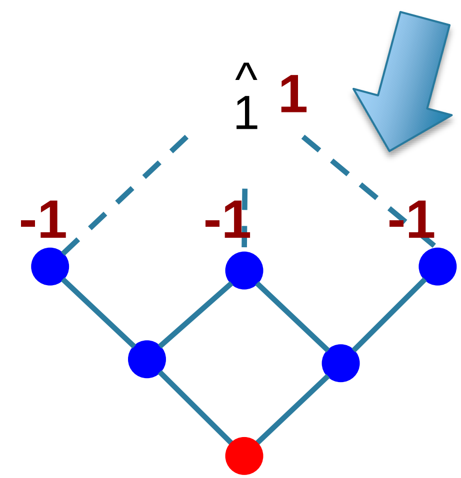
$\mu(\hat{1}, \hat{1}) = 1$

$\mu(u, \hat{1}) = - \Sigma_{u < v \leq \hat{1}} \; \mu(v, \hat{1})$



Möbius' Inversion Formula:

$P(Q) = - \Sigma_{Qi < \hat{1}} \; \mu(Q_i, \hat{1}) \; P(Q_i)$

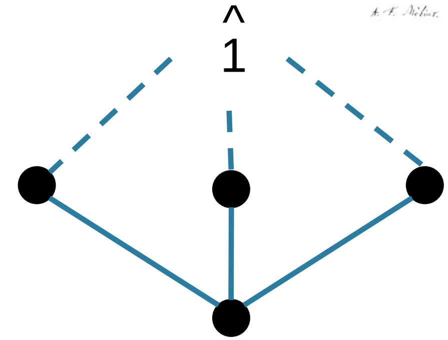


The Möbius' Function

Def. The Möbius function:

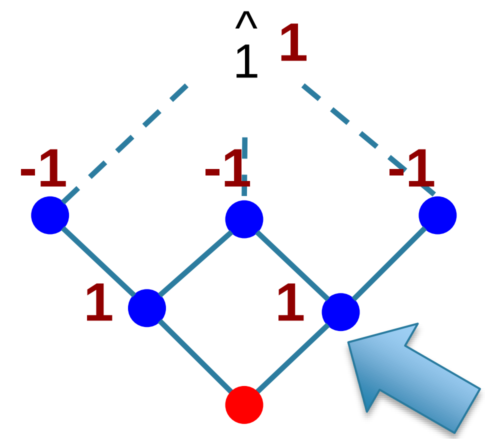
$\mu(\hat{1},\hat{1}) = 1$

$\mu(u,\hat{1}) = -\sum_{u < v \leq \hat{1}} \mu(v,\hat{1})$



Möbius' Inversion Formula:

$$P(Q) = -\sum_{Q_i < \hat{1}} \mu(Q_i,\hat{1})\, P(Q_i)$$

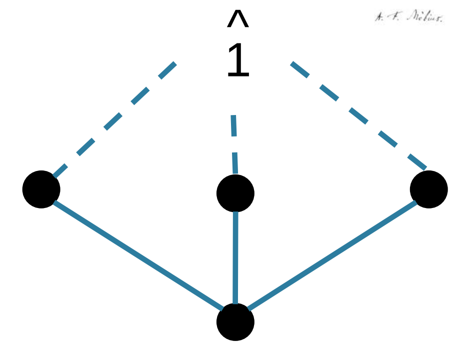


The Möbius' Function

Def. The Möbius function:

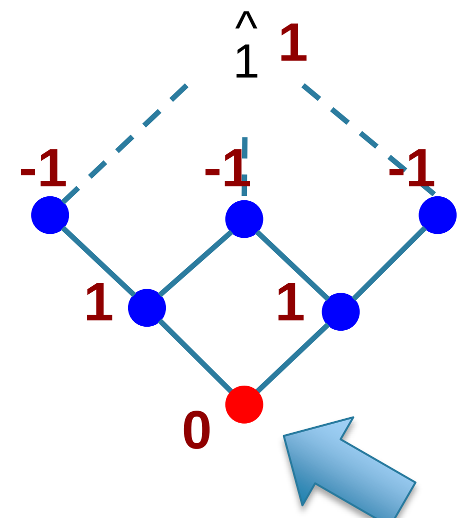
$\mu(\hat{1}, \hat{1}) = 1$

$\mu(u, \hat{1}) = - \Sigma_{u < v \leq \hat{1}} \mu(v, \hat{1})$



Möbius' Inversion Formula:

$$P(Q) = - \Sigma_{Qi < \hat{1}} \mu(Q_i, \hat{1}) P(Q_i)$$

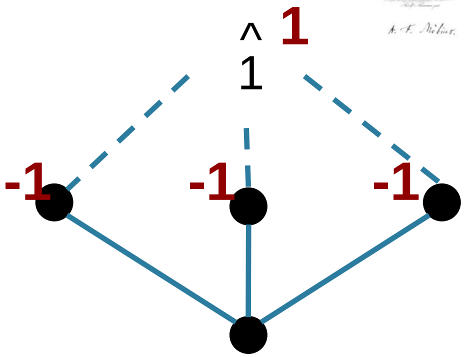


The Möbius' Function

Def. The Möbius function:

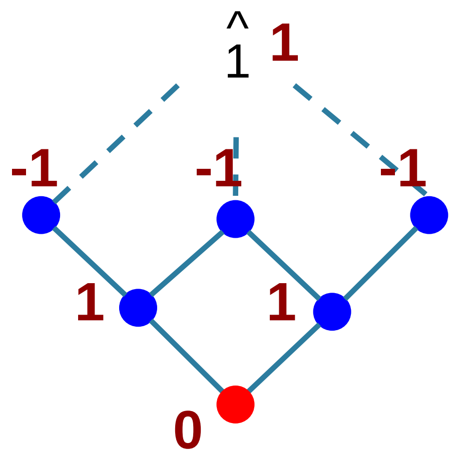
$\mu(\hat{1},\hat{1}) = 1$

$\mu(u,\hat{1}) = -\sum_{u < v \leq \hat{1}} \mu(v,\hat{1})$



Möbius' Inversion Formula:

$P(Q) = -\sum_{Q_i < \hat{1}} \mu(Q_i,\hat{1})\, P(Q_i)$

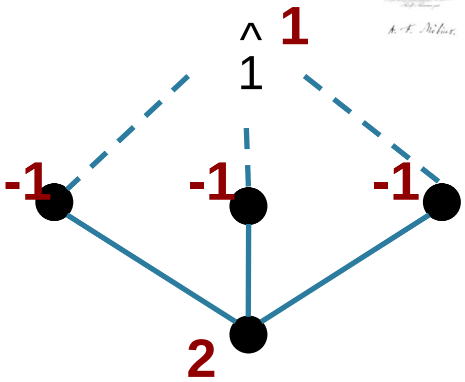


The Möbius' Function

Def. The Möbius function:

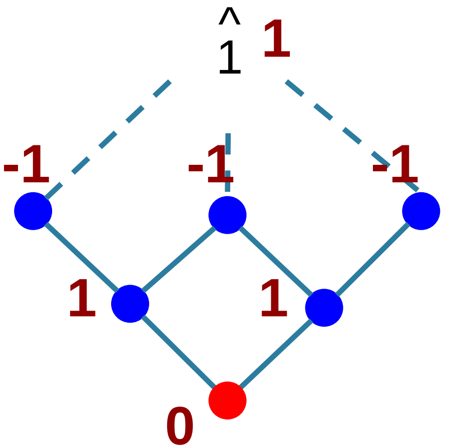
$\mu(\hat{1}, \hat{1}) = 1$

$\mu(u, \hat{1}) = - \Sigma_{u < v \leq \hat{1}} \mu(v, \hat{1})$



Möbius' Inversion Formula:

$$P(Q) = - \Sigma_{Q_i < \hat{1}} \mu(Q_i, \hat{1}) P(Q_i)$$

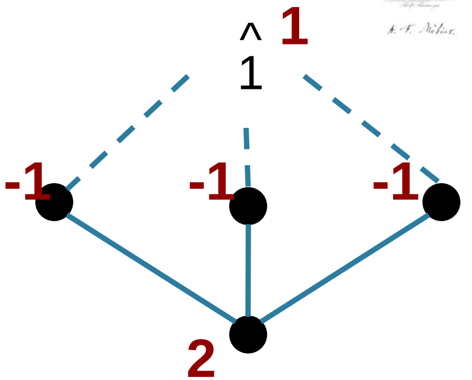


The Möbius' Function

Def. The Möbius function:

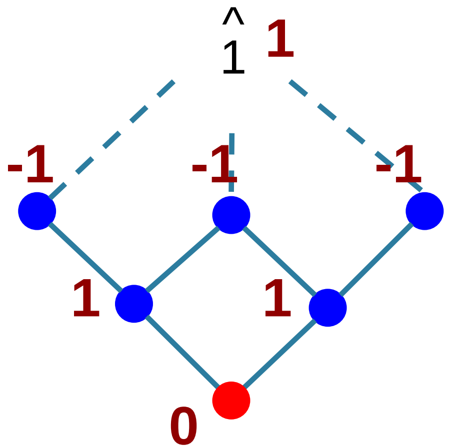
$\mu(\hat{1}, \hat{1}) = 1$

$\mu(u, \hat{1}) = - \Sigma_{u < v \leq \hat{1}} \mu(v, \hat{1})$



Möbius' Inversion Formula:

$$P(Q) = - \Sigma_{Qi < \hat{1}} \mu(Q_i, \hat{1}) P(Q_i)$$



<u>New Rule</u>

Inclusion/Exclusion

→ Mobius' Inversion Formula

The Dichotomy Theorem

Dichotomy Theorem [Dalvi'12] Fix a Positive-CNF Q.
1. If Q is liftable, then P(Q) is in PTIME (obviously)
2. If Q is not liftable, then P(Q) is #P-complete

Note 1: for the theorem to hold one must replace the inclusion/exclusion rule with the Mobius' rule

Note 2: Original formulation for UCQ; holds for Positive CNF-FO by duality.

Discussion

- This answers Question 1: lifted inference rules are complete for Positive CNF-FO

- Beyond Positive CNF-FO?
  – See poster on Saturday
  – Take-away: rules+resolution conjectured to be complete for CNF-FO; strong evidence that no complete rules exists for FO

2. Are lifted rules stronger than grounded?

Alternative to lifting:

1. Ground the FO sentence
2. Do WMC on the propositional formula

Symmetric WFOMC:

Grounded WMC does not use symmetries.

Query H_0 is:

- Liftable on symmetric,
- #P-hard on asymmetric

Asymmetric WFOMC

Query Q_W is in PTIME:

- DPLL-based search has exponential time
- Decision-DNNF have exponential size

Symmetric WFOMC

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

We have seen that H_0 is #P-hard (over asymmetric spaces!)
But over symmetric spaces it can be lifted:

$$P(H_0) = \sum_{k=0}^{n} \sum_{\ell=0}^{n} \binom{n}{k} \binom{n}{\ell} p_{\text{Smoker}}{}^{n-k} \cdot (1 - p_{\text{Smoker}})^{k} \cdot p_{\text{Jogger}}{}^{n-\ell} \cdot (1 - p_{\text{Jogger}})^{\ell} \cdot p_{\text{Friend}}{}^{k \cdot \ell}$$

Lifted inference is strictly more powerful than grounded inference

Symmetric WFOMC

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

We have seen that H_0 is #P-hard (over asymmetric spaces!)
But over symmetric spaces it can be lifted:

$$P(H_0) = \sum_{k=0}^{n} \sum_{\ell=0}^{n} \binom{n}{k} \binom{n}{\ell} p_{\text{Smoker}}{}^{n-k} \cdot (1 - p_{\text{Smoker}})^{k} \cdot p_{\text{Jogger}}{}^{n-\ell} \cdot (1 - p_{\text{Jogger}})^{\ell} \cdot p_{\text{Friend}}{}^{k\cdot\ell}$$

Lifted inference is strictly more powerful than grounded inference

---

Theorem [V.d.Broeck'14]: every query in FO^2 is liftable over symmetric spaces

FO^2 includes H_0, and some quite complex complex sentences like:

$Q = \forall x \forall y \forall z \forall u\ (\text{Friend}(x,y) \vee \text{Enemy}(y,z) \vee \text{Friend}(z,u) \vee \text{Enemy}(u,v))$
$\quad = \forall x \forall y\ (\text{Friend}(x,y) \vee \forall x\ (\text{Enemy}(y,x) \vee \forall y\ (\text{Friend}(x,y) \vee \forall x\ (\text{Enemy}(y,x)))))$

Asymmetric WFOMC

- Lifted inference does no longer have a fundamental reason to be stronger than grounded WMC
- However, we can prove that lifted inference is stronger than WMC algorithms used in practice today:
  - DPLL search (with caching; with components)
  - Decision-DNNF

Basic DPLL

```
// basic DPLL:
Function P(F):
        if F = false then return 0
        if F = true then return 1
        select a variable x, return
    ½ P(F_{X=0}) + ½ P(F_{X=1})
```

Davis, **P**utnam, **L**ogemann, **L**oveland
[Davis'60, '62]

Assume uniform distribution for simplicity

Basic DPLL

F: $(x \vee y) \wedge (x \vee \neg u \vee w) \wedge (\neg x \vee \neg u \vee w \vee z)$

// basic DPLL:
Function **P(F)**:
  if **F = false** then return **0**
  if **F = true** then return **1**
  select a variable **x**, return
  $\frac{1}{2}\, P(F_{X=0}) + \frac{1}{2}\, P(F_{X=1})$

Davis, **P**utnam, **L**ogemann, **L**oveland
[Davis'60, '62]

Assume uniform distribution for simplicity

Basic DPLL

F: $(x \vee y) \wedge (x \vee \neg u \vee w) \wedge (\neg x \vee \neg u \vee w \vee z)$

x

// basic DPLL:

Function **P(F)**:

if **F = false** then return **0**

if **F = true** then return **1**

select a variable **x**, return

$\frac{1}{2}$ **P(F$_{X=0}$)** + $\frac{1}{2}$ **P(F$_{X=1}$)**

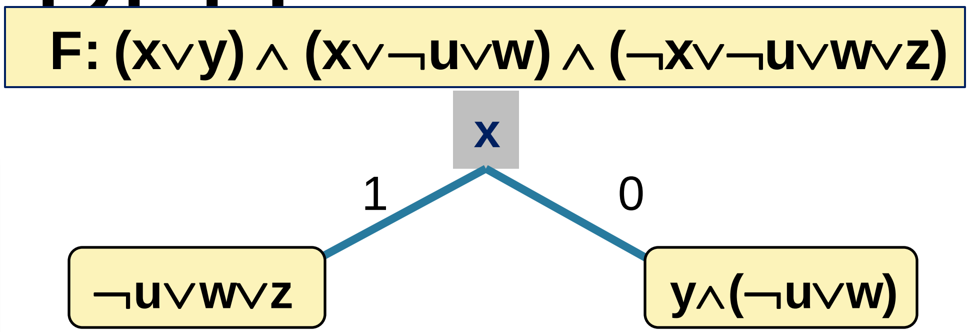
Davis, **P**utnam, **L**ogemann, **L**oveland
[Davis'60, '62]

Assume uniform distribution for simplicity

Basic DPLL

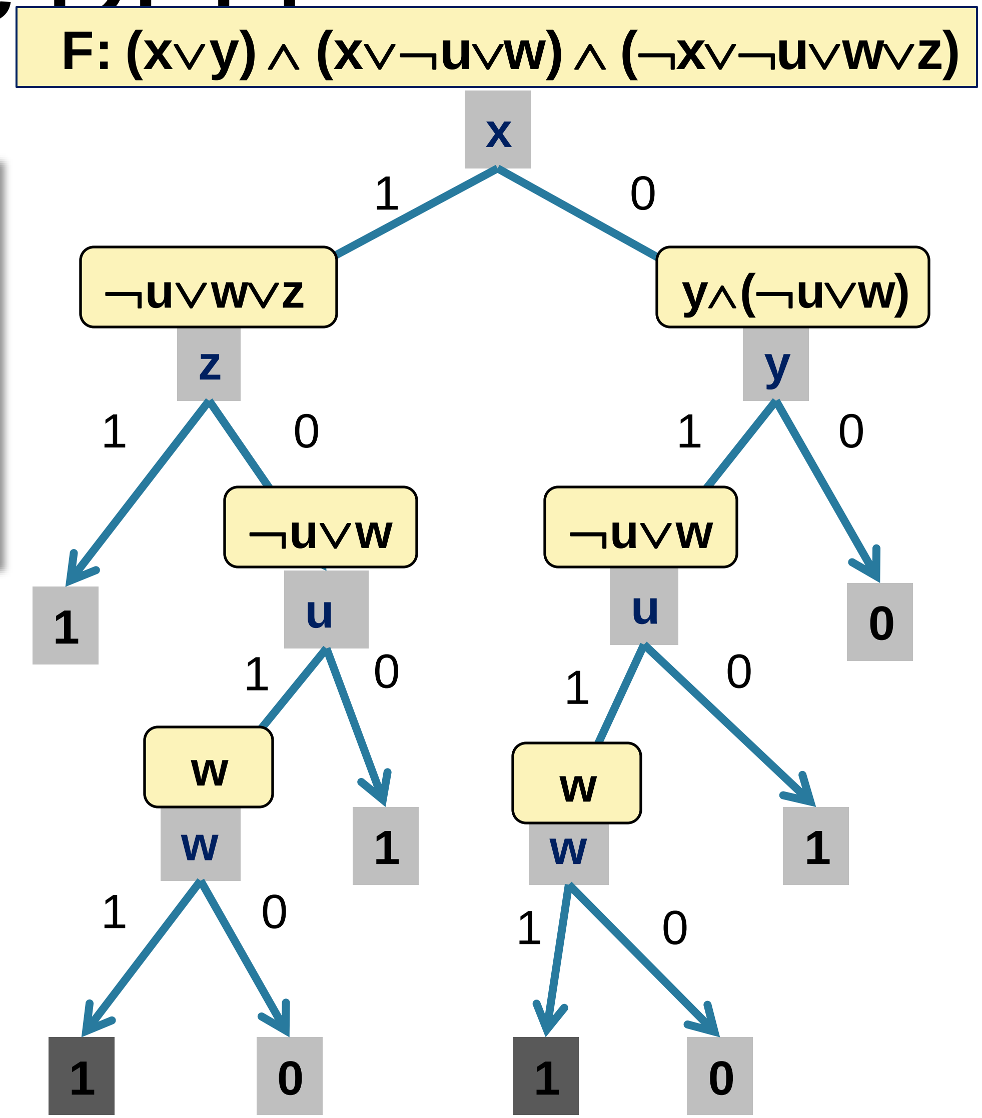
// basic DPLL:
Function **P(F)**:
    if **F = false** then return **0**
    if **F = true** then return **1**
    select a variable **x**, return
  $\frac{1}{2} P(F_{X=0}) + \frac{1}{2} P(F_{X=1})$

Davis, **P**utnam, **L**ogemann, **L**oveland
[Davis'60, '62]



Assume uniform distribution for simplicity

Basic DPLL



// basic DPLL:

Function **P(F)**:

if **F = false** then return **0**

if **F = true** then return **1**

select a variable **x**, return

½ $\mathbf{P(F_{X=0})}$ + ½ $\mathbf{P(F_{X=1})}$

Davis, **P**utnam, **L**ogemann, **L**oveland

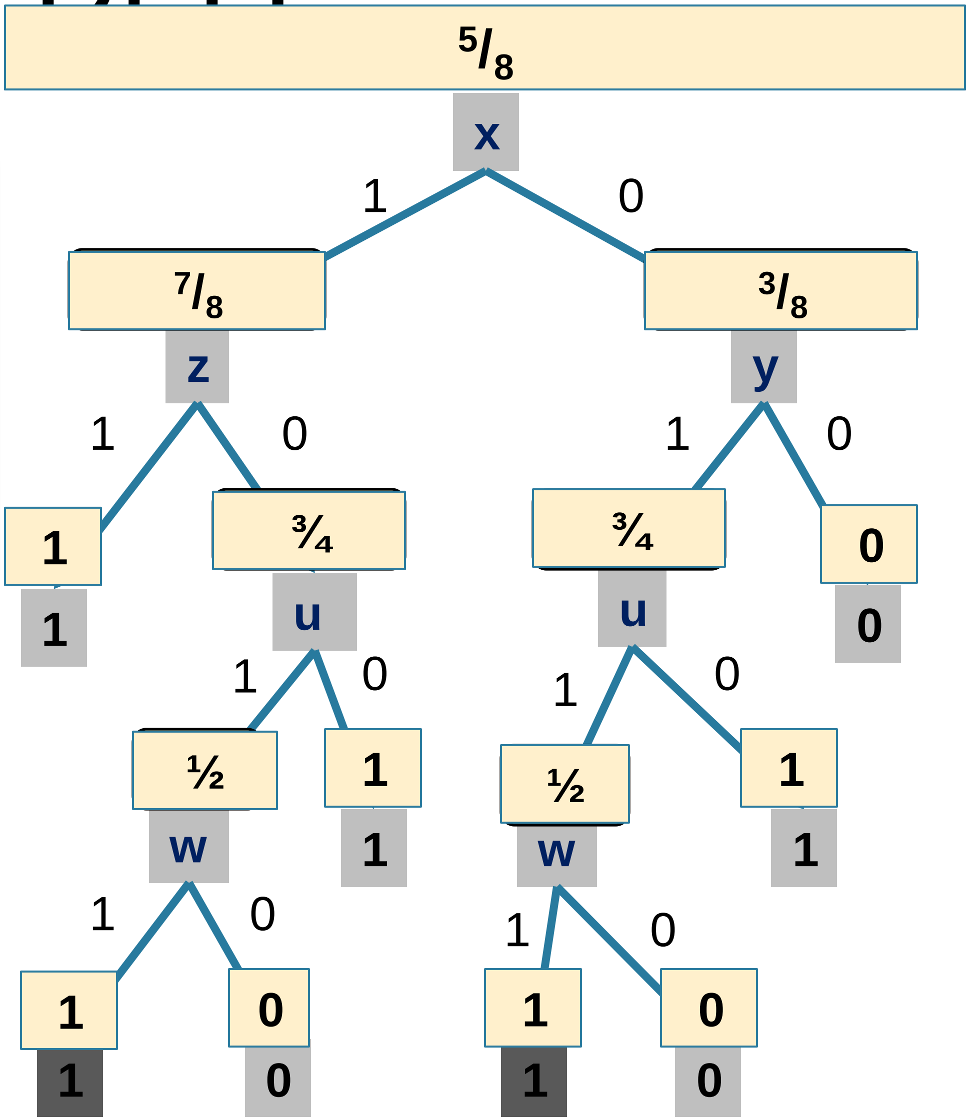
[Davis'60, '62]

Assume uniform distribution for simplicity

Basic DPLL

// basic DPLL:
Function **P(F)**:
if **F = false** then return **0**
if **F = true** then return **1**
select a variable **x**, return
½ **P($F_{X=0}$)** + ½ **P($F_{X=1}$)**

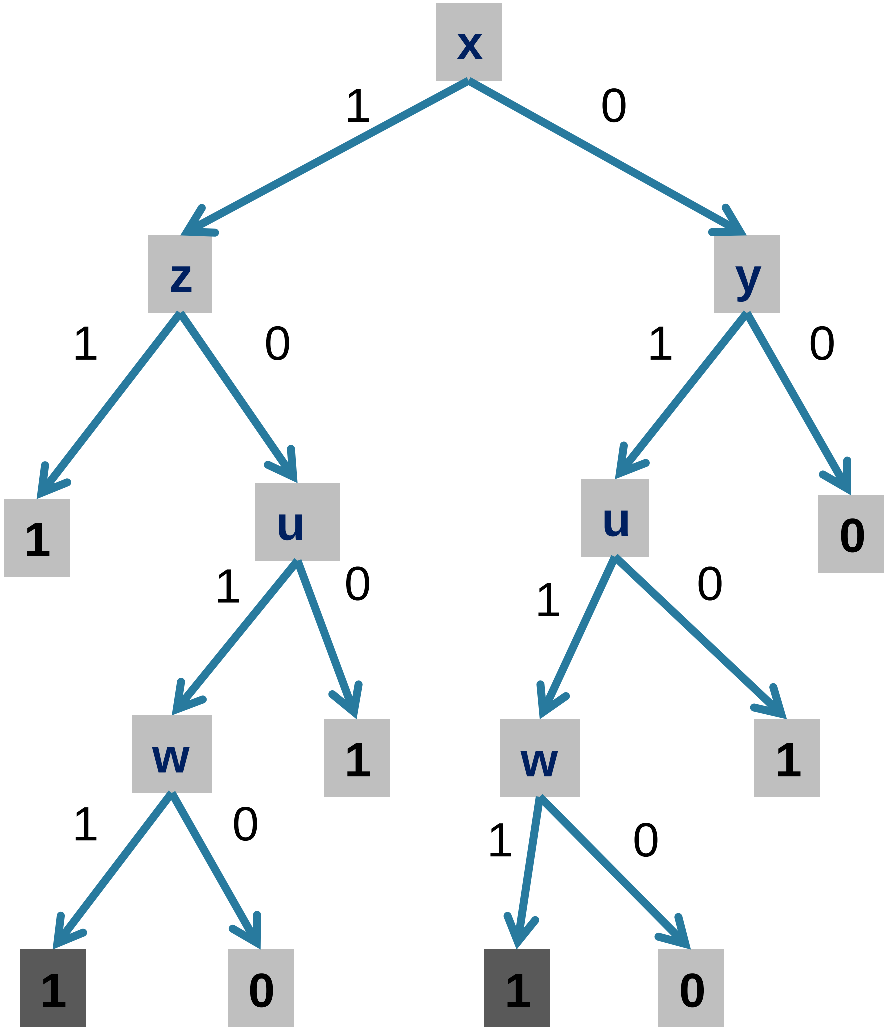
Davis, **P**utnam, **L**ogemann, **L**oveland
[Davis'60, '62]



Assume uniform distribution for simplicity

Basic DPLL

F: $(x \vee y) \wedge (x \vee \neg u \vee w) \wedge (\neg x \vee \neg u \vee w \vee z)$



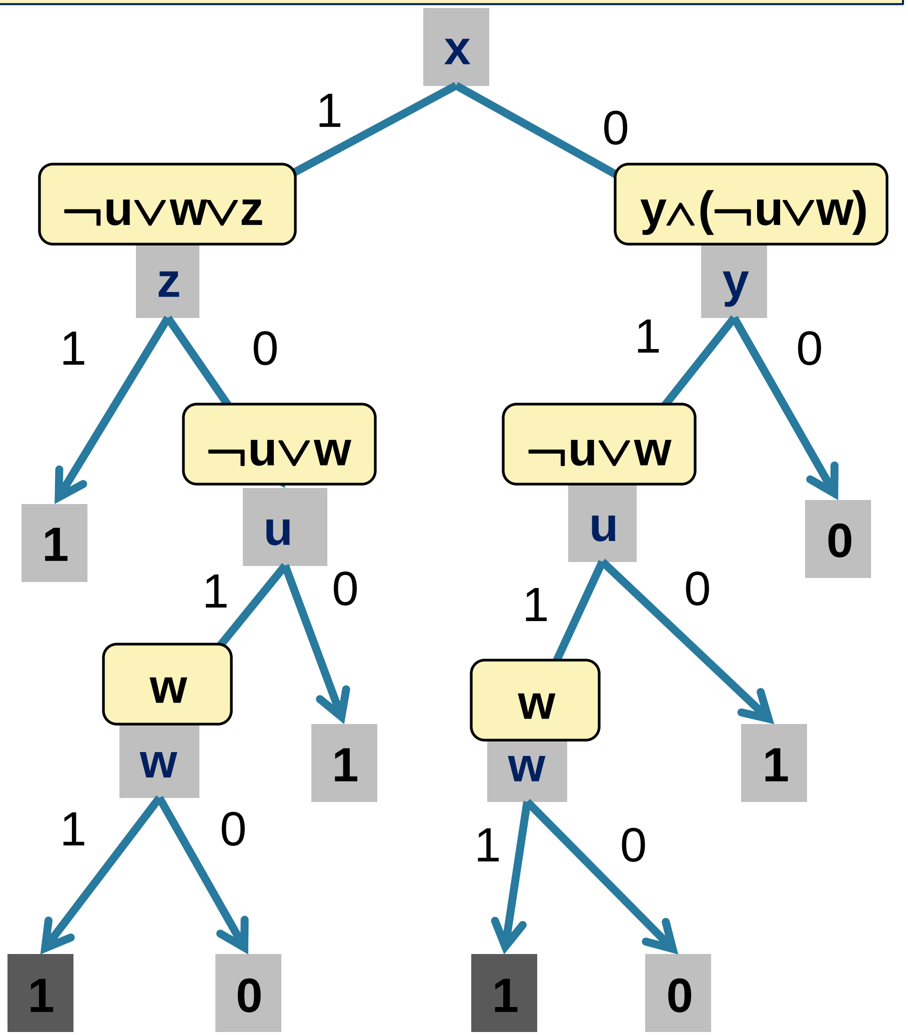
The trace is a **Decision-Tree** for **F**

Caching

// basic DPLL:
Function **P(F)**:
 if **F = false** then return **0**
 if **F = true** then return **1**
 select a variable **x**, return
 ½ $P(F_{X=0})$ + ½ $P(F_{X=1})$

// DPLL with caching:
Cache **F** and **P(F)**;
look it up before computing

F: (x∨y) ∧ (x∨¬u∨w) ∧ (¬x∨¬u∨w∨z)



Caching

// basic DPLL:

Function **P(F)**:

if **F = false** then return **0**

if **F = true** then return **1**

select a variable **x**, return

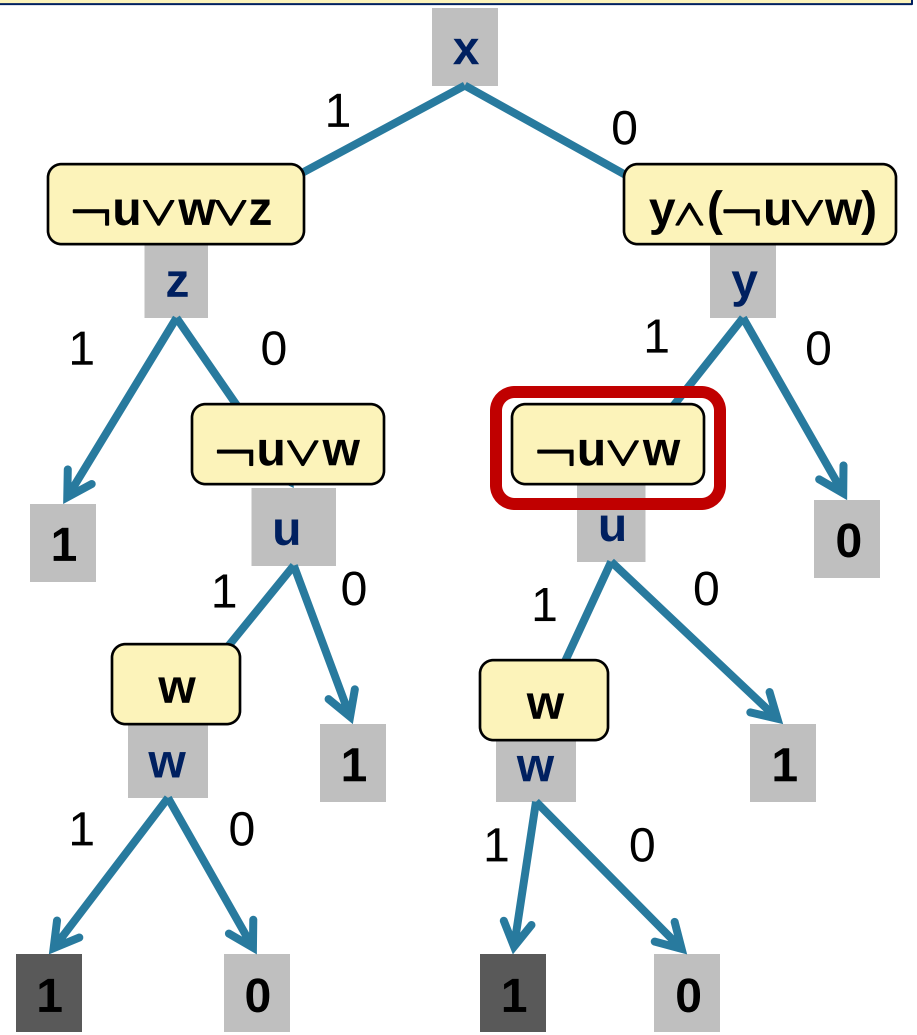
½ $P(F_{X=0})$ + ½ $P(F_{X=1})$

// DPLL with caching:

Cache **F** and **P(F)**;

look it up before computing

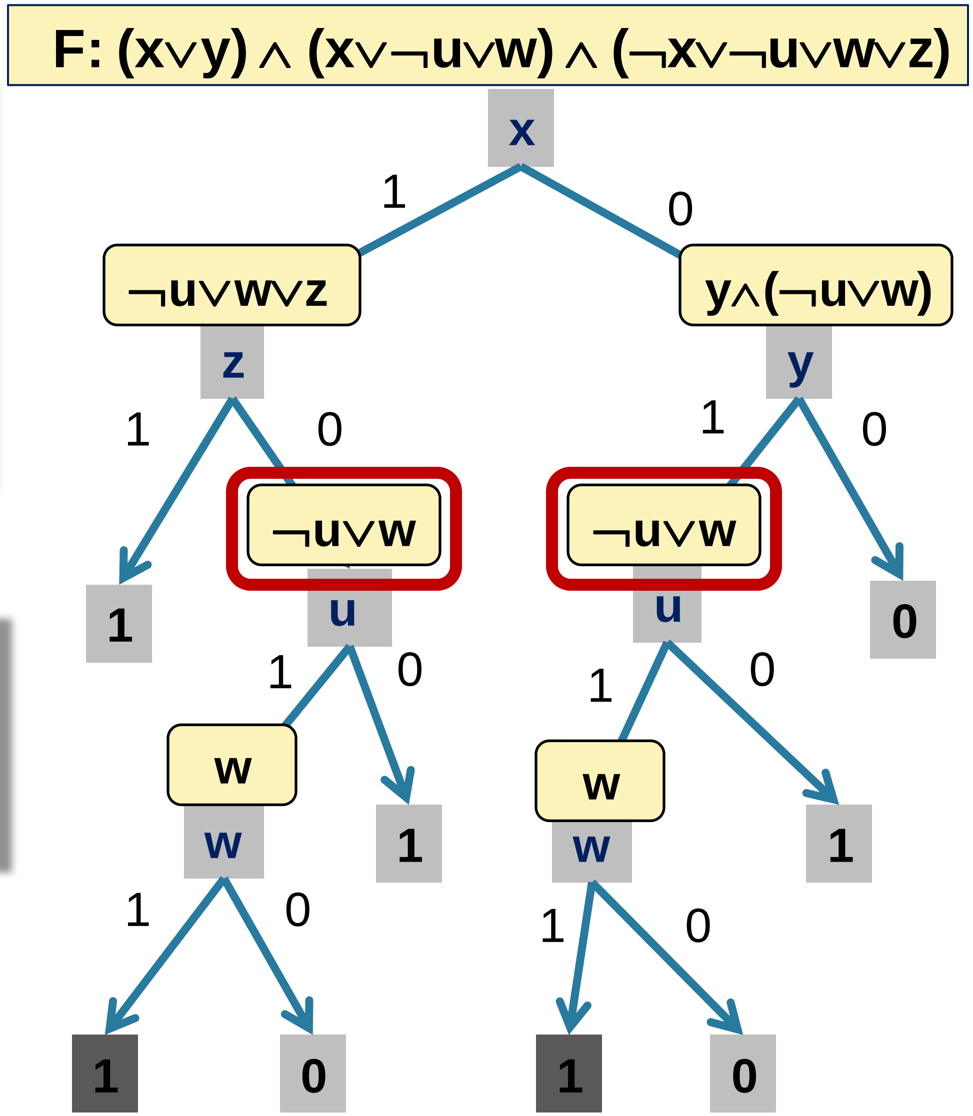
F: (x∨y) ∧ (x∨¬u∨w) ∧ (¬x∨¬u∨w∨z)



Caching

// basic DPLL:
Function **P(F)**:
if **F = false** then return **0**
if **F = true** then return **1**
select a variable **x**, return
½ $P(F_{X=0})$ + ½ $P(F_{X=1})$

// DPLL with caching:
Cache **F** and **P(F)**;
look it up before computing



Caching

// basic DPLL:

Function **P(F)**:

if **F = false** then return **0**

if **F = true** then return **1**

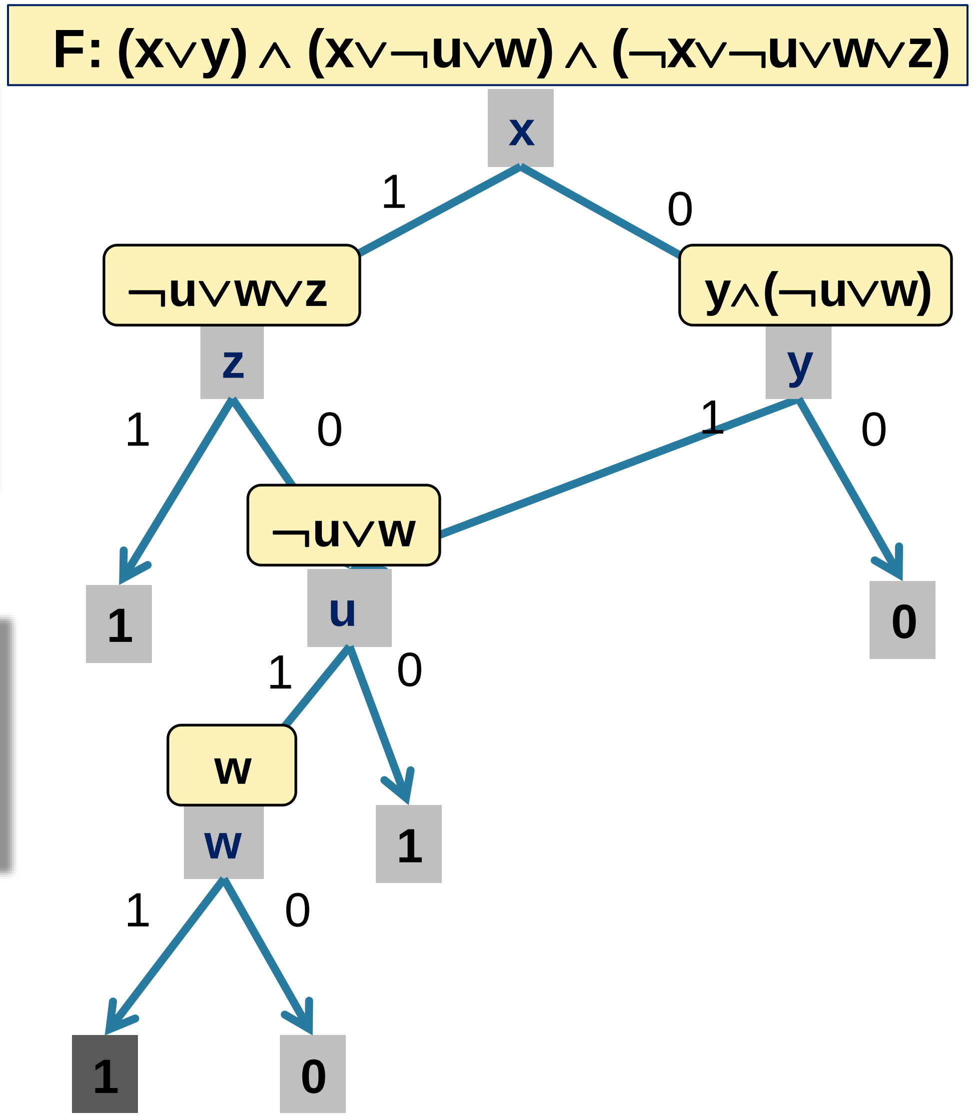
select a variable **x**, return

½ $P(F_{X=0})$ + ½ $P(F_{X=1})$

// DPLL with caching:

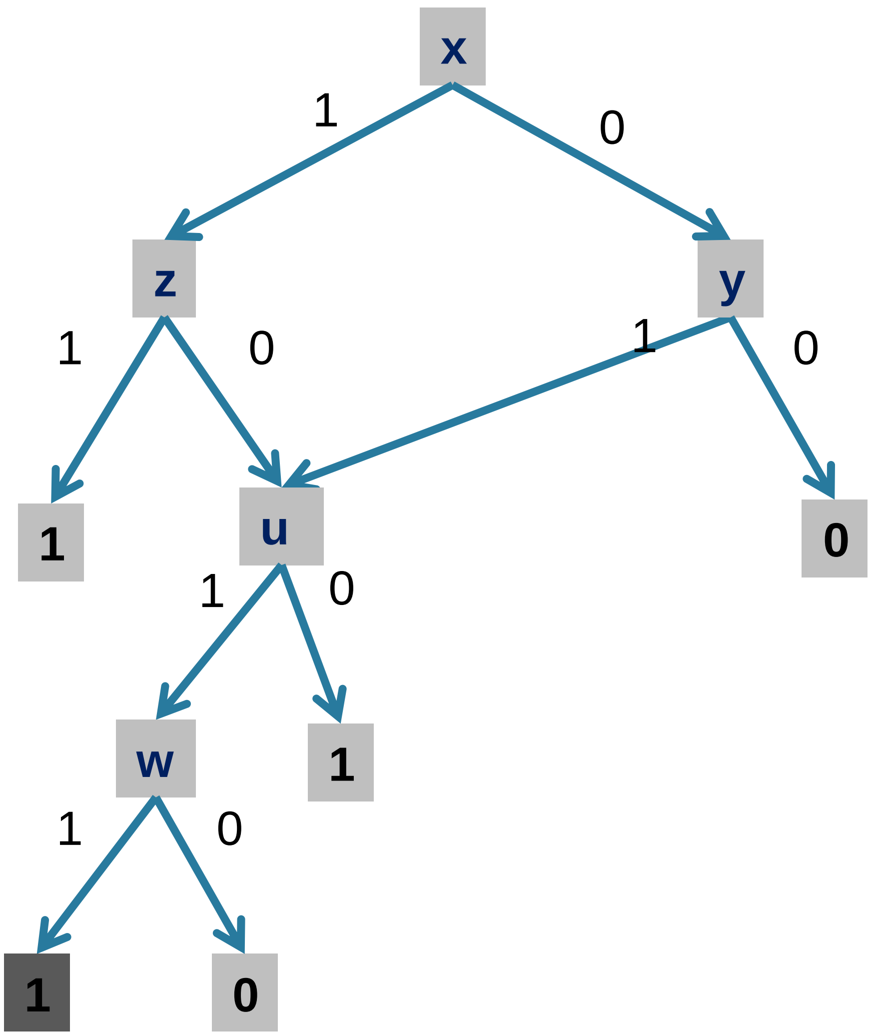
Cache **F** and **P(F)**;

look it up before computing



Caching & FBDDs

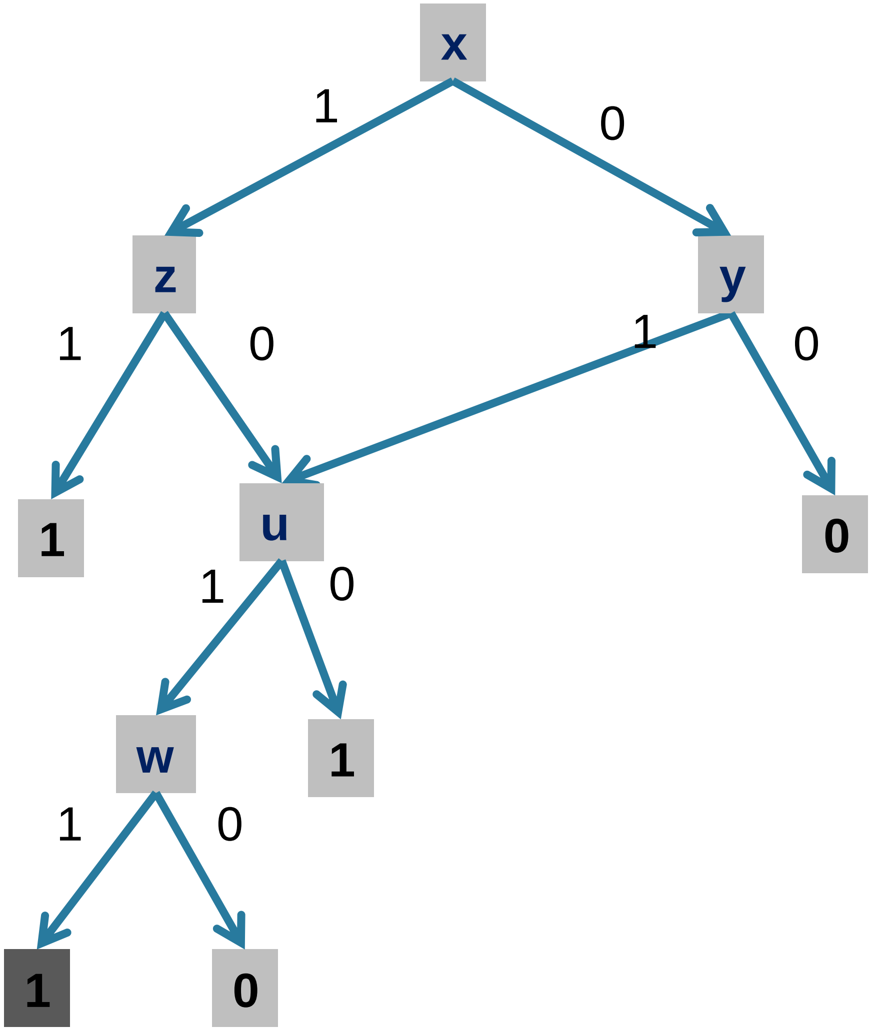
The trace is a decision-DAG for **F**



Caching & FBDDs

The trace is a decision-DAG for **F**

FBDD (Free Binary Decision Diagram)
or
ROBP (Read Once Branching Program)

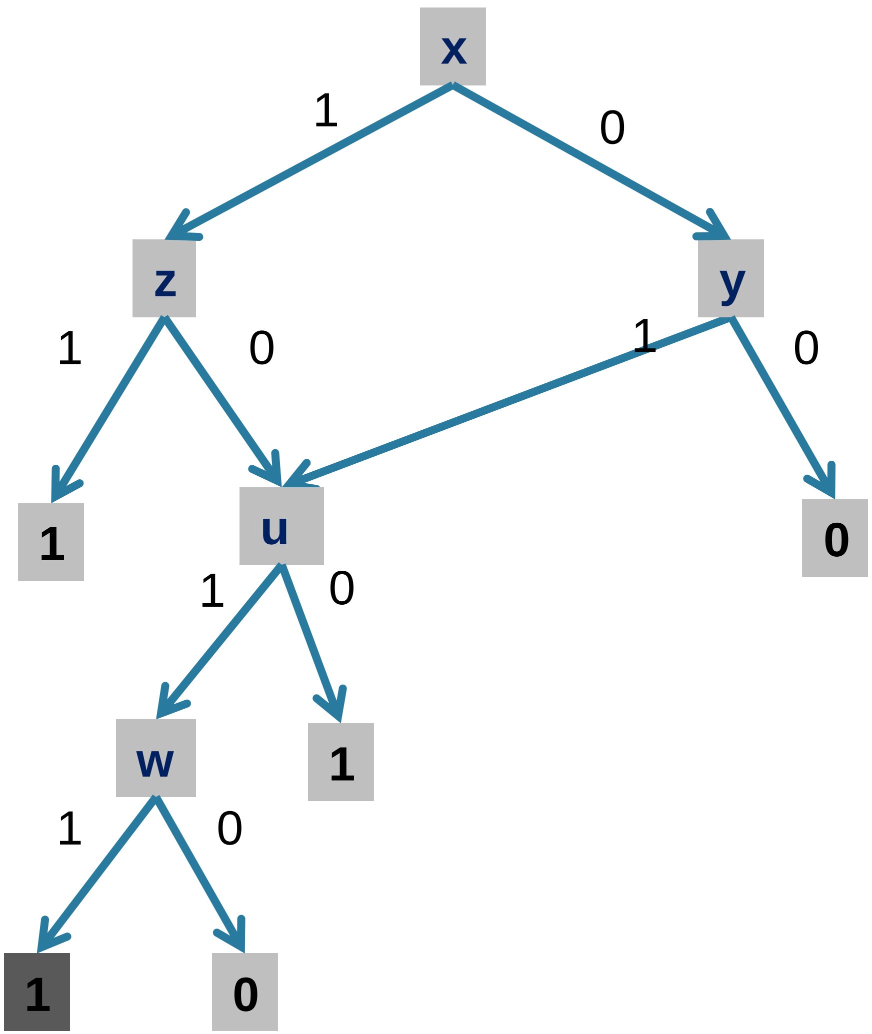


Caching & FBDDs

The trace is a decision-DAG for **F**

FBDD (Free Binary Decision Diagram)
or
ROBP (Read Once Branching Program)

• Every variable is tested at most once on any path



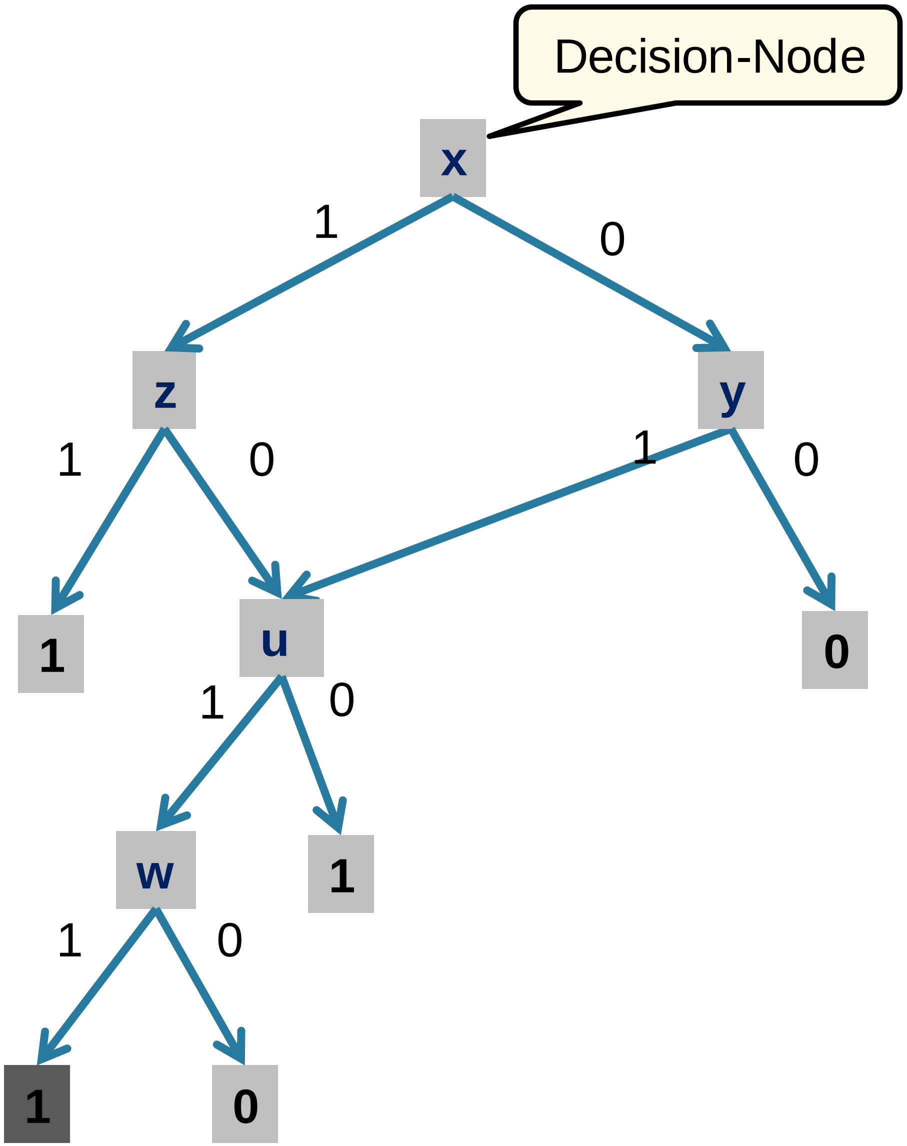
Caching & FBDDs

The trace is a decision-DAG for **F**

FBDD (Free Binary Decision Diagram)
or
ROBP (Read Once Branching Program)

• Every variable is tested at most once on any path

• All internal nodes are decision-nodes

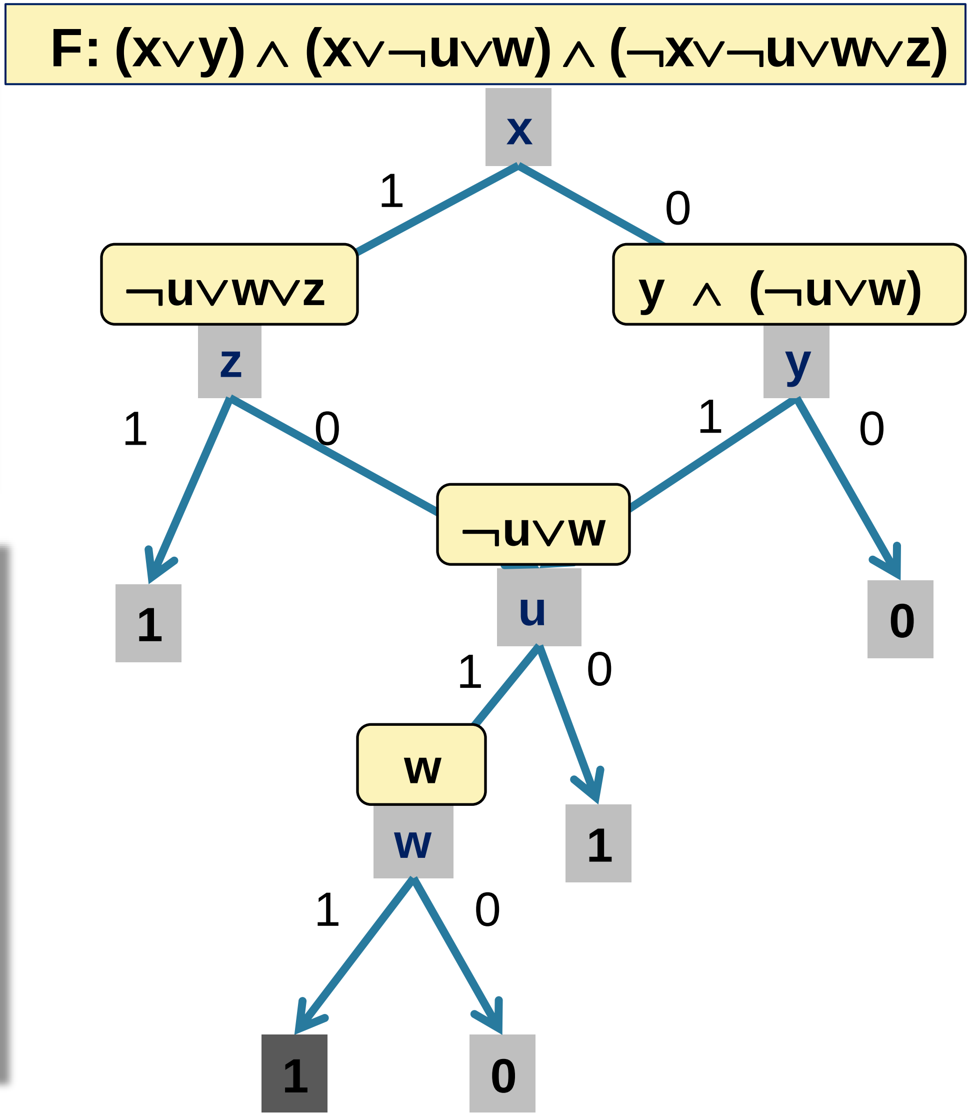


Component Analysis

// basic DPLL:
Function **P(F)**:
if **F = false** then return **0**
if **F = true** then return **1**
select a variable **x**, return
½ $P(F_{X=0})$ + ½ $P(F_{X=1})$

// DPLL with component analysis (and caching):

if **F = G ∧ H**
where **G** and **H** have disjoint set of variables
P(F) = P(G) × P(H)



Component Analysis

// basic DPLL:

Function **P(F)**:

if **F = false** then return **0**

if **F = true** then return **1**

select a variable **x**, return

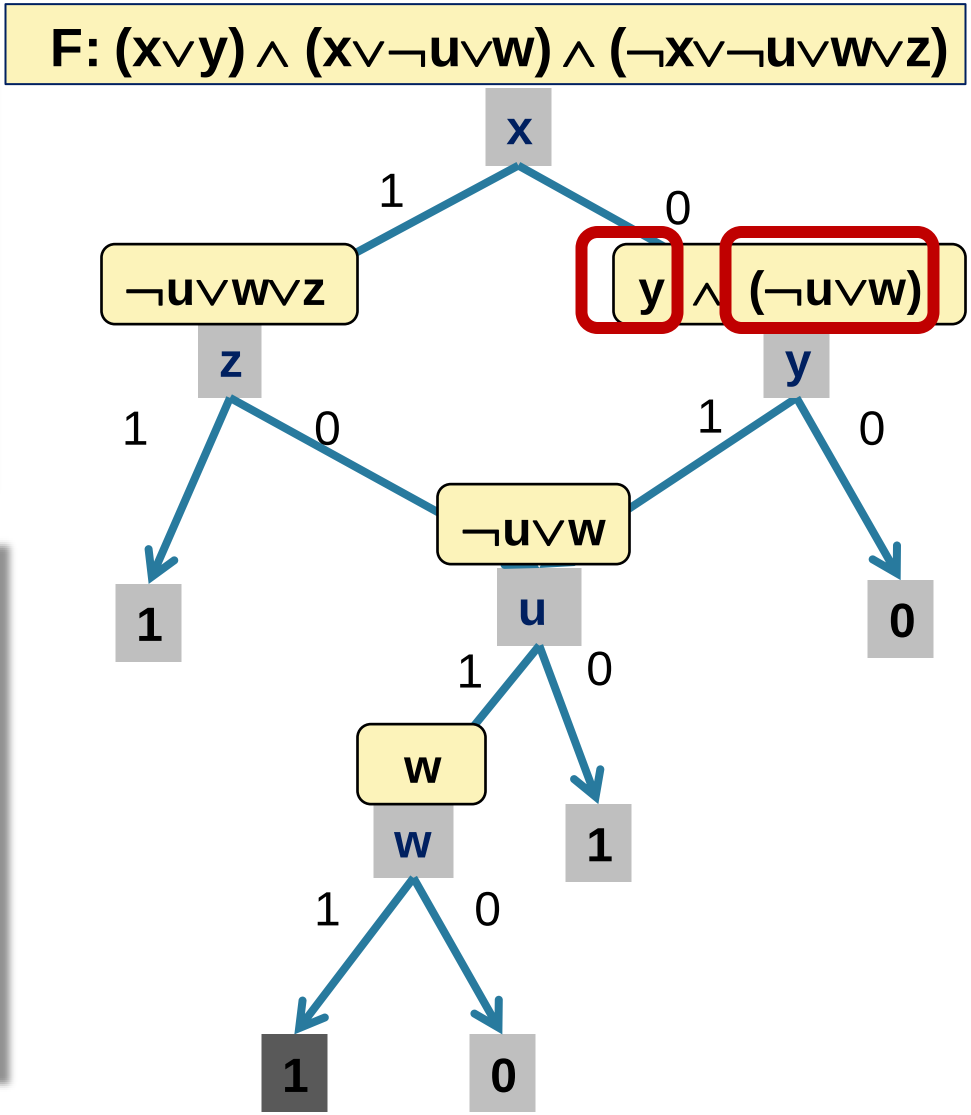
$\frac{1}{2}\, P(F_{X=0}) + \frac{1}{2}\, P(F_{X=1})$

// DPLL with component analysis (and caching):

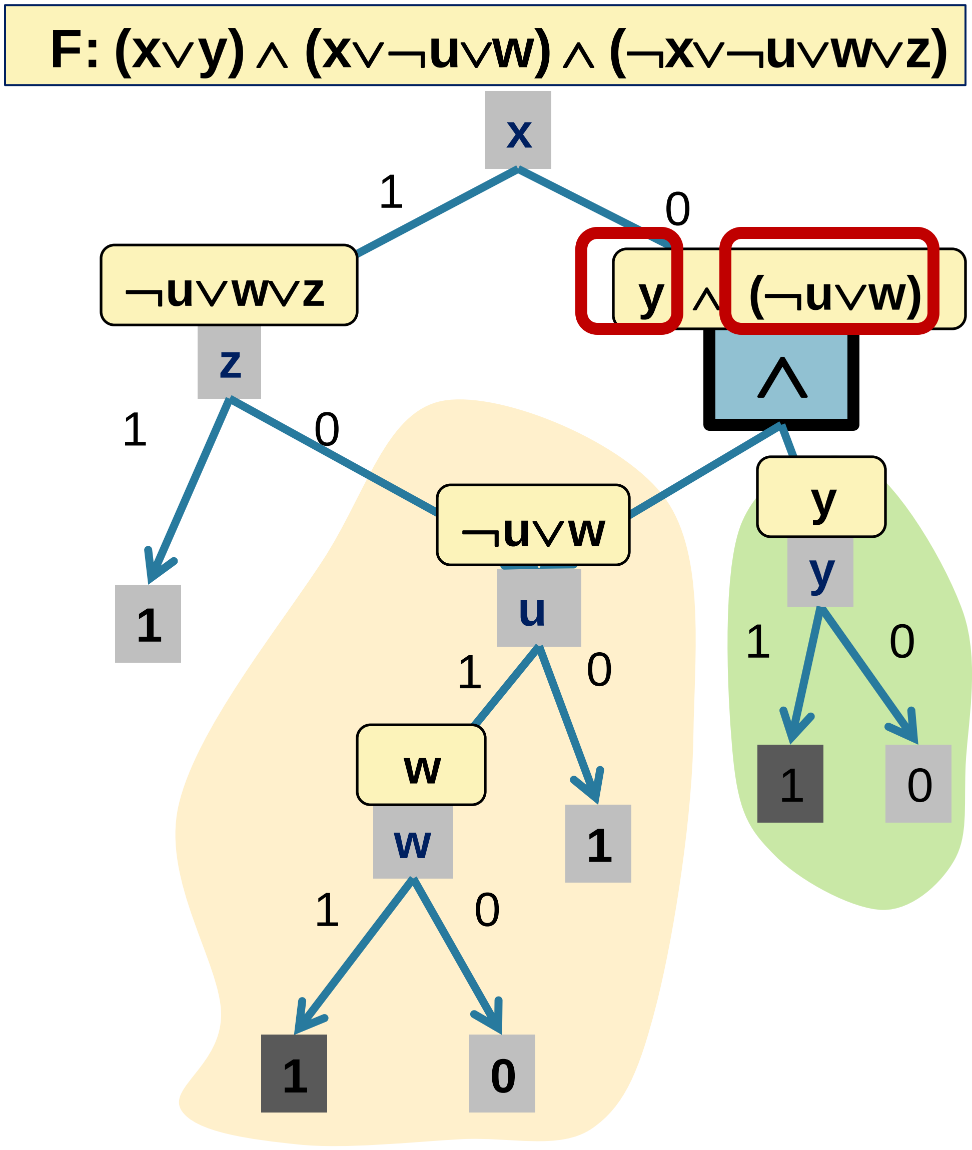
if **F = G ∧ H**

where **G** and **H** have disjoint set of variables

P(F) = P(G) × P(H)



Components & Decision-DNNF

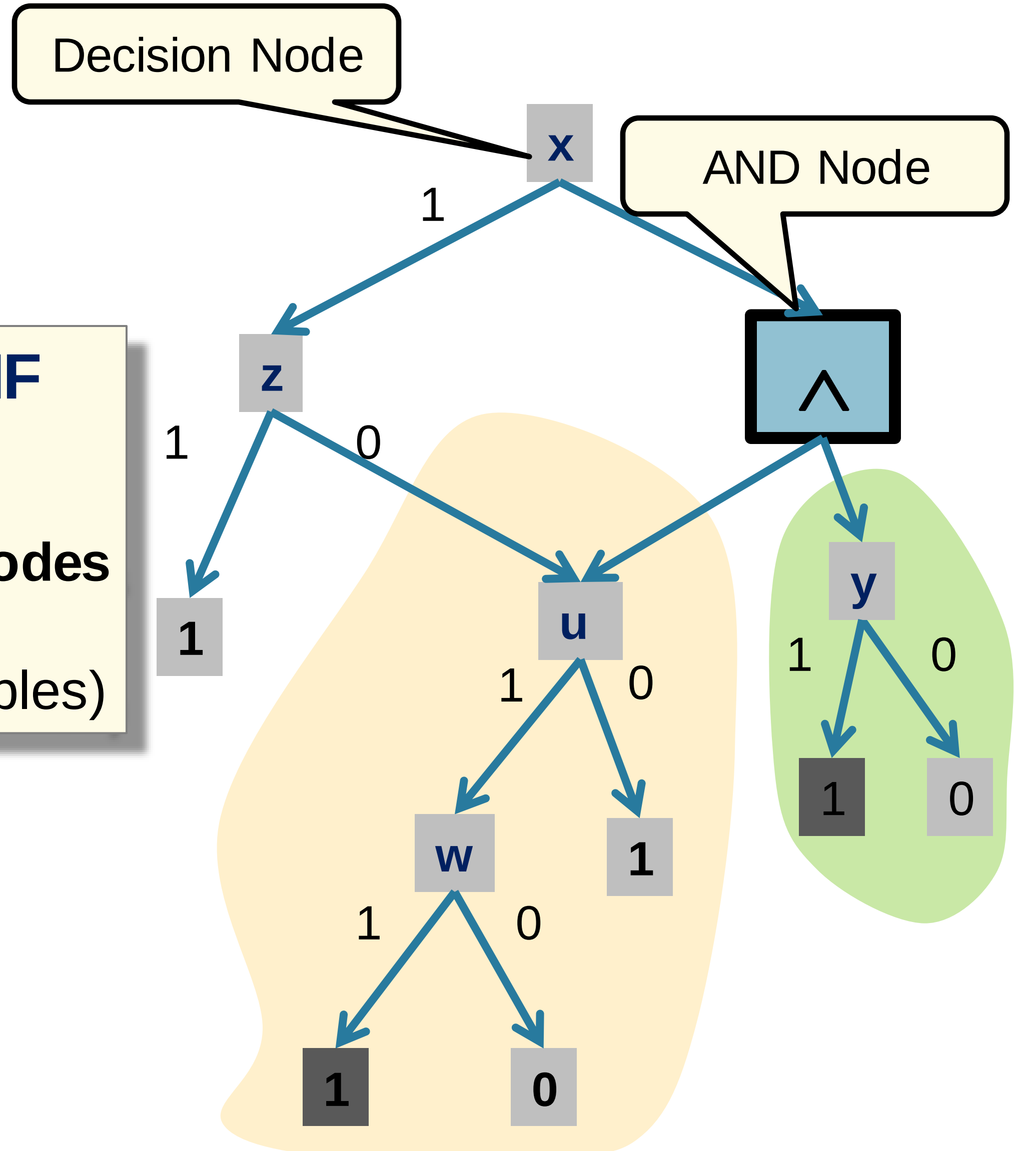


Components & Decision-DNNF

The trace is a **Decision-DNNF**
[Huang'05, '07]

FBDD + "Decomposable" AND-nodes

(Two sub-DAGs do not share variables)



New Queries From H_k

Consider the $k+1$ clauses that form H_k

$H_{k0} = \forall x_0 \forall y_0\, (R(x_0) \vee S_1(x_0,y_0))$

$H_{k1} = \forall x_1 \forall y_1\, (S_1(x_1,y_1) \vee S_2(x_1,y_1))$

$H_{k2} = \forall x_2 \forall y_2\, (S_2(x_2,y_2) \vee S_3(x_2,y_2))$

…

$H_{kk} = \forall x_k \forall y_k\, (S_k(x_k,y_k) \vee T(y_k))$

Asymmetric WFOMC

Theorem. [Beame'14] If the query Q is *any Boolean combination* of the formulas $H_{k0}, \ldots, H_{kk}$ then:

- Any DPLL-based algorithm takes time $\Omega(2^{\sqrt{n}})$ time
- Any Decision-DNNF has $\Omega(2^{\sqrt{n}})$ nodes.

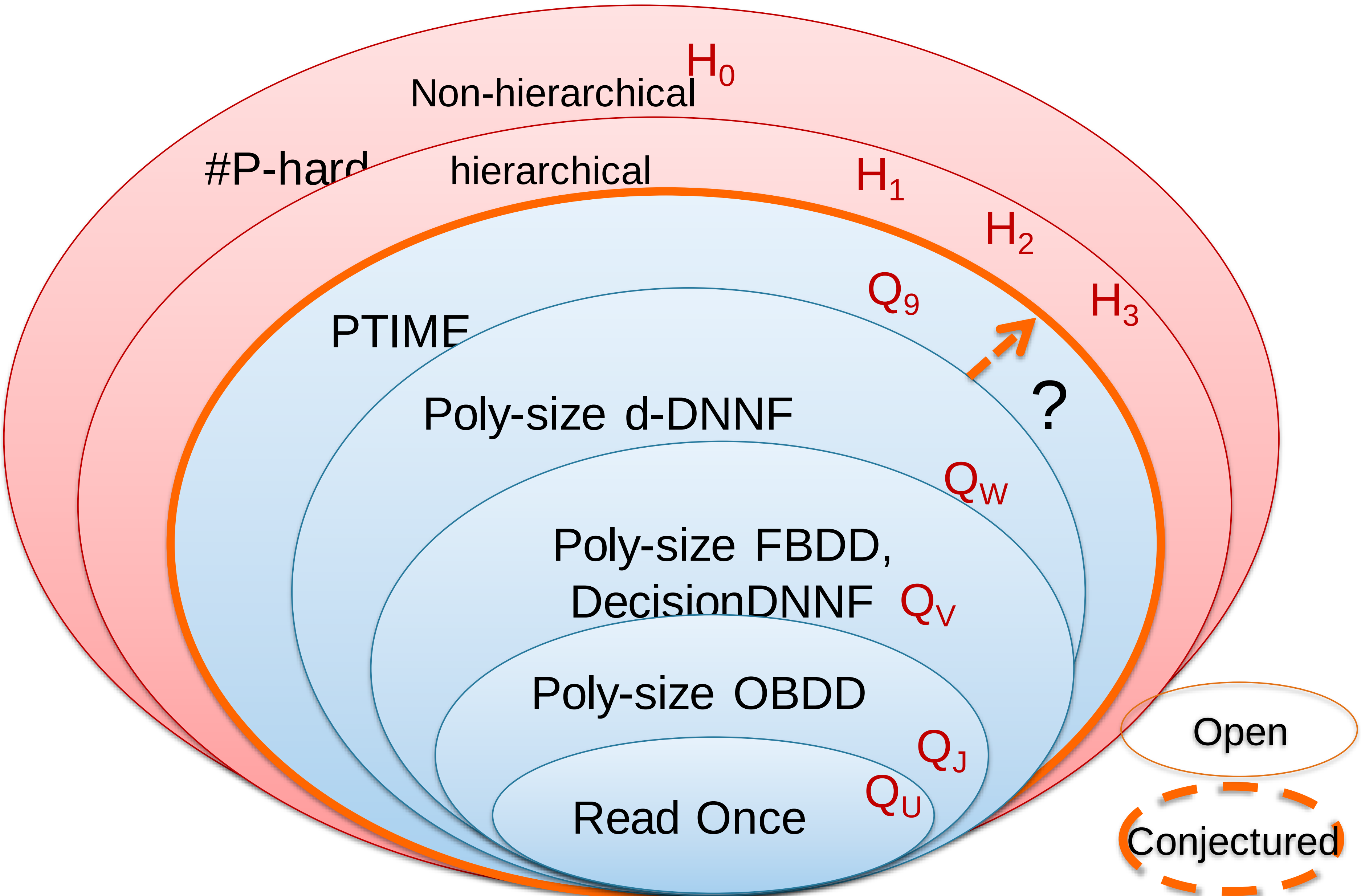
For example, Q_W is a Boolean combination of H_{30}, H_{31}, H_{32}, H_{33}.
Liftable (hence PTIME), yet grounded WMC takes exponential time

Discussion

- This answers question 2: there exists queries that (a) are liftable, and (b) grounded algorithms like DPLL search or Decision-DNNF run in exponential time
- Perhaps there are more powerful grounded algorithms? We don’t know. Open problem: do d-DNNFs compute these queries in PTIME?

Möbius Über Alles

[Suciu'11]



Outline

- Part 1: Motivation
- Part 2: Probabilistic Databases
- Part 3: Weighted Model Counting
- Part 4: Lifted Inference for WFOMC
- Part 5: The Power of Lifted Inference
- Part 6: Conclusion/Open Problems

Summary

- Relational models = the vast majority of data today, plus probabilistic Databases
- Weighted Model Counting = Uniform approach to Probabilistic Inference
- Lifted Inference = really simple rules
- The Power of Lifted Inference = we can prove that lifted inference is better

Lifted Algorithms (in the AI community)

- Exact Probabilistic Inference
  - First-Order Variable Elimination [Poole'03, Braz'05, Milch'08, Taghipour'13]
  - First-Order Knowledge Compilation [V.d.Broeck'11a, '11b, '12a, '13a]
  - Probabilistic Theorem Proving [Gogate'11]
- Approximate Probabilistic Inference
  - Lifted Belief Propagation [Jaimovich'07, Singla'08, Kersting'09]
  - Lifted Bisimulation/Mini-buckets [Sen'08, '09]
  - Lifted Importance Sampling [Gogate'11, '12]
  - Lifted Relax, Compensate & Recover [V.d.Broeck'12b]
  - Lifted MCMC [Niepert'12, Niepert'13, Venugopal'12]
  - Lifted Variational Inference [Choi'12, Bui'12]
  - Lifted MAP-LP [Mladenov'14, Apsel'14]
- Special-Purpose Inference:
  - Lifted Kalman Filter [Ahmadi'11, Choi'11]
  - Lifted Linear Programming [Mladenov'12]

[Kimmig'14]

“But my application has no symmetries?”

1. Statistical relational models have **abundant symmetries**

2. Some **tasks** do not require symmetries in data
   Weight learning, partition functions, single marginals, etc.

3. Symmetries of **computation** are not symmetries of data
   Belief propagation and MAP-LP require weaker automorphisms

4. Over-symmetric **approximations**
   - Approximate P(Q|DB) by P(Q|DB')
   - DB' has more symmetries than DB (is more liftable)

   ➔ Very high speed improvements

   ➔ Low approximation error

[Kersting’09, Mladenov’14, V.d.Broeck’13b]

Open Problems

Symmetric spaces:

- Prove hardness for ANY lifted inference task. Likely needed: #P1-hardness.
- Are lifted inference rules complete beyond FO^2?

Asymmetric spaces:

- Prove completeness for CNF FO formulas
- Extend lifted inference algorithms beyond liftable formulas (need approximations)
- Measure of complexity as a function of the FO formula AND the database D. E.g. if D has bounded treewidth then tractable

Final Thoughts

Long-term outlook: probabilistic inference exploits

- 1988: conditional independence
- 2000: contextual independence (local structure)

201?: Exchangeability/Symmetries
Need lifted inference!

Thank You!

Questions?

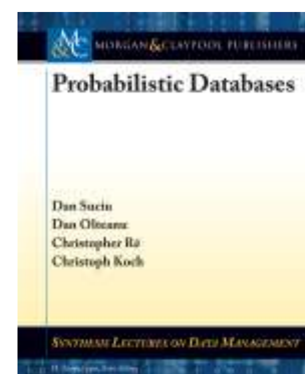
Thank You!

Questions?

StarAI Workshop
@ AAAI on Sunday

Probabilistic
Inference
Inside!

[Suciu'11]



References

[Gartner’06]
Gartner (2006). Market Share: Relational Database Management Systems by Operating System, Worldwide

[Carlson’10]
Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., & Mitchell, T. M. (2010, July). Toward an Architecture for Never-Ending Language Learning. In AAAI (Vol. 5, p. 3).

[Dong’14]
Dong, X. L., Murphy, K., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Strohmann, T., Sun, S. & Zhang, W. (2014). Knowledge Vault: A Web-scale approach to probabilistic knowledge fusion.

[Niu’12]
Niu, F., Zhang, C., Ré, C., & Shavlik, J. W. (2012). DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. In VLDS (pp. 25-28).

[Richardson’06]
Richardson, M., & Domingos, P. (2006). Markov logic networks. Machine learning, 62(1-2), 107-136.

[Niepert’14]
Niepert, M., & Van den Broeck, G. (2014). Tractability through exchangeability: A new perspective on efficient probabilistic inference. Proceedings of AAAI.

[Vardi’82]
Vardi, M. Y. (1982). The complexity of relational query languages. In Proceedings of the fourteenth annual ACM symposium on Theory of computing (pp. 137-146). ACM.

References

[Suciu'11]
Suciu, D., Olteanu, D., Ré, C., & Koch, C. (2011). Probabilistic databases. Synthesis Lectures on Data Management, 3(2), 1-180

[Olteanu'08]
Olteanu, D., & Huang, J. (2008). Using OBDDs for efficient query evaluation on probabilistic databases. In Scalable Uncertainty Management (pp. 326-340). Springer Berlin Heidelberg.

[Jha'13]
Jha, A., & Suciu, D. (2013). Knowledge compilation meets database theory: compiling queries to decision diagrams. Theory of Computing Systems, 52(3), 403-440.

[Dalvi'04]
Dalvi, N. and Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In VLDB.

[V.d.Broeck'11a]
Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., & De Raedt, L. (2011, July). Lifted probabilistic inference by first-order knowledge compilation. In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence (pp. 2178-2185). AAAI Press.

[Dalvi'12]
Dalvi, N., & Suciu, D. (2012). The dichotomy of probabilistic inference for unions of conjunctive queries. Journal of the ACM (JACM), 59(6), 30.

References

[Chavira'05]
Chavira, M., & Darwiche, A. (2005). Compiling Bayesian networks with local structure. In IJCAI (Vol. 5, pp. 1306-1312).

[Sang'05]
Sang, T., Beame, P., & Kautz, H. A. (2005, July). Performing Bayesian inference by weighted model counting. In AAAI (Vol. 5, pp. 475-481).

[Chavira'08]
Chavira, M., & Darwiche, A. (2008). On probabilistic inference by weighted model counting. Artificial Intelligence, 172(6), 772-799.

[Choi'13]
Choi, A., Kisa, D., & Darwiche, A. (2013). Compiling probabilistic graphical models using sentential decision diagrams. In Symbolic and Quantitative Approaches to Reasoning with Uncertainty (pp. 121-132). Springer Berlin Heidelberg.

[Chavira'06]
Chavira, M., Darwiche, A., & Jaeger, M. (2006). Compiling relational Bayesian networks for exact inference. International Journal of Approximate Reasoning, 42(1), 4-20.

[Fierens'11]
Fierens, D., Broeck, G. V. D., Thon, I., Gutmann, B., & De Raedt, L. (2011). Inference in probabilistic logic programs using weighted CNF's. in UAI.

References

[Fierens'13]
Fierens, D., Van den Broeck, G., Renkens, J., Shterionov, D., Gutmann, B., Thon, I., ... & De Raedt, L. (2013). Inference and learning in probabilistic logic programs using weighted boolean formulas. Theory and Practice of Logic Programming, 1-44.

[Gogate'11]
Gogate, V., & Domingos, P. (2012). Probabilistic theorem proving. Proceedings of Uncertainty in AI.

[V.d.Boeck'13a]
Van den Broeck, G. (2013). Lifted Inference and Learning in Statistical Relational Models (Doctoral dissertation, Ph. D. Dissertation, KU Leuven).

[V.d.Boeck'14]
Van den Broeck, G., Meert, W., & Darwiche, A. (2013). Skolemization for weighted first-order model counting. Proceedings of KR.

[Gribkoff'14]
Gribkoff, E., Van den Broeck, G., & Suciu, D. (2014). Understanding the Complexity of Lifted Inference and Asymmetric Weighted Model Counting. Proceedings of Uncertainty in AI.

[Jha'12]
Jha, A., & Suciu, D. (2012). Probabilistic databases with MarkoViews. Proceedings of the VLDB Endowment, 5(11), 1160-1171.

References

[V.d.Broeck'11b]
Van den Broeck, G. (2011). On the completeness of first-order knowledge compilation for lifted probabilistic inference. In Advances in Neural Information Processing Systems (pp. 1386-1394).

[Jaeger-'12]
Jaeger, M., & Van den Broeck, G. (2012, August). Liftability of probabilistic inference: Upper and lower bounds. In Proceedings of the 2nd International Workshop on Statistical Relational AI.

[Cook'71]
Cook, S. A. (1971, May). The complexity of theorem-proving procedures. In Proceedings of the third annual ACM symposium on Theory of computing (pp. 151-158). ACM.

[Valiant'79]
Valiant, L. G. (1979). The complexity of computing the permanent. Theoretical computer science, 8(2), 189-201.

[Provan'83]
Provan, J. S., & Ball, M. O. (1983). The complexity of counting cuts and of computing the probability that a graph is connected. SIAM Journal on Computing, 12(4), 777-788.

[Stanley'97]
Stanley, R. P. 1997. Enumerative Combinatorics. Cambridge University Press.

References

[Davis'60]
Davis, M., & Putnam, H. (1960). A computing procedure for quantification theory. Journal of the ACM (JACM), 7(3), 201-215.

[Davis'62]
Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem-proving. Communications of the ACM, 5(7), 394-397.

[Huang'05]
Huang, J., & Darwiche, A. (2005, July). DPLL with a trace: From SAT to knowledge compilation. In IJCAI (Vol. 5, pp. 156-162).

[Huang'07]
Huang, J., & Darwiche, A. (2007). The Language of Search. J. Artif. Intell. Res.(JAIR), 29, 191-219.

[Beame'14]
Beame, P., Li, J., Roy, S., & Suciu, D. (2014). Counting of Query Expressions: Limitations of Propositional Methods. In ICDT (pp. 177-188).

[Poole'03]
Poole, D. (2003, August). First-order probabilistic inference. In IJCAI (Vol. 3, pp. 985-991).

[Braz'05]
Braz, R., Amir, E., & Roth, D. (2005, July). Lifted first-order probabilistic inference. In Proceedings of the 19th international joint conference on Artificial intelligence (pp. 1319-1325).

References

[Milch'08]
Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008, July). Lifted Probabilistic Inference with Counting Formulas. In AAAI (Vol. 8, pp. 1062-1068).

[Taghipour'13]
Taghipour, N., Fierens, D., Davis, J., & Blockeel, H. (2014). Lifted variable elimination: Decoupling the operators from the constraint language. JAIR

[V.d.Broeck'12a]
Van den Broeck, G., & Davis, J. (2012, July). Conditioning in First-Order Knowledge Compilation and Lifted Probabilistic Inference. In AAAI.

[Jaimovich'07]
Jaimovich, A., Meshi, O., & Friedman, N. (2007). Template based inference in symmetric relational Markov random fields. Proceedings of Uncertainty in AI

[Singla'08]
Singla, P., & Domingos, P. (2008, July). Lifted First-Order Belief Propagation. In AAAI (Vol. 8, pp. 1094-1099).

[Kersting'09]
Kersting, K., Ahmadi, B., & Natarajan, S. (2009, June). Counting belief propagation. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (pp. 277-284). AUAI Press.

References

[Sen’08]
Sen, P., Deshpande, A., & Getoor, L. (2008). Exploiting shared correlations in probabilistic databases. Proceedings of the VLDB Endowment, 1(1), 809-820.

[Sen’09]
Sen, P., Deshpande, A., & Getoor, L. (2009). Bisimulation-based approximate lifted inference. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (pp. 496-505). AUAI Press.

[Gogate’12]
Gogate, V., Jha, A. K., & Venugopal, D. (2012). Advances in Lifted Importance Sampling. In AAAI.

[V.d.Broeck’12b]
Van den Broeck, G., Choi, A., & Darwiche, A. (2012). Lifted relax, compensate and then recover: From approximate to exact lifted probabilistic inference. Proceedings of Uncertainty in AI

[Nieper’12]
Niepert, M. (2012). Markov chains on orbits of permutation groups. Proceedings of Uncertainty in AI

[Niepert’13]
Niepert, M. (2013). Symmetry-Aware Marginal Density Estimation. Proceedings of AAAI.

[Venugopal’12]
Venugopal, D., & Gogate, V. (2012). On lifting the gibbs sampling algorithm. In Advances in Neural Information Processing Systems (pp. 1655-1663).

References

[Choi'12]
Choi, J., & Amir, E. (2012). Lifted relational variational inference. Proceedings of Uncertainty in AI

[Bui'12]
Bui, H. H., Huynh, T. N., & Riedel, S. (2012). Automorphism groups of graphical models and lifted variational inference. StarAI

[Mladenov'14]
Mladenov, M., Kersting, K., & Globerson, A. (2014). Efficient Lifting of MAP LP Relaxations Using k-Locality. In Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (pp. 623-632).

[Apsel'14]
Apsel, U., Kersting, K., & Mladenov, M. (2014). Lifting Relational MAP-LPs using Cluster Signatures. Proceedings of AAAI

[Ahmadi'11]
Ahmadi, B., Kersting, K., & Sanner, S. (2011, July). Multi-evidence lifted message passing, with application to pagerank and the kalman filter. In IJCAI Proceedings-International Joint Conference on Artificial Intelligence (Vol. 22, No. 1, p. 1152).

[Choi'11]
Choi, J., Guzman-Rivera, A., & Amir, E. (2011, June). Lifted Relational Kalman Filtering. In IJCAI (pp. 2092-2099).

References

[Mladenov'12]
Mladenov, M., Ahmadi, B., & Kersting, K. (2012). Lifted linear programming. In International Conference on Artificial Intelligence and Statistics (pp. 788-797).

[Kimmig'14]
Kimmig, A., Mihalkova, L., & Getoor, L. (2014). Lifted graphical models: a survey. *Machine Learning*, 1-45.

[V.d.Broeck'13b]
Van den Broeck, G., & Darwiche, A. (2013). On the complexity and approximation of binary evidence in lifted inference. In Advances in Neural Information Processing Systems (pp. 2868-2876).