# A TUTORIAL ON PROBABILISTIC DATABASES

Dan Suciu
University of Washington

# Probabilistic Databases

- Data: standard relational data, plus probabilities that measure the degree of uncertainty
- Queries: standard SQL queries, whose answers are annotated with output probabilities

# A Little History of Probabilistic DBs

## Early days

- Wong'82
- Shoshani'82
- Cavallo&Pittarelli'87
- Barbara'92
- Lakshmanan'97,'01
- Fuhr&Roellke'97
- Zimanyi'97

Main challenge:
Query Evaluation
(=Probabilistic Inference)

## Recent work

- Stanford (Trio)
- UW (MystiQ)
- Cornell (MayBMS)
- Oxford (MayBMS)
- U.of Maryland
- IBM Almaden (MCDB)
- Rice (MCDB)
- U. of Waterloo
- UBC
- U. of Florida
- Purdue University
- U. of Wisconsin

# Why?

Many applications need to manage uncertain data

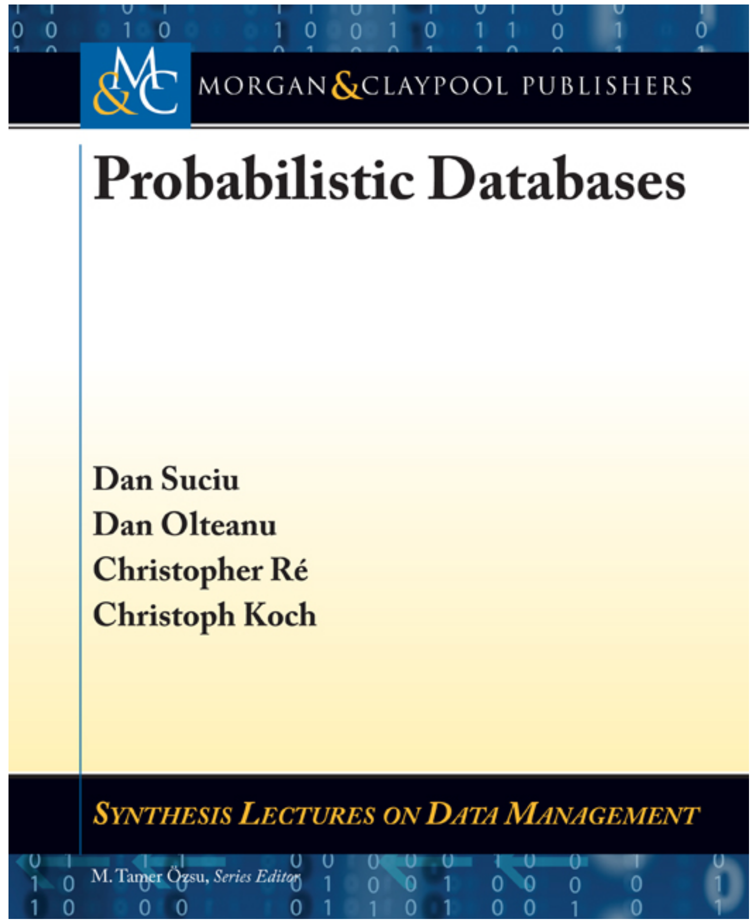
- Information extraction
- Knowledge representation
- Fuzzy matching
- Business intelligence
- Data integration
- Scientific data management
- Data anonymization

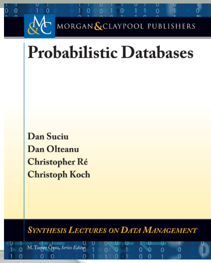
# What?

- Probabilistic Databases extend Relational Databases with probabilities
- Combine Formal Logic with Probabilistic Inference
- Requires a new thinking for both databases and probabilistic inference

# This Tutorial: Query Evaluation

Based on the book:






# Outline of the Tutorial

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

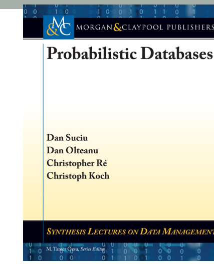
3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

Part 3

5. Extensional Evaluation — Chapter 4.1

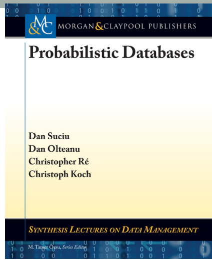
Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions




# What You Will Learn

- Background:
  - Relational data model: tables, queries, relational algebra
  - PTIME, NP, #P
  - Model counting: DPLL, OBDD, FBDD, d-DNNF
- In detail:
  - Extensional plans, extensional evaluation, running them in postgres
  - The landscape of query complexity: from PTIME to #P-complete,
  - Query compilation: Read-Once Formulas, OBDD, FBDD, d-DNNF
- Less detail:
  - The #P-hardness proof, complexity of BDDs
- Omitted:
  - Richer data models: BID, GM, XML, continuous random values)
  - Approximate query evaluation,
  - Ranking query answers



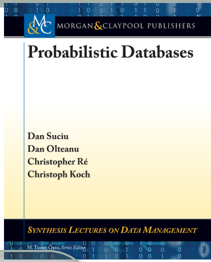

# Related Work. See book, plus:

These references are not in the book

- Wegener: Branching programs and binary decision diagrams: theory and applications, 2000
- Dalvi, S.: The dichotomy of probabilistic inference for unions of conjunctive queries, JACM'2012
- Huang, Darwiche: DPLL with a Trace: From SAT to Knowledge Compilation, IJCAI 2005
- Beame, Li, Roy, S.: Lower Bounds for Exact Model Counting and Applications in Probabilistic Databases, UAI'13
- Gatterbauer, S.: Oblivious Bounds on the Probability of Boolean Functions, under review

The applications are from:

- Ré, Letchner, Balazinska, S: Event queries on correlated probabilistic streams. SIGMOD Conference 2008
- Gupta, Sarawagi: Creating Probabilistic Databases from Information Extraction Models. VLDB 2006
- Stoyanovich, Davidson, Milo, Tannen: Deriving probabilistic databases with inference ensembles. ICDE 2011
- Beskales, Soliman, Ilyas, Ben-David: Modeling and Querying Possible Repairs in Duplicate Detection. PVLDB 2009
- Kumar, Ré: Probabilistic Management of OCR Data using an RDBMS. PVLDB 2011




# Outline

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

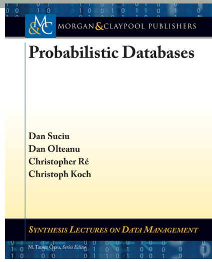
Part 3

5. Extensional Evaluation — Chapter 4.1

Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions

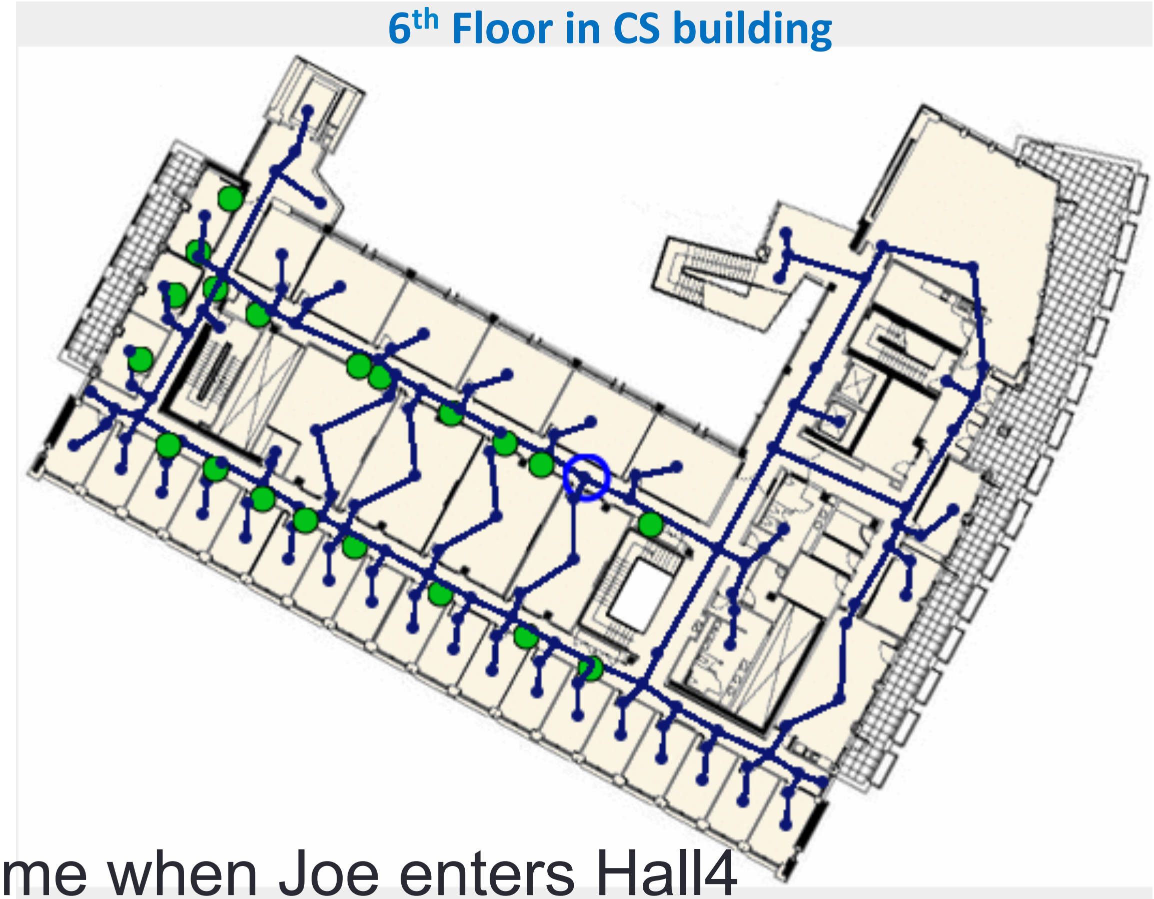
# Motivation




- Many applications have structured, but uncertain data
- Uncertain data is modeled as probabilistic data
- Outputs to SQL queries annotated with probabilities

[Re' 08]

# Example1: RFID Tracking




Simple Query: "Alert me when Joe enters Hall4"

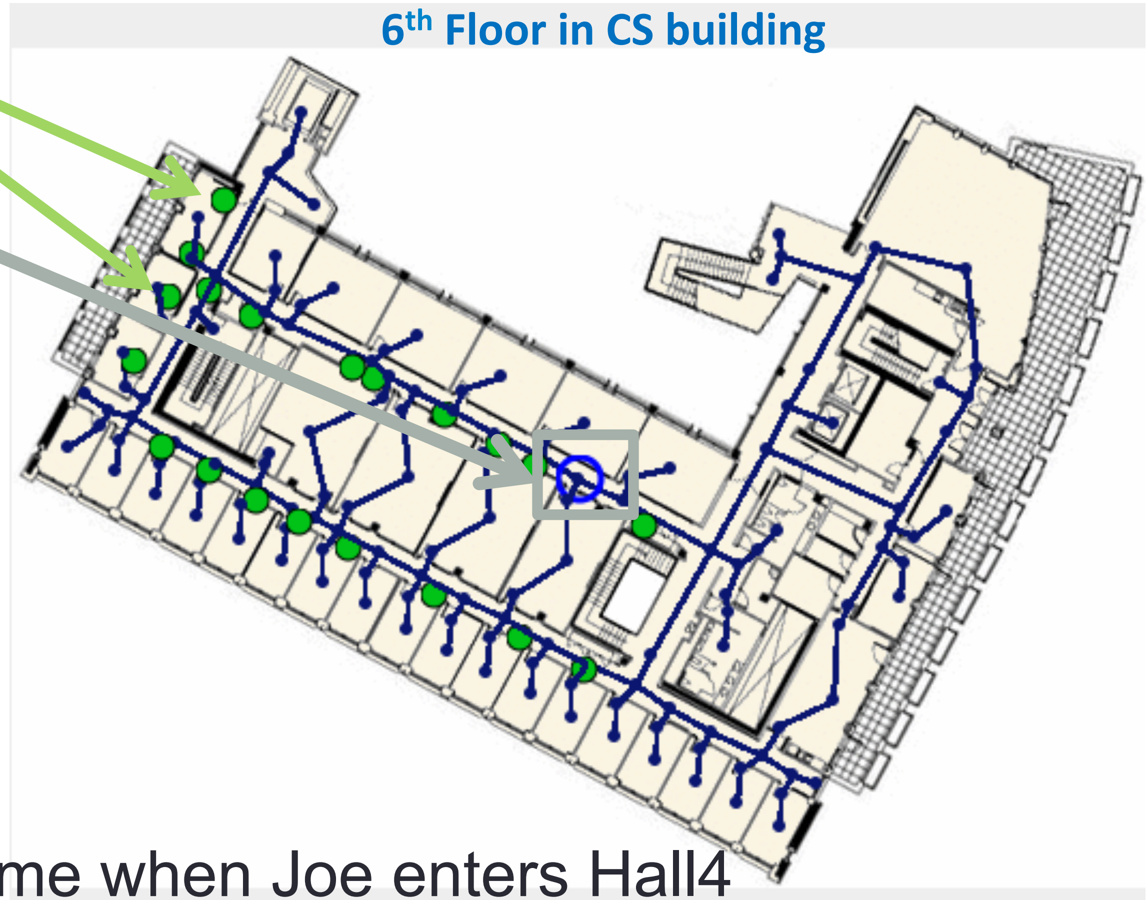
Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

# Example1: RFID Tracking

Antennas

Blue ring is Joe's Location



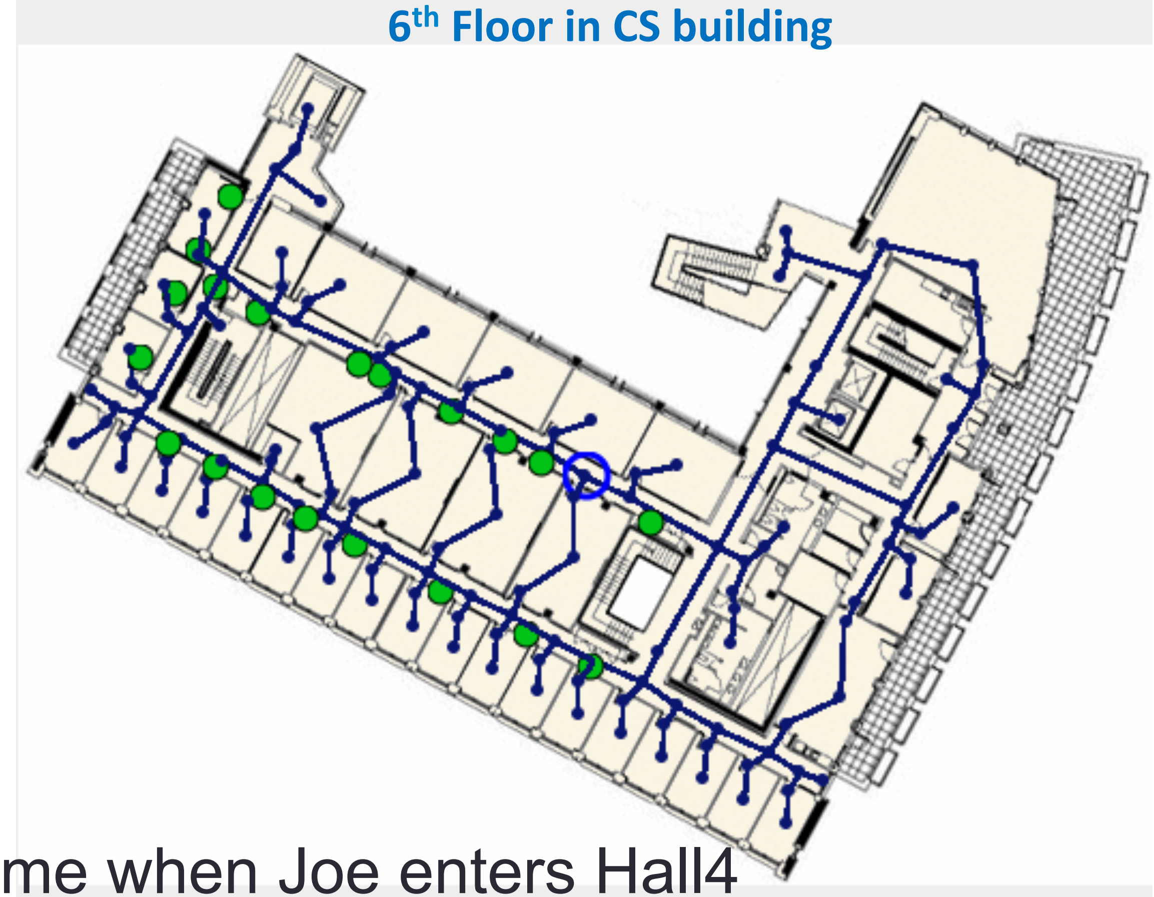

Simple Query: "Alert me when Joe enters Hall4"

Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

# Example1: RFID Tracking

(Watch movie)




Simple Query: "Alert me when Joe enters Hall4"

Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

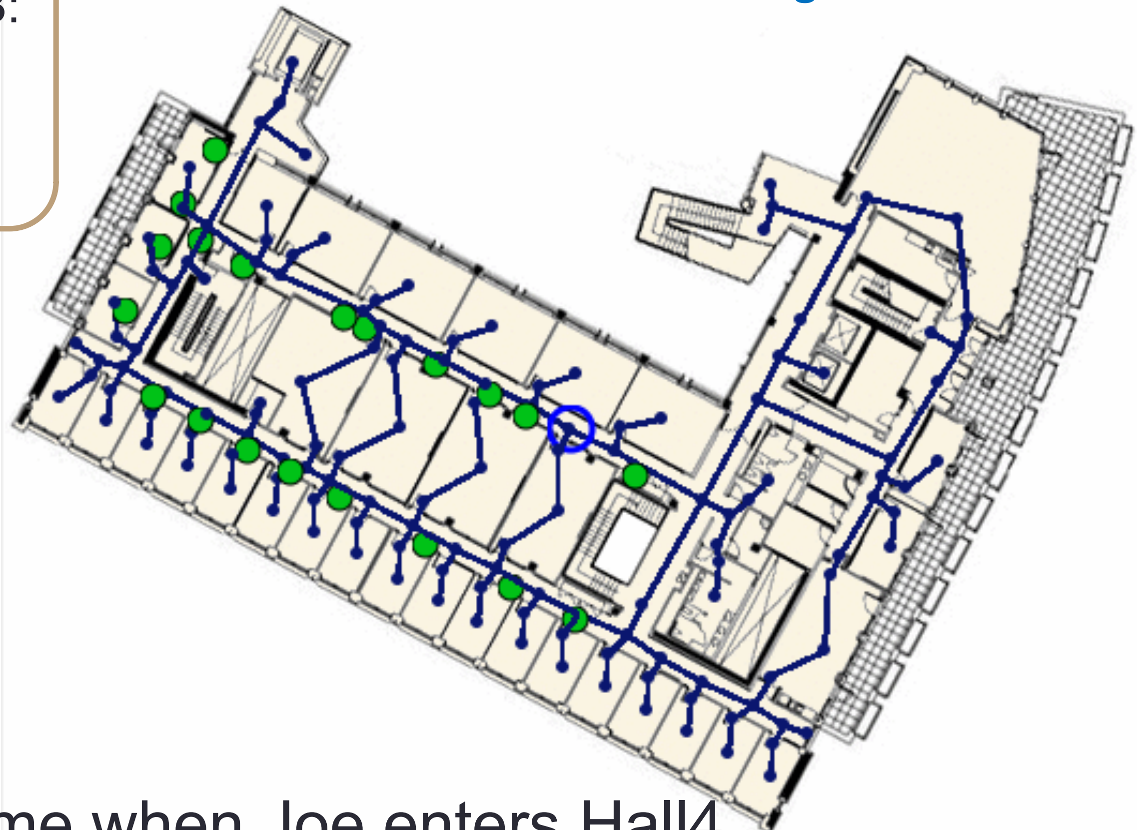
# Example1: RFID Tracking

Standard Deterministic DB:

- Missed readings
- Duplicate readings
- Granuarity mismatch

| Tag | Loc | Time |
|---|---|---|
| Joe | Hall4 | 3 |
| Joe | Office3 | 4 |
| Joe | Hall4 | 7 |
| Joe | Office3 | 7 |
| Bob | …. | … |

6th Floor in CS building



Simple Query: "Alert me when Joe enters Hall4"

Complex Query: "Who brought their laptops to the meeting"

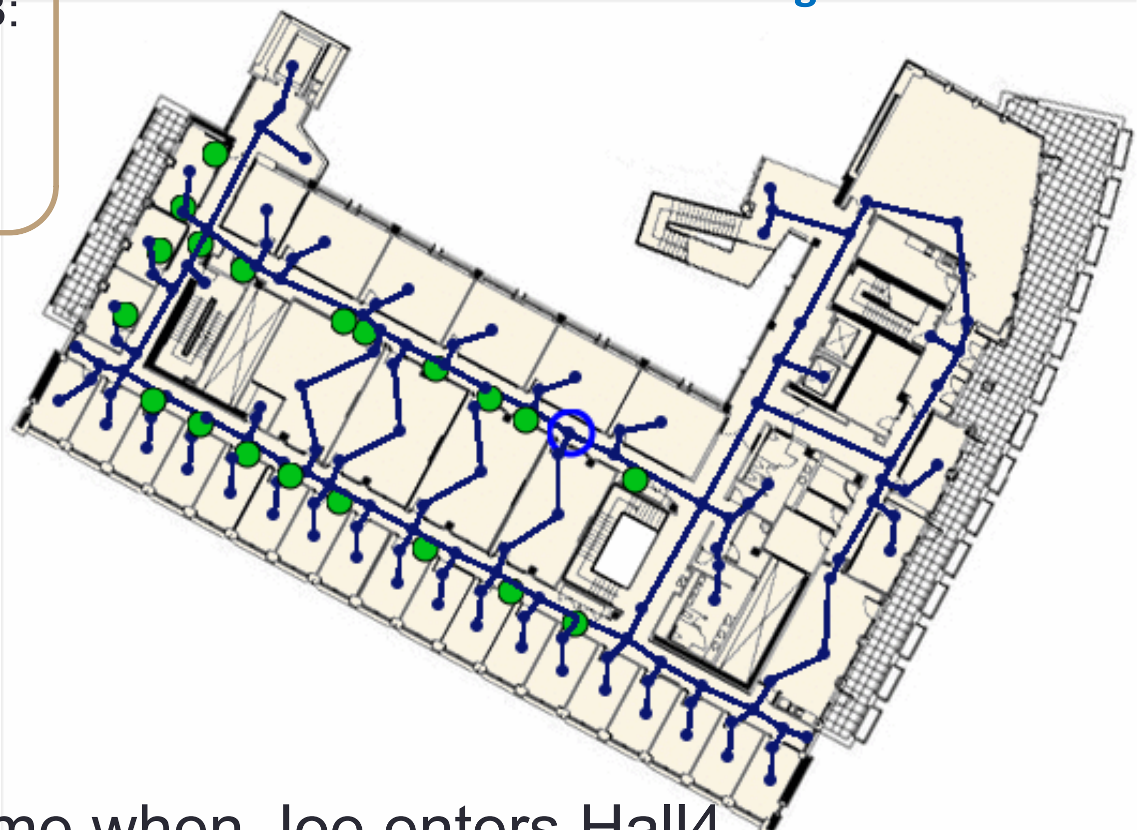
[Re' 08]

# Example1: RFID Tracking

Standard Deterministic DB:
- Missed readings
- Duplicate readings
- Granuarity mismatch

| Tag | Loc | Time |
|---|---|---|
| Joe | Hall4 | 3 |
| Joe | Office3 | 4 |
| Joe | Hall4 | 7 |
| Joe | Office3 | 7 |
| Bob | …. | … |

6th Floor in CS building



Simple Query: "Alert me when Joe enters Hall4"

Complex Query: "Who brought their laptops to the meeting"

Queries are virtually impossible to answer using SQL

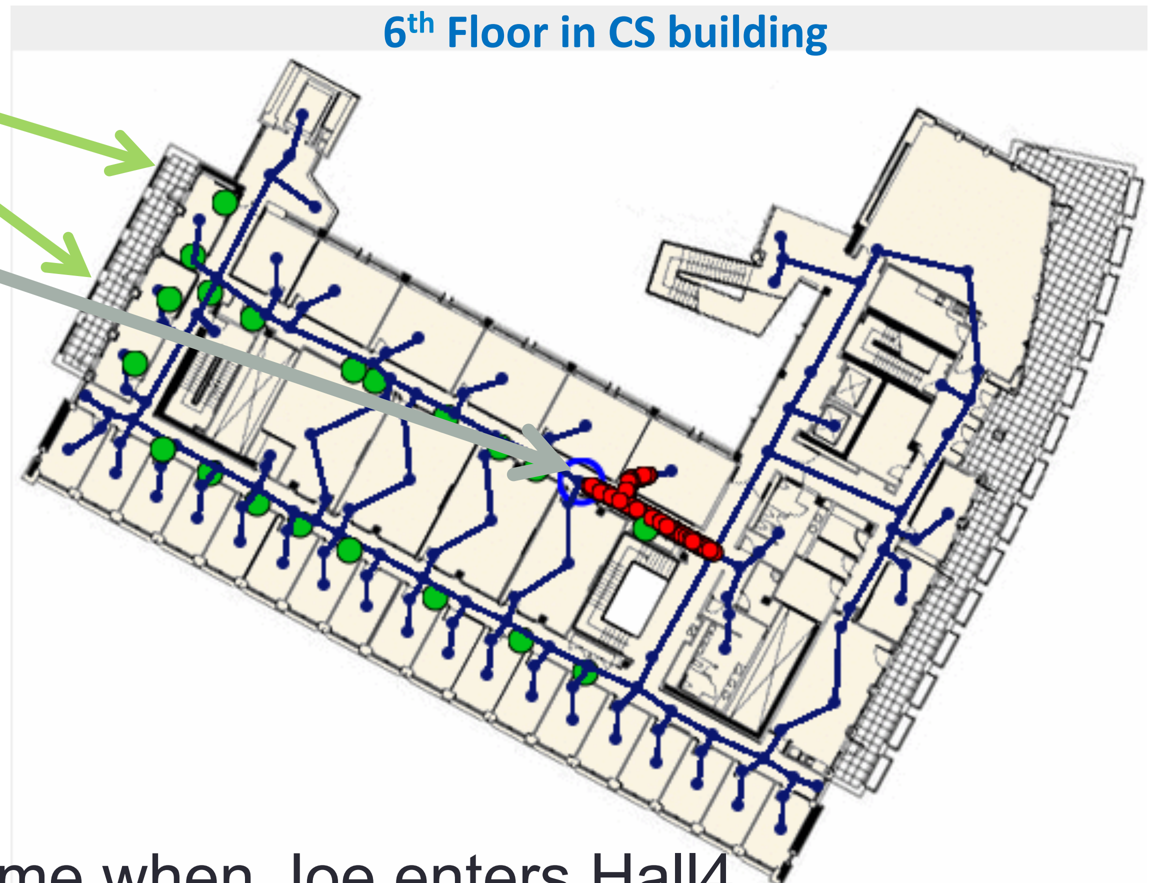
[Re' 08]

Particle Filters
[Doucet et al' 01]

# Example1: RFID Tracking

Antennas

Blue ring is *ground truth*




Simple Query: "Alert me when Joe enters Hall4

Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

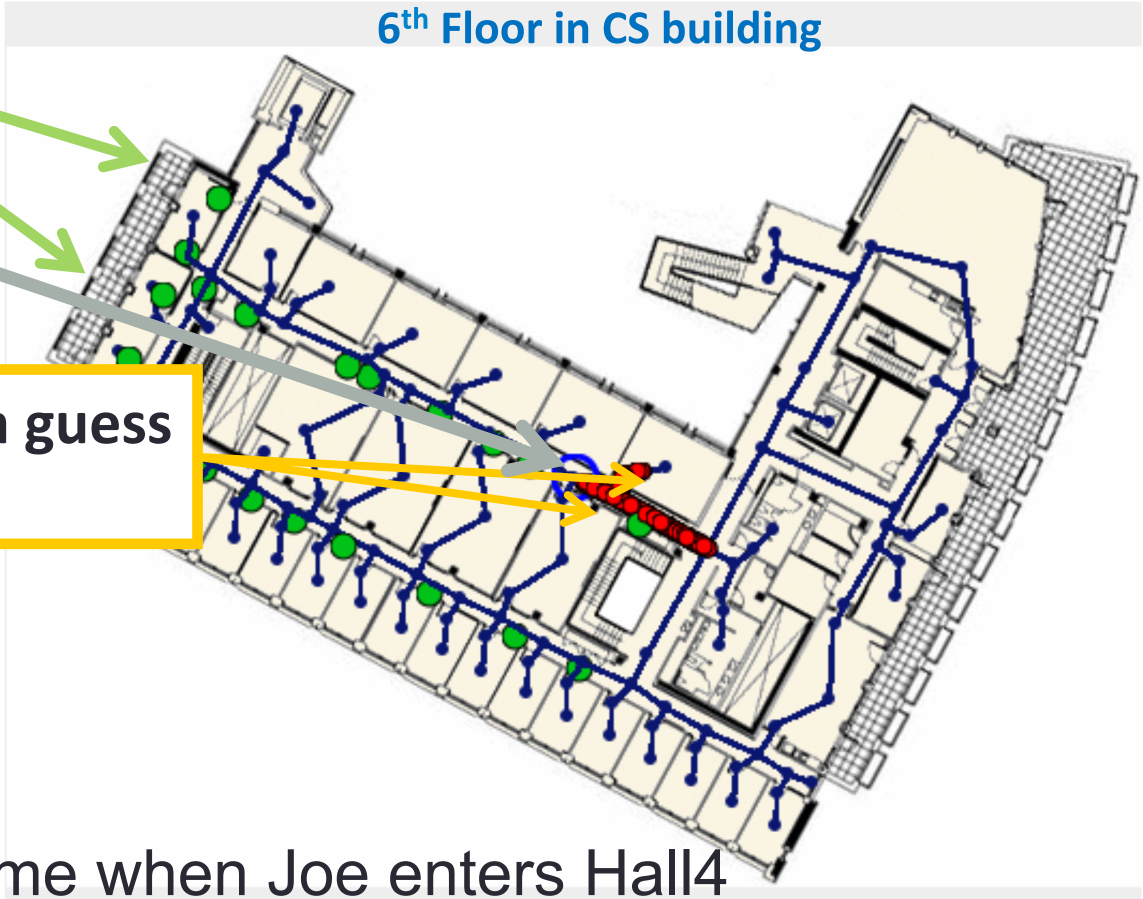
Particle Filters [Doucet et al' 01]

# Example1: RFID Tracking

Antennas

Blue ring is *ground truth*

**Each orange particle is a guess of Joe's location**



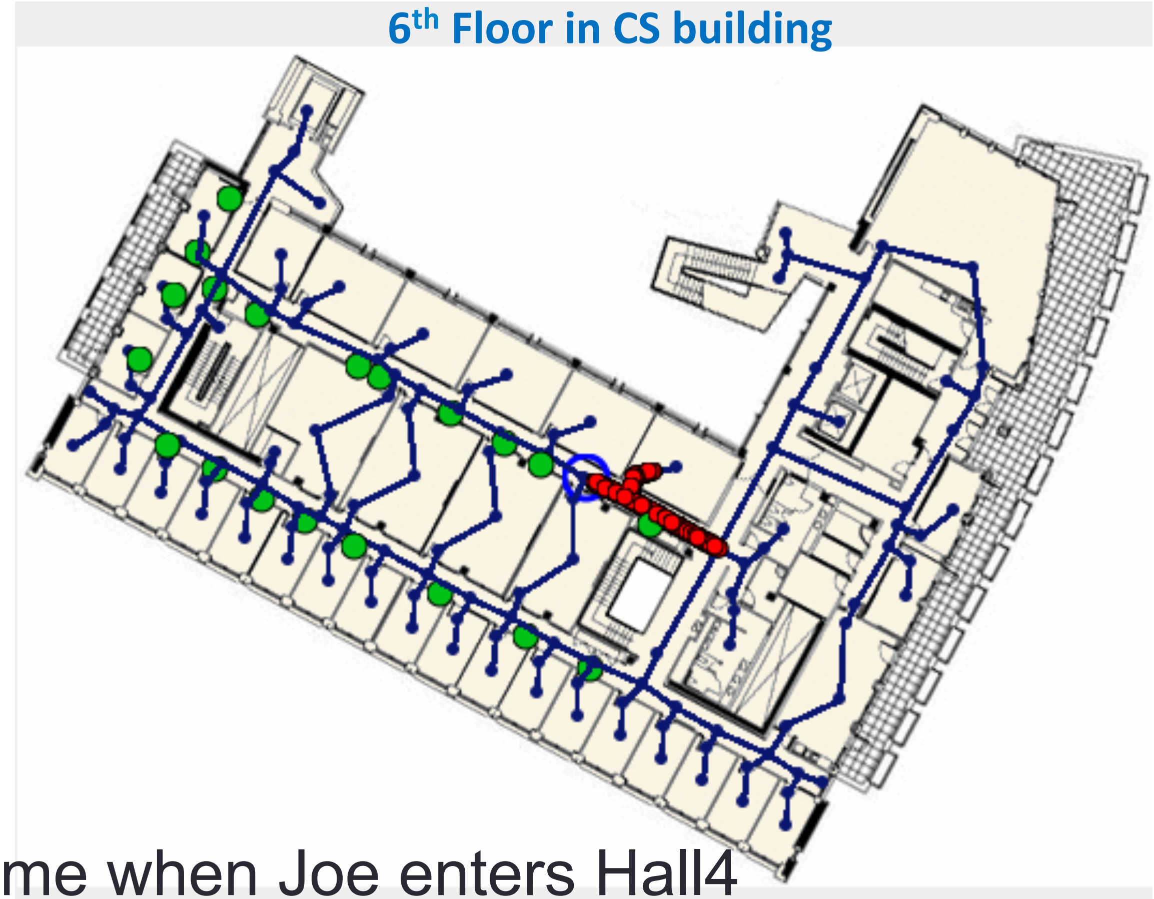

Simple Query: "Alert me when Joe enters Hall4"
Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

Particle Filters [Doucet et al' 01]

# Example1: RFID Tracking

(Watch movie)




Simple Query: "Alert me when Joe enters Hall4

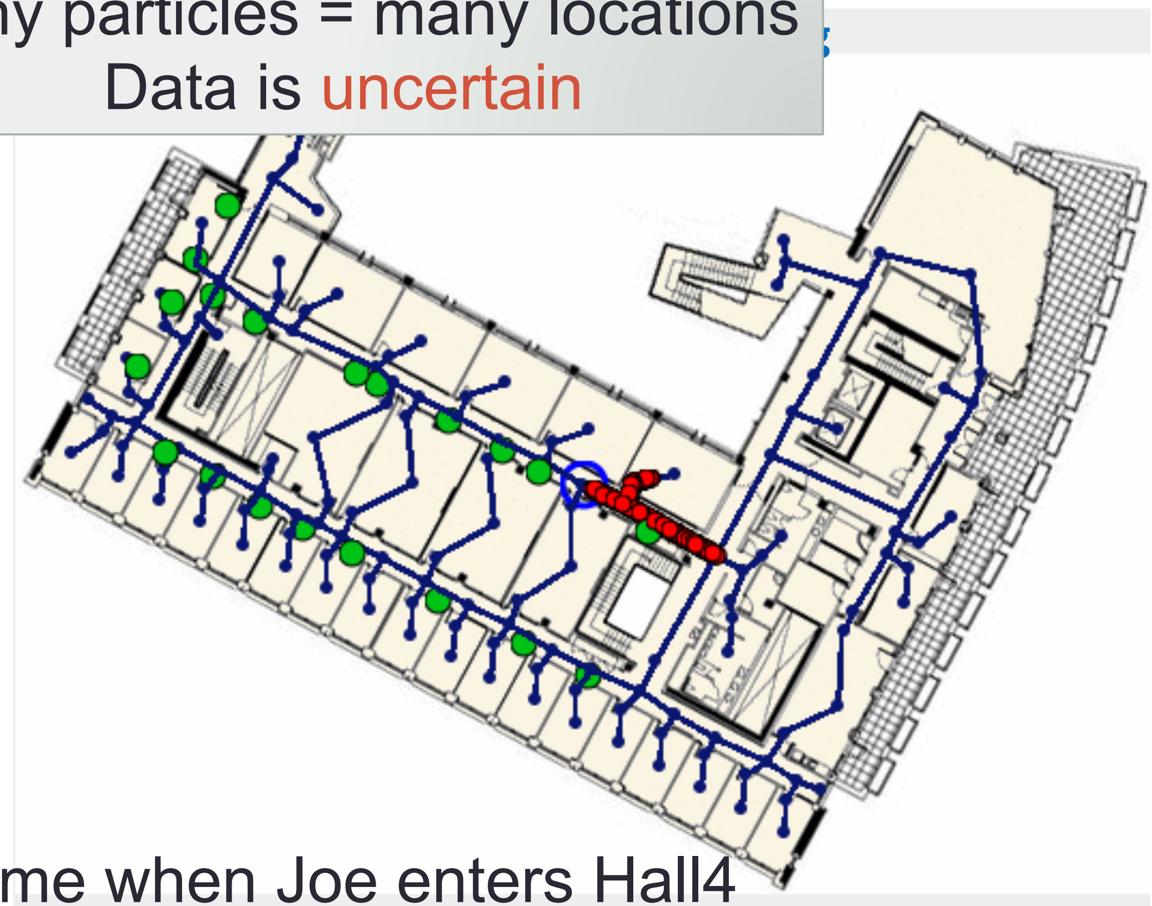
Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

Particle Filters [Doucet et al' 01]

# Example1: RFID Tracking

Many particles = many locations
Data is uncertain



Simple Query: "Alert me when Joe enters Hall4"
Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

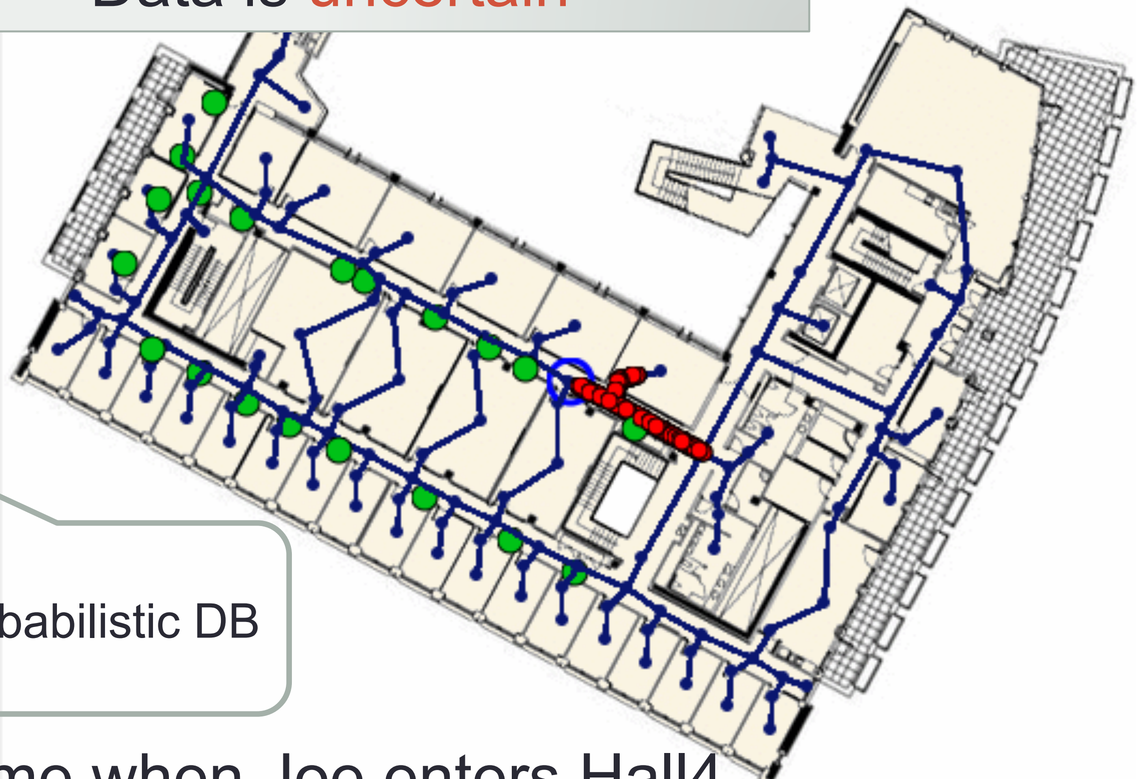
Particle Filters
[Doucet et al' 01]

# Example1: RFID Tracking

Many particles = many locations
Data is uncertain

| Tag | Loc | Time | P |
|---|---|---|---|
| Joe | Hall4 | 3 | 0.8 |
| Joe | Office3 | 3 | 0.2 |
| Joe | Hall4 | 4 | 0.4 |
| Joe | Office3 | 4 | 0.6 |
| Bob | …. | … | … |

Probabilistic DB



Simple Query: "Alert me when Joe enters Hall4"
Complex Query: "Who brought their laptops to the meeting"

[Re' 08]

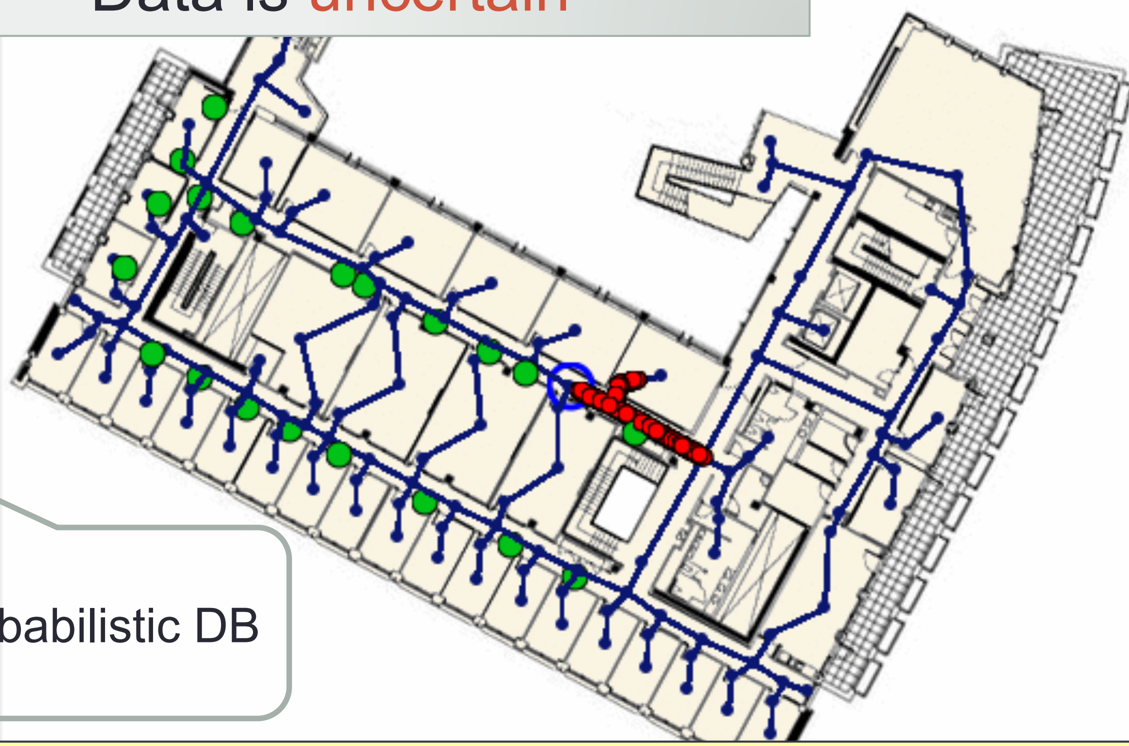
Particle Filters
[Doucet et al' 01]

# Example1: RFID Tracking

Many particles = many locations
Data is uncertain

| Tag | Loc | Time | P |
|---|---|---|---|
| Joe | Hall4 | 3 | 0.8 |
| Joe | Office3 | 3 | 0.2 |
| Joe | Hall4 | 4 | 0.4 |
| Joe | Office3 | 4 | 0.6 |
| Bob | …. | … | … |

Probabilistic DB



Simp
Com

Answers to SQL queries:
"Joe entered Hall4 at t=3 with probability p=0.8"
"Fred was at the meeting and brought his laptop p=0.14"

# Example1: RFID Tracking -- Lessons

- Data is uncertain:
  - Missed readings
  - Duplicate readings
  - Granularity mismatch
- Deterministic databases
  - Can store uncertain data,
  - But impossible to query in SQL
- Probabilistic databases:
  - Store uncertain data annotated with probability p
  - Answer SQL queries by computing their output probability p'

[Gupta'2006]

# Example 2: Information Extraction

```
52-A Goregaon West Mumbai 400 076
```

CRF

Standard DB: keep the most likely extraction

| Id | House_no | Area | City | Pincode | Prob |
|---|---|---|---|---|---|
| 1 | 52 | Goregaon West | Mumbai | 400 062 | 0.1 |
| 1 | 52-A | Goregaon | West Mumbai | 400 062 | 0.2 |
| 1 | 52-A | Goregaon West | Mumbai | 400 062 | 0.5 |
| 1 | 52 | Goregaon | West Mumbai | 400 062 | 0.2 |

Probabilistic DB: keep most/all extractions to increase recall

**Key finding**: the probabilities given by CRFs correlate well with the precision of the extraction.

[Stoyanovich'2011]

# Example 3: Modeling Missing Data

| id | age | edu | inc | nw |
|---|---|---|---|---|
| t1 | 20 | HS | ? | ? |
| t2 | 20 | BS | 50K | 100K |
| t3 | 20 | ? | 50K | ? |
| t4 | 20 | HS | 100K | 500K |
| t5 | 20 | ? | ? | ? |
| t6 | 20 | HS | 50K | 100K |
| t7 | 20 | HS | 50K | 500K |
| t8 | ? | HS | ? | ? |
| t9 | 30 | BS | 100K | 100K |
| t10 | 30 | ? | 100K | ? |
| t11 | 30 | HS | ? | ? |
| t12 | 30 | MS | ? | ? |
| t13 | 40 | BS | 100K | 100K |
| t14 | 40 | HS | ? | ? |
| t15 | 40 | BS | 50K | 500K |
| t16 | 40 | HS | ? | 500K |
| t17 | 40 | HS | 100K | 500K |

Standard DB: NULL

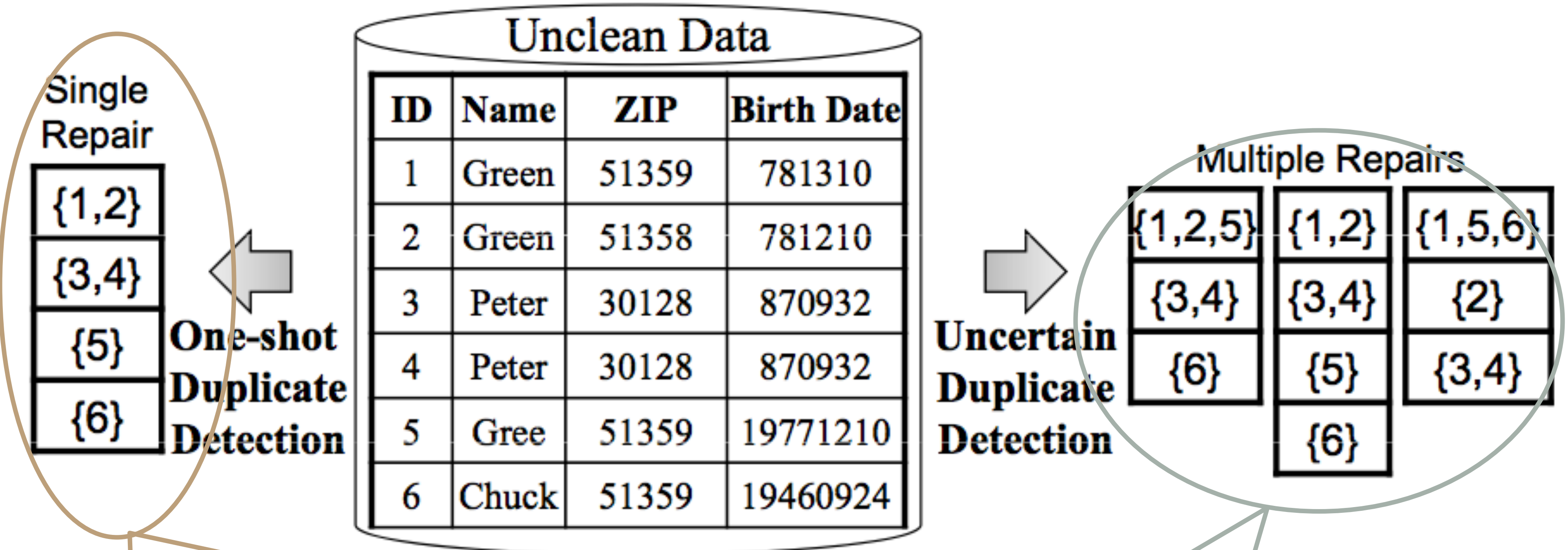
Probabilistic DB: distribution on possible values

| id | age | edu | inc | nw | prob |
|---|---|---|---|---|---|
| t12.1 | 30 | MS | 50K | 100K | 0.30 |
| t12.2 | 30 | MS | 50K | 500K | 0.45 |
| t12.3 | 30 | MS | 100K | 100K | 0.10 |
| t12.4 | 30 | MS | 100K | 500K | 0.15 |

**Key technique**: Meta Rule Semi-Lattice for inferring missing attributes.

[Beskales'2009]

# Example 4: Data Cleaning




| ID | Name | ZIP | Birth Date |
|---|---|---|---|
| 1 | Green | 51359 | 781310 |
| 2 | Green | 51358 | 781210 |
| 3 | Peter | 30128 | 870932 |
| 4 | Peter | 30128 | 870932 |
| 5 | Gree | 51359 | 19771210 |
| 6 | Chuck | 51359 | 19460924 |

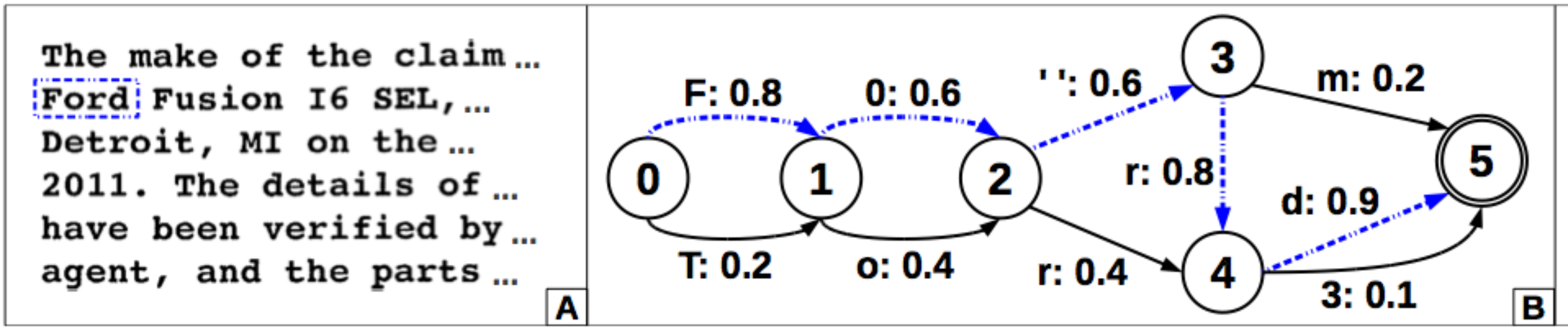
Standard DB
cleaning data means choosing one possible repair

Probabilistic DB
keep many/all possible repairs

**Challenge**: Representing multiple repairs.
[Beskaes'2009] restrict to hierarchical repairs.

[Kumar'2011]

# Example 5: OCR



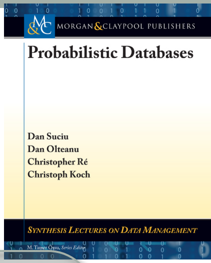

They use OCRopus from Google Books: output is a stohastic automaton
Traditionally: retain only the Maximum Apriori Estimate (MAP)
With a probabilistic database: may retain several alternative recognitions: increase recal

```
SELECT DocId, Loss
FROM Claims
WHERE Year = 2010
   AND DocData LIKE '%Ford%';
```

# Summary of Applications

- Structured, but uncertain data
- Modeled as probabilistic data
- Answers to SQL queries annotated with probabilities

Probabilistic database:
- Combine data management with probabilistic inference




# Outline

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

Part 3

5. Extensional Evaluation — Chapter 4.1

Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions

# Probabilistic Data Model: Outline

- Review: Relational Data Model
- Incomplete Databases
- Probabilistic Databases (PDB)
- Query semantics
- Block Independent Disjoint

# Review: Relational Data Model

Data:
stored in relations (= tables)

Owner

| **Name** | **Object** |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| **Object** | **Time** | **Loc** |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Review: Relational Data Model

Data:
stored in relations (= tables)

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Queries: SQL,

Find all owners of objects in the Office

```
-- SQL: e.g. postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

# Review: Relational Data Model

Data:
stored in relations (= tables)

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Queries: SQL, Unions of Conjunctive Queries

Find all owners of objects in the Office

```
-- SQL: e.g. postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

$$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,y)$$

Note that x,t,y are existentially quantified:

$$Q(z) = \exists x\ \exists t\ \exists y\ (\text{Owner}(z,x), \text{Location}(x,t,y))$$

# Review: Relational Data Model

Data:
stored in relations (= tables)

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Queries: SQL, Unions of Conjunctive Queries

Find all owners of objects in the Office

```
-- SQL: e.g. postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
   and  Location.loc = 'Office'
```

Q(z) = Owner(z,x), Location(x,t,y)

Note that x,t,y are existentially quantified:

$$Q(z) = \exists x\ \exists t\ \exists y\ \ (Owner(z,x), Location(x,t,y))$$

Query answer: Q=

| Name |
|---|
| Joe |
| Jim |

# Review: Relational Data Model

**Data:**
stored in relations (= tables)

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Queries: SQL, Unions of Conjunctive Queries

Find all owners of objects in the Office

```
-- SQL: e.g. postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
   and  Location.loc = 'Office'
```

$Q(z) = Owner(z,x), Location(x,t,y)$

Note that x,t,y are existentially quantified:

$Q(z) = \exists x\ \exists t\ \exists y\ (Owner(z,x), Location(x,t,y))$

Query answer: Q=

| Name |
|---|
| Joe |
| Jim |

# Review: Complexity of Query Evaluation

Query Q, database D

- <u>Data complexity</u>:
  fix Q, complexity = f(D)

Today's talk

- <u>Query complexity</u>:
  fix D, complexity = f(Q)

- <u>Combined complexity</u>: complexity = f(D,Q)

Moshe Vardi

Data complexity is unique to database research

# Incomplete Database

**Definition** An Incomplete Database is a finite set of database instances **W** = $(W_1, W_2, \ldots, W_n)$

Each $W_i$ is called a possible world

# Incomplete Database

**Definition** An Incomplete Database is a finite set of database instances **W** = ($W_1$, $W_2$, …, $W_n$)

Each $W_i$ is called a possible world

| $W_1$ | $W_2$ | $W_3$ | $W_4$ |
|---|---|---|---|
| | | | |

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Incomplete Database

**Definition** An Incomplete Database is a finite set of database instances **W** = ($W_1$, $W_2$, …, $W_n$)

Each $W_i$ is called a possible world

### $W_1$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

### $W_2$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

### $W_3$

### $W_4$

# Incomplete Database

**Definition** An Incomplete Database is a finite set of database instances **W** = ($W_1$, $W_2$, …, $W_n$)

Each $W_i$ is called a possible world

## $W_1$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

## $W_2$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

## $W_3$

Owner

| Name | Object |
|---|---|
| Jim | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

## $W_4$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Incomplete Database: Query Semantics

**Definition** Given query Q, incomplete database **W**:

- An answer t is certain, if $\forall W_i,\ t \in Q(W_i)$
- An answer t is possible if $\exists W_i,\ t \in Q(W_i)$

# Incomplete Database: Query Semantics

**Definition** Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i,\ t \in Q(W_i)$
- An answer t is possible if $\exists W_i,\ t \in Q(W_i)$

Q(z) = Owner(z,x), Location(x,t,y)

## $W_1$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

## $W_2$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

## $W_3$

Owner

| Name | Object |
|---|---|
| Joe | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

## $W_4$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Incomplete Database: Query Semantics

**Definition** Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i$, $t \in Q(W_i)$
- An answer t is possible if $\exists W_i$, $t \in Q(W_i)$

$Q(z) = Owner(z,x), Location(x,t,y)$

## $W_1$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

## $W_2$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |

## $W_3$

Owner

| Name | Object |
|---|---|
| Joe | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

Q=

| |
|---|
| Joe |

## $W_4$

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

# Incomplete Database: Query Semantics

**Definition** Given query Q, incomplete database **W**:
- An answer t is certain, if $\forall W_i,\ t \in Q(W_i)$
- An answer t is possible if $\exists W_i,\ t \in Q(W_i)$

Q(z) = Owner(z,x), Location(x,t,y)

Certain answers to Q: Joe
Possible answers to Q: Joe, Jim

## $W_1$

Owner

| **Name** | **Object** |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| **Object** | **Time** | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

## $W_2$

Owner

| **Name** | **Object** |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| **Object** | **Time** | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |

## $W_3$

Owner

| **Name** | **Object** |
|---|---|
| Joe | Laptop77 |

Location

| **Object** | **Time** | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

Q=

| |
|---|
| Joe |

## $W_4$

Owner

| **Name** | **Object** |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| **Object** | **Time** | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

# Probabilistic Database

**Definition** A Probabilistic Database is (**W**, P), where **W** is an incomplete database, and P: **W** → [0,1] a probability distribution: $\Sigma_{i=1,n} P(W_i) = 1$

# Probabilistic Database

**Definition** A Probabilistic Database is (**W**, P), where **W** is an incomplete database, and P: **W** → [0,1] a probability distribution: $\Sigma_{i=1,n} P(W_i) = 1$

## $W_1$ — 0.3

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

## $W_2$ — 0.4

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

## $W_3$ — 0.2

Owner

| Name | Object |
|---|---|
| Jim | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

## $W_4$ — 0.1

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Probabilistic Database: Query Semantics

**Definition** Given query Q, probabilistic database (**W**,P):

- The marginal probability of an answer t is:

  $P(t) = \Sigma \ \{ P(W_i) \mid W_i \in \mathbf{W},\ t \in Q(W_i) \}$

# Probabilistic Database: Query Semantics

**Definition** Given query Q, probabilistic database (**W**,P):
- The marginal probability of an answer t is:

$P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W},\ t \in Q(W_i) \}$

$Q(z) = Owner(z,x), Location(x,t,y)$

## $W_1$ — 0.3

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

## $W_2$ — 0.4

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

## $W_3$ — 0.2

Owner

| Name | Object |
|---|---|
| Jim | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

## $W_4$ — 0.1

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

# Probabilistic Database: Query Semantics

**Definition** Given query Q, probabilistic database (**W**,P):

- The marginal probability of an answer t is:

$P(t) = \Sigma \ \{ P(W_i) \mid W_i \in \mathbf{W}, \ t \in Q(W_i) \}$

Q(z) = Owner(z,x), Location(x,t,y)

## $W_1$

Owner 0.3

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

## $W_2$

Owner 0.4

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |

## $W_3$

Owner 0.2

| Name | Object |
|---|---|
| Joe | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

Q=

| |
|---|
| Joe |

## $W_4$

Owner 0.1

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

# Probabilistic Database: Query Semantics

**Definition** Given query Q, probabilistic database (**W**,P):

- The marginal probability of an answer t is:

$P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W},\ t \in Q(W_i) \}$

Q(z) = Owner(z,x), Location(x,t,y)

P(Joe) = 1.0
P(Jim) = 0.4

## $W_1$ — 0.3

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Joe | Laptop77 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

## $W_2$ — 0.4

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |

## $W_3$ — 0.2

Owner

| Name | Object |
|---|---|
| Joe | Laptop77 |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |

Q=

| |
|---|
| Joe |

## $W_4$ — 0.1

Owner

| Name | Object |
|---|---|
| Joe | Book302 |
| Jim | Laptop77 |
| Fred | GgleGlass |

Location

| Object | Time | Loc |
|---|---|---|
| Laptop77 | 5:07 | Hall |
| Laptop77 | 9:05 | Office |
| Book302 | 8:18 | Office |

Q=

| |
|---|
| Joe |
| Jim |

# Discussion

- Intuition: a probabilistic database says that the database can be in one of possible states, each with a probability
- Possible query answers: a set of answers annotated with probabilities:

  $(t_1, p_1), (t_2, p_2), (t_3, p_3), \ldots$

  Usually: $p_1 \geq p_2 \geq p_3 \geq \ldots$
- Problem: the number of possible world in a probabilistic database is astronomically large. To represent it, we impose some restrictions

# Independent, Disjoint Tuples

**Definition** Given a probabilistic database (**W**, P).
Two tuples $t_1$, $t_2$ are called:

- Independent, if: $P(t_1\ t_2) = P(t_1)\ P(t_2)$
- Disjoint (or exclusive), if: $P(t_1 t_2) = 0$

# Independent, Disjoint Tuples

**Definition** Given a probabilistic database (**W**, P).
Two tuples $t_1$, $t_2$ are called:

- Independent, if: $P(t_1\ t_2) = P(t_1)\ P(t_2)$
- Disjoint (or exclusive), if: $P(t_1 t_2) = 0$

**Definition** A probabilistic database is called Block-Independent-Disjoint (BID), if its tuples are grouped into blocks such that:

- Tuples from the same block are disjoint
- Tuples from different blocks are independent

# Example: BID Table

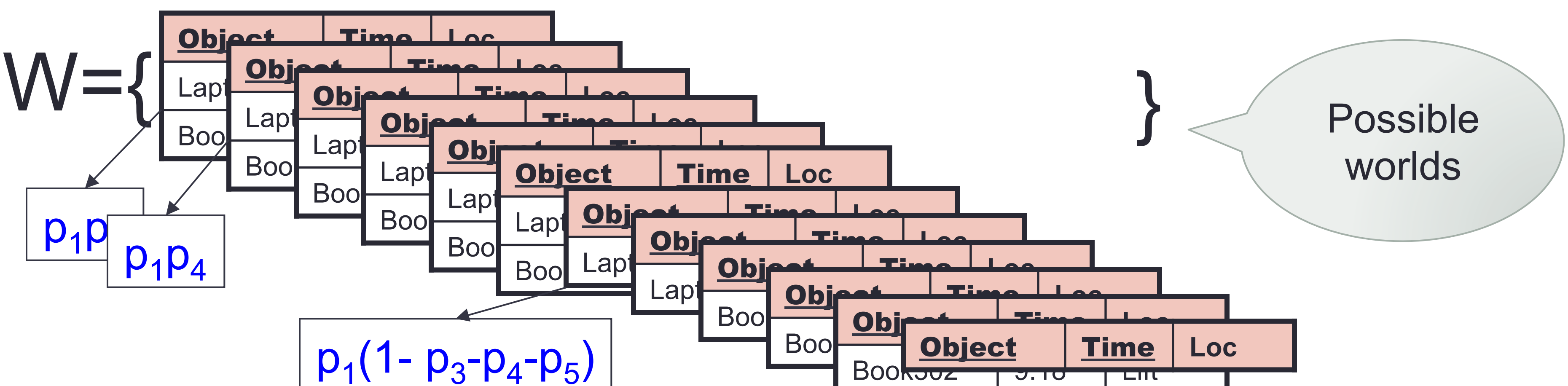
BID Table

| Object | Time | Loc | P |
|---|---|---|---|
| Laptop77 | 9:07 | Rm444 | $p_1$ |
| Laptop77 | 9:07 | Hall | $p_2$ |
| Book302 | 9:18 | Office | $p_3$ |
| Book302 | 9:18 | Rm444 | $p_4$ |
| Book302 | 9:18 | Lift | $p_5$ |

Disjoint

Disjoint

Inde-pen-dent



# The Query Evaluation Problem

Given: a BID database D, a query Q, and output tuple t

Compute: P(t)

Note: D has, say, 1000000 tuples,
while the number of possible worlds is $2^{1000000}$

Challenge: compute P(t) efficiently, in the size of D

Data complexity: the complexity of P depends dramatically on Q

# An Example

Boolean query

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1- \{1- p1*[\ 1-(1-q1)*(1-q2)\ ]\} * \{1- p2*[\ 1-(1-q3)*(1-q4)*(1-q5)]\}$$

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

R

| x | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

S

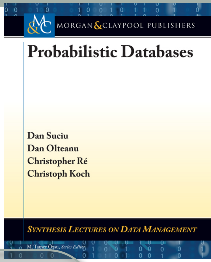
| x | y | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

# Summary of the Probabilistic Data Model

- Possible Worlds Semantics: very powerful but difficult to represent
- Block-Independent-Disjoint databases have efficient representation: D is stored in a traditional database
- Independent Databases: even simpler

This tutorial

Main challenge: evaluate Q efficiently in the size of D




# Outline

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

Part 3

5. Extensional Evaluation — Chapter 4.1

Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions

# Background: Relational Algebra

1. Join $\bowtie$
2. Projection (w/ duplicate elimination) $\Pi$
3. Union $\cup$
4. Selection $\sigma$

# Background: Query Plans

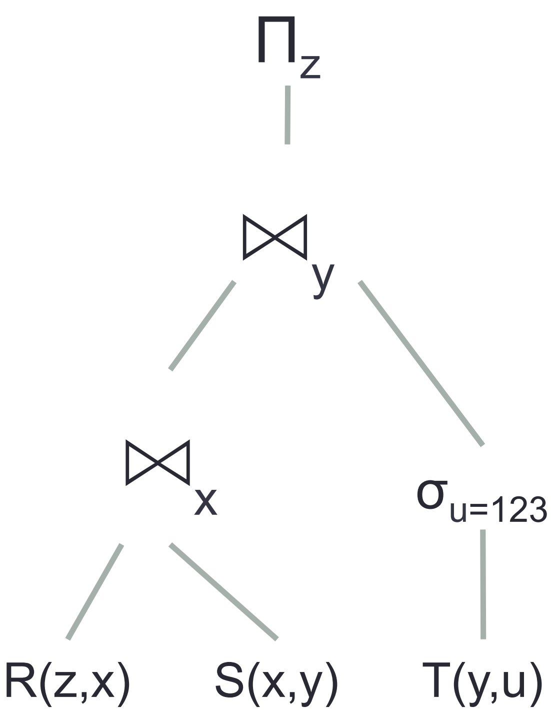
```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

Q(z) = R(z,x), S(x,y),T(y,u)

# Background: Query Plans

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

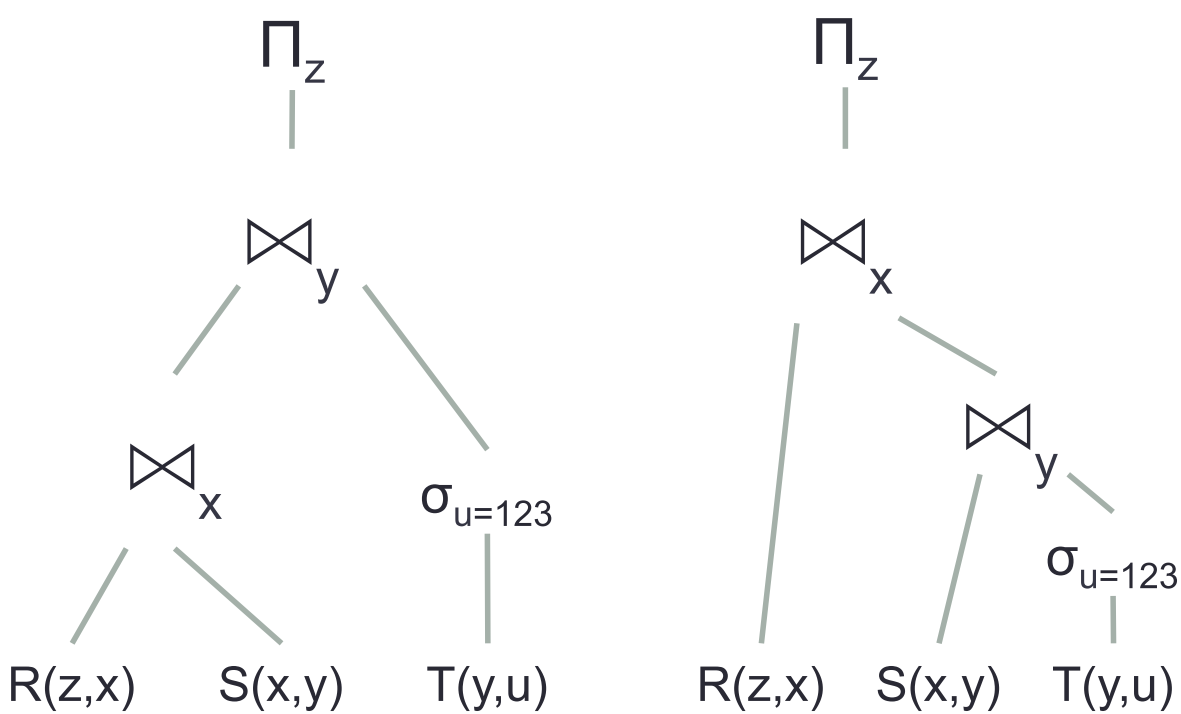
$Q(z) = R(z,x), S(x,y), T(y,u)$



# Background: Query Plans

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

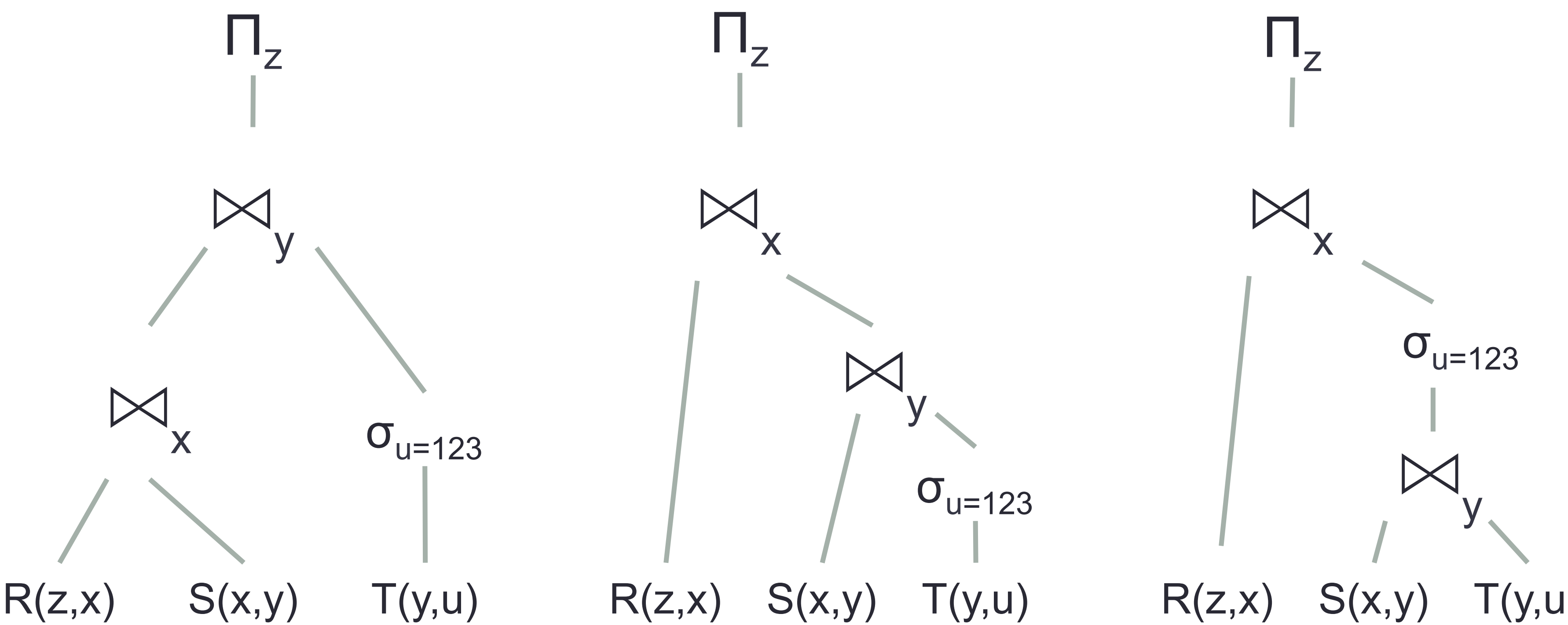
Q(z) = R(z,x), S(x,y),T(y,u)



# Background: Query Plans

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

$Q(z) = R(z,x), S(x,y), T(y,u)$



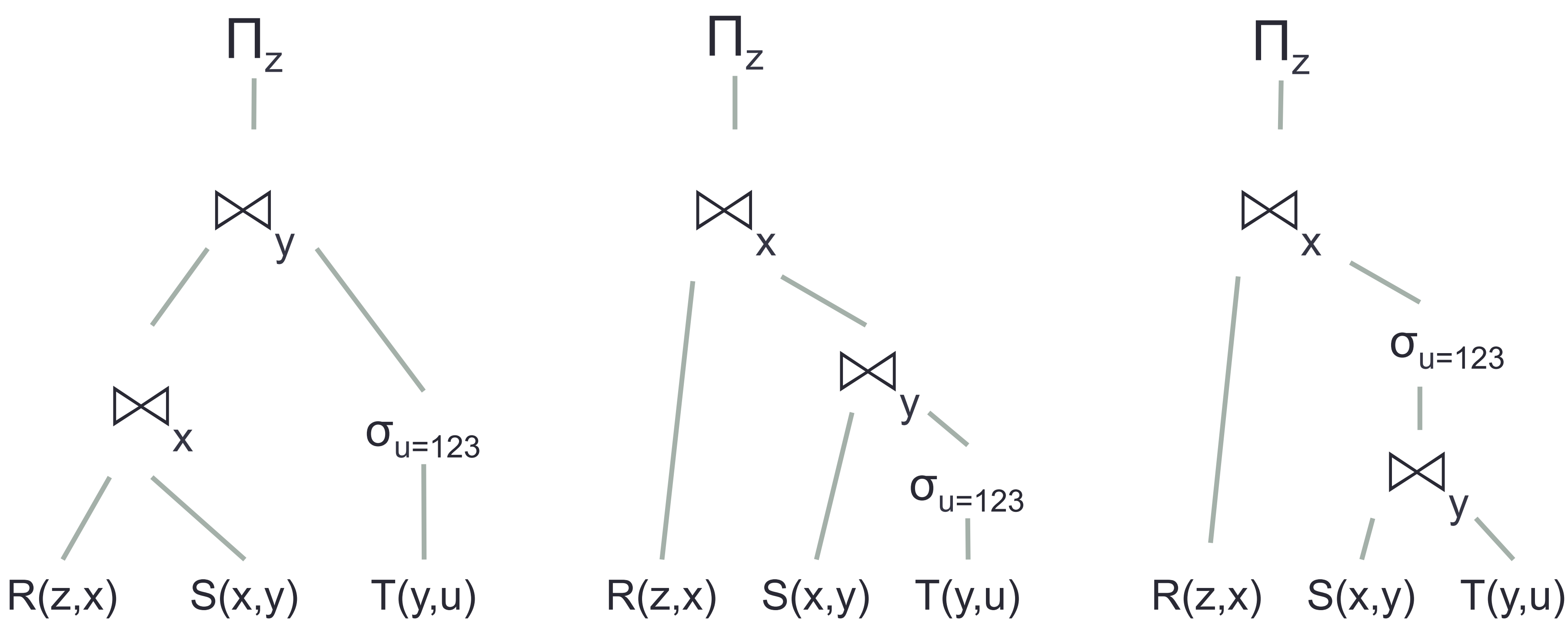
# Background: Query Plans

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

Q(z) = R(z,x), S(x,y),T(y,u)

These plans are equivalent!

The query optimizer will select a plan with minimal cost.



# Extensional Plans

- Main idea:
  - Modify each operator to compute output probabilities

- Must make some assumption:
  - Probabilities are independent, or
  - Probabilities are disjoint (exclusive)

# Extensional Operators

Independent join

| A | B | P |
|---|---|---|
| a1 | b1 | p1*q1 |
| a1 | b2 | p1*q2 |
| a2 | b3 | p2*q3 |
| a2 | b4 | p2*q4 |
| a2 | b5 | p2*q5 |

$\bowtie_i$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

# Extensional Operators

Independent join

| A | B | P |
|---|---|---|
| a1 | b1 | p1*q1 |
| a1 | b2 | p1*q2 |
| a2 | b3 | p2*q3 |
| a2 | b4 | p2*q4 |
| a2 | b5 | p2*q5 |

$\bowtie^{i}$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

Independent project

| A | P |
|---|---|
| a1 | 1 - (1-q1)*(1-q2) |
| a2 | 1 - (1-q3)*(1-q4)*(1-q5) |

$\Pi^{i}_{A}$

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

# Extensional Operators

## Independent join

| A | B | P |
|---|---|---|
| a1 | b1 | p1*q1 |
| a1 | b2 | p1*q2 |
| a2 | b3 | p2*q3 |
| a2 | b4 | p2*q4 |
| a2 | b5 | p2*q5 |

$\bowtie^{i}$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

## Independent project

| A | P |
|---|---|
| a1 | 1 - (1-q1)*(1-q2) |
| a2 | 1 - (1-q3)*(1-q4)*(1-q5) |

$\Pi^{i}_{A}$

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

## Selection

| A | B | P |
|---|---|---|
| a2 | b2 | q3 |
| a2 | b3 | q4 |
| a2 | b2 | q5 |

$\sigma_{A=a2}$

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b1 | q2 |
| a2 | b2 | q3 |
| a2 | b3 | q4 |
| a2 | b2 | q5 |

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

$Q() = R(x), S(x,y)$

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

# Example

R

| x | P |
|---|---|
| $a_1$ | $p_1$ |
| $a_2$ | $p_2$ |
| $a_3$ | $p_3$ |

S

| x | y | P |
|---|---|---|
| $a_1$ | $b_1$ | $q_1$ |
| $a_1$ | $b_2$ | $q_2$ |
| $a_2$ | $b_3$ | $q_3$ |
| $a_2$ | $b_4$ | $q_4$ |
| $a_2$ | $b_5$ | $q_5$ |

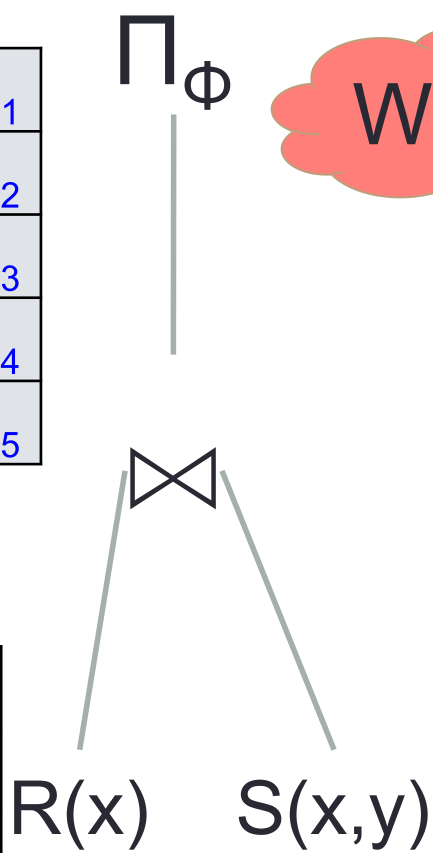
```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

$Q() = R(x), S(x,y)$

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

$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$

| |
|---|
| $p_1\ q_1$ |
| $p_1\ q_2$ |
| $p_2\ q_3$ |
| $p_2\ q_4$ |
| $p_2\ q_5$ |



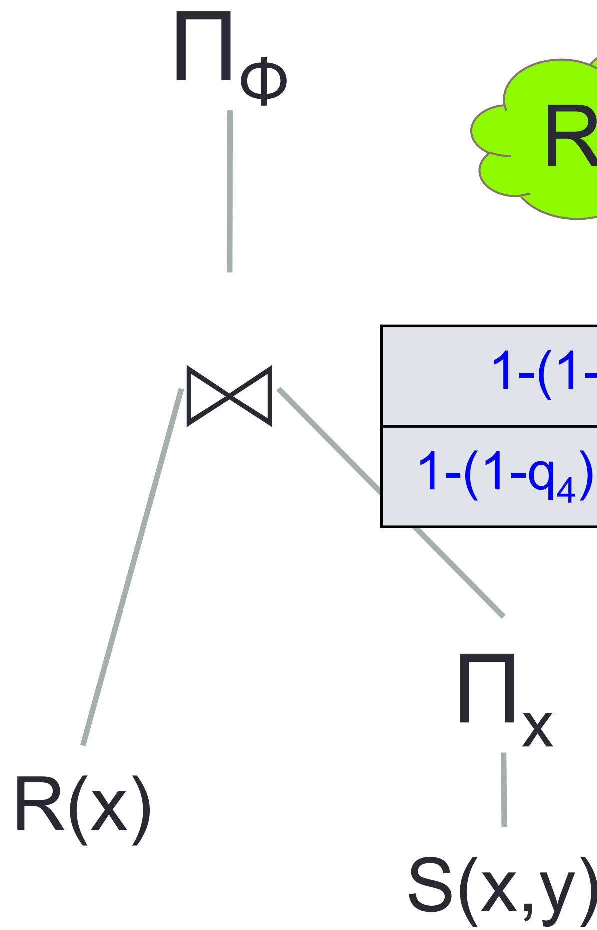

Wrong

| x | P |
|---|---|
| $a_1$ | $p_1$ |
| $a_2$ | $p_2$ |
| $a_3$ | $p_3$ |

| x | y | P |
|---|---|---|
| $a_1$ | $b_1$ | $q_1$ |
| $a_1$ | $b_2$ | $q_2$ |
| $a_2$ | $b_3$ | $q_3$ |
| $a_2$ | $b_4$ | $q_4$ |
| $a_2$ | $b_5$ | $q_5$ |

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

Right




| |
|---|
| $1-(1-q_1)(1-q_2)$ |
| $1-(1-q_4)(1-q_5)\ (1-q_6)$ |

# Safe Plans

- Fix a schema for the probabilistic database
  - E.g. all relations are tuple-independent, or BID with a given key

**Definition**: A plan is safe if it computes probabilities correctly

- Query optimization = find a safe plan

# Lesson 1

- Equivalent plans may become in-equivalent when interpreted as extensional plans
- A correct extensional plan is called a *safe plan*
- Goal: find a safe plan!
- Does every query have a safe plan?

# Unsafe Queries

R

| X | P |
|---|---|
| x1 | p1 |
| x2 | p2 |

S

| X | Y |
|---|---|
| x1 | y1 |
| x1 | y2 |
| x2 | y2 |

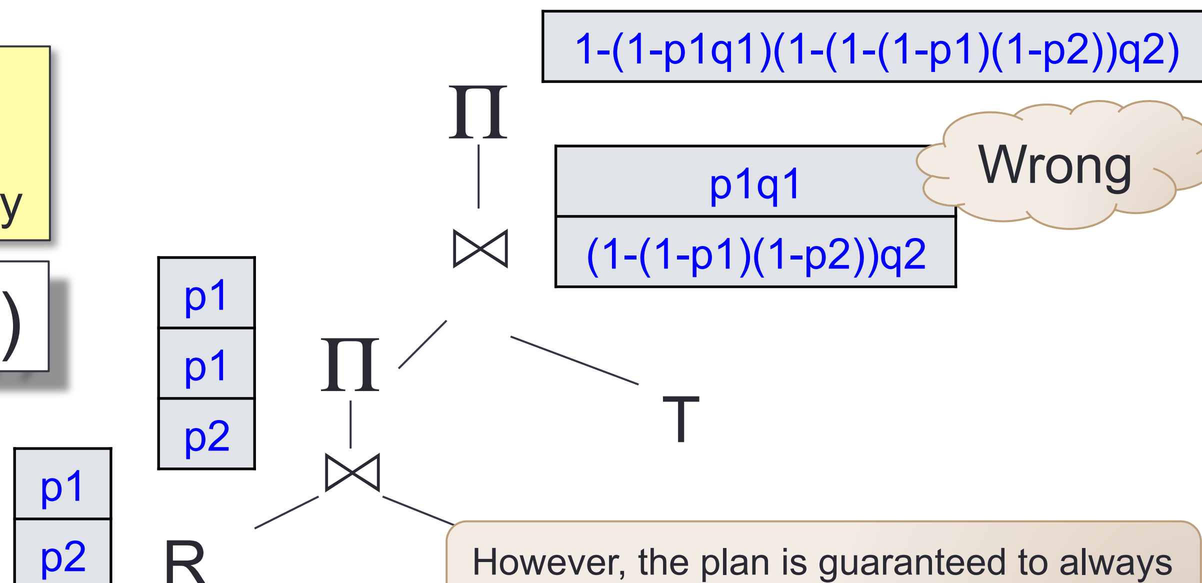
T

| Y | P |
|---|---|
| y1 | q1 |
| y2 | q2 |

```
SELECT DISTINCT 'yes'
FROM R, S, T
WHERE R.x = S.x and S.y = T.y
```

$H_0$ :- R(x),S(x,y),T(y)

#P-hard




However, the plan is guaranteed to always return upper bound [Gatterbauer'13]

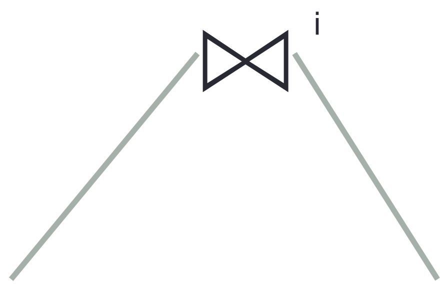
# Discussion

- Safe queries admit a safe plan, and can be computed efficiently
- Unsafe queries do not admit safe plans, and we will prove that they cannot be computed efficiently
- Every extensional plan (safe or unsafe) can be written directly in SQL – will illustrate with postgres
- Every query (safe/unsafe) admits extensional plans that compute upper bounds, or lower bounds of its probability

# Extensional Plans in Postgres

| A | B | P |
|---|---|---|
| a1 | b1 | p1*q1 |
| a1 | b2 | p1*q2 |
| a2 | b3 | p2*q3 |
| a2 | b4 | p2*q4 |
| a2 | b5 | p2*q5 |

```
SELECT R.A, S.B, R.P*S.P
FROM   R, S
WHERE R.A=S.A
```




R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

# Extensional Plans in Postgres

| A | B | P |
|---|---|---|
| a1 | b1 | p1*q1 |
| a1 | b2 | p1*q2 |
| a2 | b3 | p2*q3 |
| a2 | b4 | p2*q4 |
| a2 | b5 | p2*q5 |

```
SELECT R.A, S.B, R.P*S.P
FROM    R, S
WHERE R.A=S.A
```

```
SELECT S.A, 1.0-prod(1.0 - S.p)
FROM    S
GROUP BY S.A
```

| A | P |
|---|---|
| a1 | 1 - (1-q1)*(1-q2) |
| a2 | 1 - (1-q3)*(1-q4)*(1-q5) |

$\bowtie^{i}$

R(A) S(A,B)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

$\Pi^{i}_{A}$

S(A,B)

| A | B | P |
|---|---|---|
| a1 | b1 | q1 |
| a1 | b2 | q2 |
| a2 | b3 | q3 |
| a2 | b4 | q4 |
| a2 | b5 | q5 |

```
create or replace
  function combine_prod(float, float)
  returns float as
  'select $1 * $2' language SQL;
create or replace
  function final_prod(float)
  returns float as
  'select $1' language SQL;
drop aggregate if exists prod (float);
create aggregate prod(float)
(   sfunc = combine_prod,
    stype = float,
    finalfunc = final_prod,
    initcond = '1.0'
);
```

# Extensional Plans in Postgres

```
SELECT DISTINCT ‘true’
FROM R, S
WHERE R.x = S.x
```

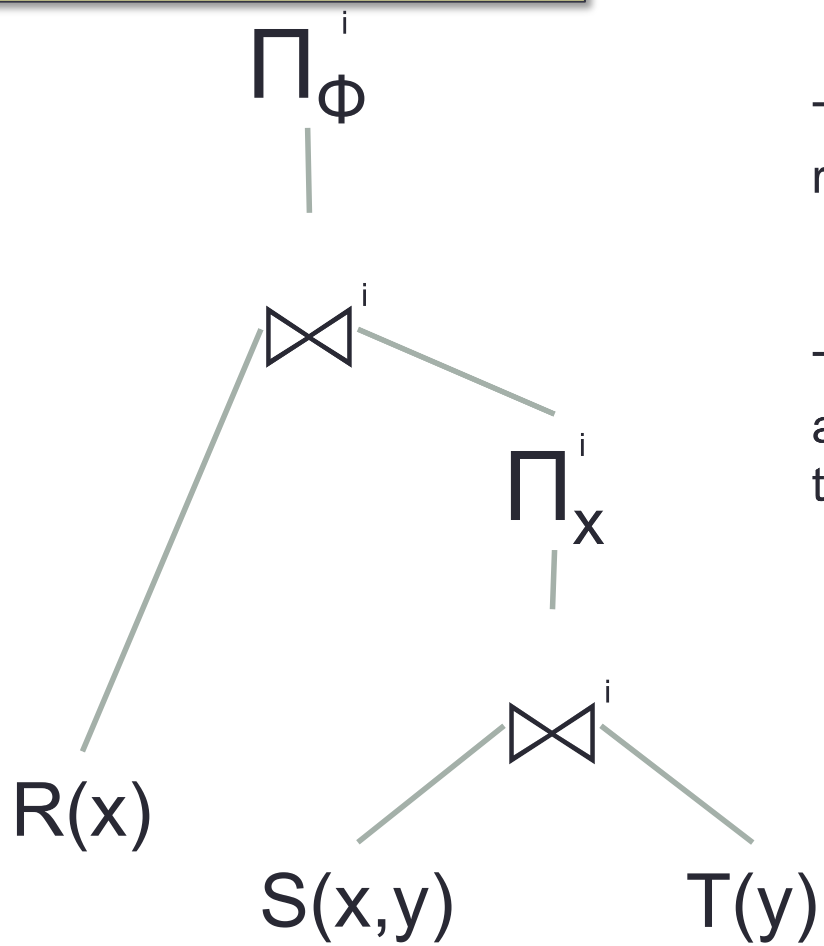

```
WITH Temp AS
   (SELECT S.x, 1.0-prod(1.0 - S.p) as p
    FROM S
    GROUP BY S.x)
SELECT ‘true’ as z, 1.0-prod(1.0 – R.P * Temp.P) as p
FROM R, Temp
WHERE R.x = Temp.x
```

# Try this in postgres:

```
-----------------------------------------------------
-- First step: download postgres from http://www.postgresql.org/
-- Second step: run the command "createdb pdb"
-- Third step: run the command "psql pdb" then cut/paste commands below
-----------------------------------------------------
-- define an aggregate function to compute the product
create or replace function combine_prod (float, float) returns float as 'select $1 * $2' language SQL;
create or replace function final_prod (float) returns float as 'select $1' language SQL;
drop aggregate if exists prod (float);
create aggregate prod (float)
(   sfunc = combine_prod,
    stype = float,
    finalfunc = final_prod,
    initcond = '1.0'
);

-----------------------------------------------------
-- simple tables, similar to those used in the tutorial
create table R(z char(8), x char(8), p float);
create table S(x char(8), y char(8), p float);

insert into R values('c', 'a1', 0.5);
insert into R values('c', 'a2', 0.5);
insert into R values('c', 'a3', 0.5);

insert into S values('a1', 'b1', 0.5);
insert into S values('a1', 'b2', 0.5);
insert into S values('a2', 'b2', 0.5);
insert into S values('a2', 'b3', 0.5);
insert into S values('a2', 'b4', 0.5);

-- computing the query Q(z) = R(z,x),S(x,y)
-- a safe plan:
with Temp as
  (select S.x, 1.0-prod(1.0-p) as p
   from S
   group by S.x)
select R.z, 1.0-prod(1-R.p*Temp.p)
from R, Temp
where R.x=Temp.x
group by R.z;

-- an unsafe plan; guaranteed to return an upper bound on the probability
select R.z, 1.0-prod(1-R.p*S.p)
from R, S
where R.x=S.x
group by R.z;
```

# Extensional Plans in Postgres

```
SELECT DISTINCT 'yes'
FROM R, S, T
WHERE R.x = S.x and S.y = T.y
```




This query is unsafe

The plan is unsafe, but guaranteed to return a probability that is an upper bound.

The same plan is guaranteed to return a lower bound, by modifying appropriately the probabilities in T

Gatterbauer,S.'2013

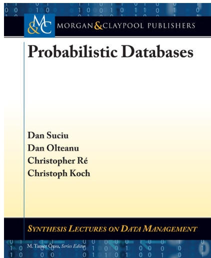
# Try this in postgres:

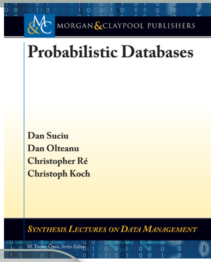
```
-----------------------------------------------------
-- The following approximation plans for unsafe queries are from
-- Gatterbauer, Suciu: Oblivious Bounds on the Probability of Boolean Functions

-- create a third table
create table T(y char(8), p float);

insert into T values('b1', 0.5);
insert into T values('b2', 0.5);
insert into T values('b3', 0.5);
insert into T values('b4', 0.5);

-- computing the query Q(z) = R(z,x),S(x,y),T(y)
-- This query has no safe plans

-- Next two unsafe plans compute upper bounds on the probability:
-- Unsafe plan #1
with Temp as
    (select S.x, 1.0-prod(1.0-S.p*T.p) as p
     from S,T
     where S.y=T.y
     group by S.x)
select R.z, 1.0-prod(1-R.p*Temp.p)
from R, Temp
where R.x=Temp.x
group by R.z;

-- Unsafe plan #2
with Temp as
   (select R.z,S.y,1.0-prod(1.0-R.p*S.p) as p
    from R,S
    where R.x=S.x
    group by R.z,S.y)
select Temp.z, 1.0-prod(1-Temp.p*T.p)
from Temp, T
where Temp.y=T.y
group by Temp.z;
```

```
-- Next two unsafe plans compute lower bounds on the probability:
with newT as
  (select T.y, 1-exp((ln(1-T.p))/count(*)) as p
   from S,T
   where S.y=T.y
   group by T.y, T.p),
Temp as
    (select S.x, 1.0-prod(1.0-S.p*newT.p) as p
     from S,newT
     where S.y=newT.y
     group by S.x)
select R.z, 1.0-prod(1-R.p*Temp.p)
from R, Temp
where R.x=Temp.x
group by R.z;

with newR as
  (select R.z, R.x, 1-exp((ln(1-R.p))/count(*)) as p
   from R,S
   where R.x=S.x
   group by R.z,R.x,R.p),
Temp as
    (select newR.z, S.y, 1.0-prod(1.0-newR.p*S.p) as p
     from newR, S
     where newR.x=S.x
     group by newR.z, S.y)
select Temp.z, 1.0-prod(1-Temp.p*T.p)
from Temp, T
where Temp.y=T.y
group by Temp.z;
```

# Lesson 2

- You don't need a probabilistic database system in order to use a probabilistic database!
- What you need is to know really well SQL and probability theory
- (You also need to read the book on probabilistic databases!)




MORGAN & CLAYPOOL PUBLISHERS
Probabilistic Databases
Dan Suciu
Dan Olteanu
Christopher Ré
Christoph Koch
SYNTHESIS LECTURES ON DATA MANAGEMENT

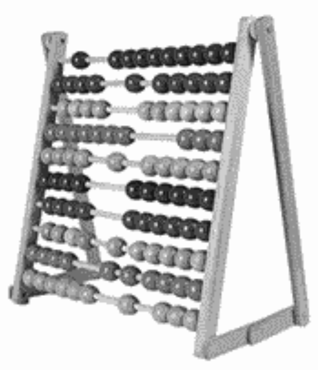
# Outline

# The Model Counting Problem

- Given a Boolean Formula F, compute the number of satisfying assignments #F
  - If F has n Boolean variables, then $0 \leq \#F \leq 2^n$

- SAT is the problem:
  Given F, check if there exists a satisfying assignment
  SAT is NP-complete

- #SAT is the model counting problem:
  Given F, compute #F
  #SAT is #P-complete



- We can reduce SAT to #SAT

- Probability computation is the problem:
  if each Boolean variable $X_i$ is set to true with probability $p_i$, compute $P(F)$, the probability that F is true
  Also #P-complete

# Example

$$F = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$$

The Model Counting Problem (#SAT): Compute #F

# Example

$$F = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$$

The Model Counting Problem (#SAT): Compute #F

$P(X_1) = p_1$, $P(X_2) = p_2$,
$P(Y_1) = q_1$, $P(Y_2) = q_2$, $P(Y_3) = q_3$, $P(Y_4) = q_4$

The Probability Computation Problem: Compute $P(F)$

# Example

$$F = X_1Y_1 \lor X_1Y_2 \lor X_2Y_3 \lor X_2Y_4$$

The Model Counting Problem (#SAT): Compute #F

$P(X_1) = p_1$, $P(X_2) = p_2$,
$P(Y_1) = q_1$, $P(Y_2) = q_2$, $P(Y_3) = q_3$, $P(Y_4) = q_4$

The Probability Computation Problem: Compute $P(F)$

Re-group: $F = [X_1 (Y_1 \lor Y_2)] \lor [X_2 (Y_3 \lor Y_4)]$

$$P(F)=1 - [1-p_1(1-(1-q_1)(1-q_2))]\,[1-p_2(1-(1-q_3)(1-q_4))]$$

# Example

$$F \quad = \quad X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$$

The Model Counting Problem (#SAT): Compute #F

$P(X_1) = p_1$, $P(X_2) = p_2$,
$P(Y_1) = q_1$, $P(Y_2) = q_2$, $P(Y_3) = q_3$, $P(Y_4) = q_4$

The Probability Computation Problem: Compute P(F)

Re-group: $F = [X_1 (Y_1 \vee Y_2)] \vee [X_2 (Y_3 \vee Y_4)]$

$$P(F)=1 - [1-p_1(1-(1-q_1)(1-q_2))]\,[1-p_2(1-(1-q_3)(1-q_4))]$$

$\#F = 2^6\, P(F)$, when $p_1 = \ldots = q_4 = 0.5$
$\#F = 34$

Now let's try this:

$$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$$

Now let's try this:

$$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$$

No clever grouping seems possible.
Use brute force:

| $X_1$ | $X_2$ | $X_3$ | F | P |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $(1-p_1)p_2p_3$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $p_1(1-p_2)p_3$ |
| 1 | 1 | 0 | 1 | $p_1p_2(1-p_3)$ |
| 1 | 1 | 1 | 1 | $p_1p_2p_3$ |

Now let's try this:

$$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$$

No clever grouping seems possible.
Use brute force:

| $X_1$ | $X_2$ | $X_3$ | F | P |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | $(1-p_1)p_2p_3$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $p_1(1-p_2)p_3$ |
| 1 | 1 | 0 | 1 | $p_1p_2(1-p_3)$ |
| 1 | 1 | 1 | 1 | $p_1p_2p_3$ |

$$P(F)=(1-p_1)p_2p_3 + p_1(1-p_2)p_3 + p_1p_2(1-p_3) + p_1p_2p_3$$

$$\#F = 4$$

# Complexity of Model Counting

SAT is the problem: Given F, check if there exists a satisfying assignment

#SAT is the problem: Given F, compute #F

**Theorem** [Valiant:1979] #SAT is #P-complete

NP = class of problems of the form "is there a witness ?" SAT
#P = class of problems of the form "how many witnesses ?" #SAT

Interesting fact:
The decision problem 2CNF is in PTIME
The counting problem #2CNF is #P-complete

# PP2DNF Formulas

**Definition**. A Positive, Partitioned 2DNF (PP2DNF) formula is:

$F = X_{i1} Y_{j1} \vee X_{i1} Y_{j1} \vee \ldots \vee X_{in} Y_{jn}$

Example: $F = X_1 Y_3 \vee X_2 Y_1 \vee X_2 Y_3 \vee X_3 Y_2$

**Theorem** [Provan and Ball 1982]
Model counting for PP2DNF is #P-complete.

Even for such simple formulas, counting the number of models is #P-complete!

# Queries are #P-Complete

**Theorem**. [Dalvi&S.04] Consider the query $H_0$ = R(x),S(x,y),T(y)
The problem: given a probabilistic database D, compute P($H_0$)
is #P-complete in the size of D

Proof: by reduction from #PP2DNF

Example: suppose $F = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_2$

R

| X | P |
|---|---|
| x1 | 0.5 |
| x2 | 0.5 |

S

| X | Y |
|---|---|
| x1 | y1 |
| x1 | y2 |
| x2 | y2 |

T

| Y | P |
|---|---|
| y1 | 0.5 |
| y2 | 0.5 |

Then P(F) = P($H_0$)

# Where We Are

- Safe queries have safe plans: in PTIME
- Unsafe queries have no safe plans: provably #P-complete
- Problem: for a given query, determine if it is safe.
  - Next: Conjunctive Queries without Self-joins
  - Later: Unions of Conjunctive Queries

# Review: Conjunctive Queries

A conjunctive query:

$$Q(z) = \exists x\ \exists t\ .(\text{Owner}(z,x) \wedge \text{Location}(x,t,\text{"Office444"}))$$

Same as:

$$Q(z) = \text{Owner}(z,x),\ \text{Location}(x,t,\text{"Office444"})$$

# Review: Conjunctive Queries

A conjunctive query:

$$Q(z) = \exists x\ \exists t\ .(\mathrm{Owner}(z,x) \land \mathrm{Location}(x,t,\text{"Office444"}))$$

Same as:

atom atom

$$Q(z) = \mathrm{Owner}(z,x),\ \mathrm{Location}(x,t,\text{"Office444"})$$

z=head variable

x, t = existential variables

# Review: Conjunctive Queries

A conjunctive query:

$Q(z) = \exists x\ \exists t .(\text{Owner}(z,x) \wedge \text{Location}(x,t,\text{"Office444"}))$

Same as:

atom

atom

$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,\text{"Office444"})$

z=head variable

x, t = existential variables

A conjunctive queries has self-joins if it has repeated atoms:

$Q() = R(z,x_1),S(z,x_1,y_1), T(z,x_2),S(z,x_2,y_2)$

$Q() = R(x,y),R(y,z)$

# Review: Conjunctive Queries

A conjunctive query:

$$Q(z) = \exists x\ \exists t\ .(\text{Owner}(z,x) \wedge \text{Location}(x,t,\text{"Office444"}))$$

Same as:

atom atom

$$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,\text{"Office444"})$$

z=head variable

x, t = existential variables

A conjunctive queries has self-joins if it has repeated atoms:

$$Q() = R(z,x_1),S(z,x_1,y_1), T(z,x_2),S(z,x_2,y_2)$$

$$Q() = R(x,y),R(y,z)$$

Conjunctive queries without self-joins (no repeated atoms):

$$Q() = R(x),S(x,y)$$

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

# Hierarchical Queries

at(x) = set of atoms containing the variable x in a key position

**Definition** A query Q is hierarchical if forall existential variables x, y:

$at(x) \subseteq at(y)$ or $at(x) \subseteq at(y)$ or $at(x) \cap at(y) = \emptyset$

# Hierarchical Queries

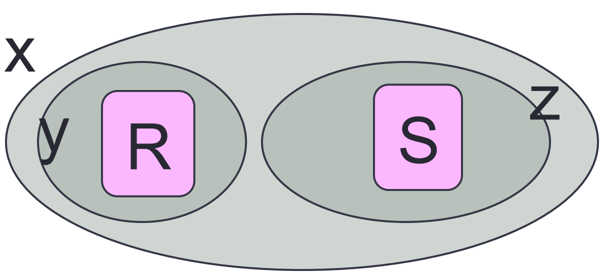
at(x) = set of atoms containing the variable x in a key position

**Definition** A query Q is hierarchical if forall existential variables x, y:

$$at(x) \subseteq at(y) \quad \text{or} \quad at(x) \subseteq at(y) \quad \text{or} \quad at(x) \cap at(y) = \varnothing$$

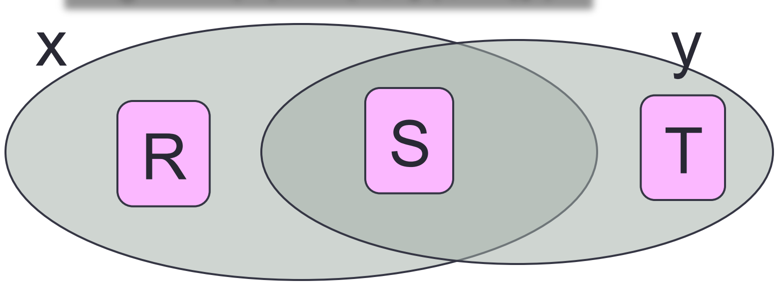
Hierarchical

Q = R(x,y),S(x,z)




Non-hierarchical

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



# The Small Dichotomy Theorem

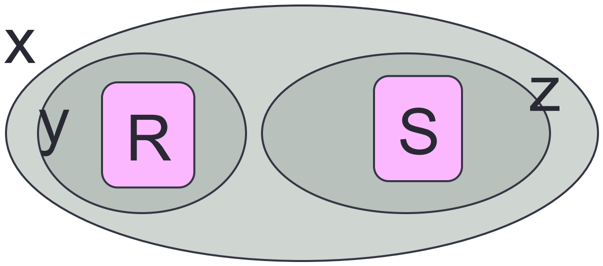
Fix a schema for probabilistic databases where all tuples are independent

**Theorem** [Dalvi&S.04] For every conjunctive query Q w/o self-joins:

- If Q is hierarchical, then computing P(Q) is in PTIME
- If Q is not hierarchical then computing P(Q) is #P-complete

Hierarchical

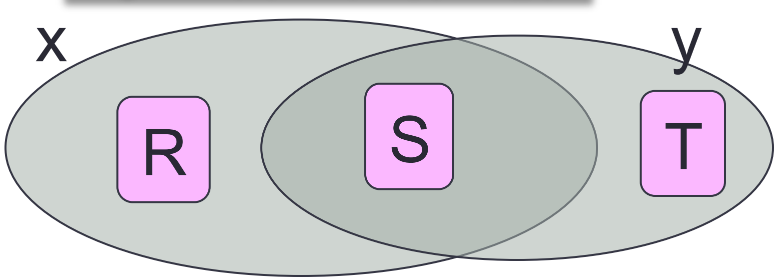
Q = R(x,y),S(x,z)




PTIME

Non-hierarchical

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




#P-complete

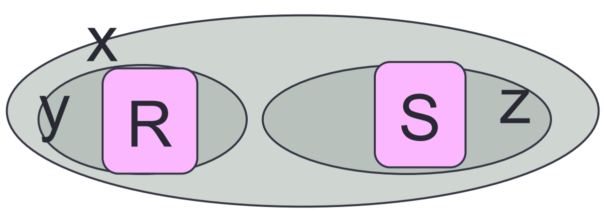
# Proof: Part I

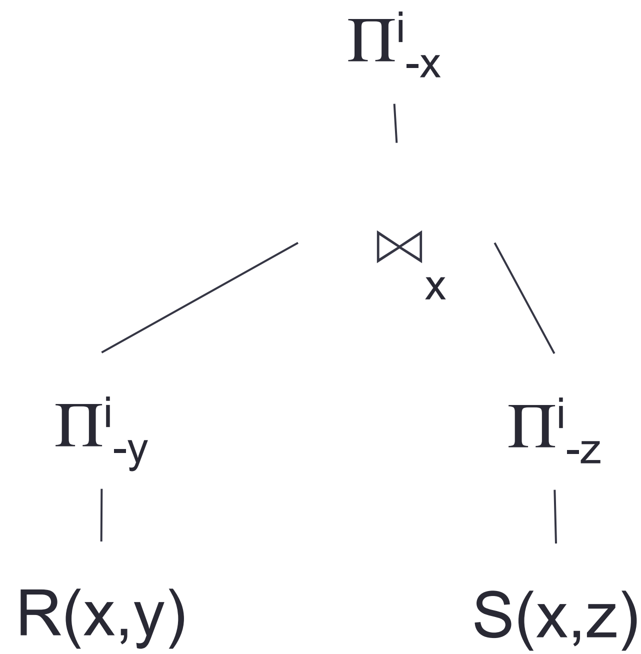
**Hierarchical → PTIME**

If Q is hierarchical, then use the query's hierarchy to derive a safe plan.

Example:

Q = R(x,y),S(x,z)



# Proof: Part II

Non-hierarchical → #P-hard

If Q is not hierarchical, then there exists x,y:

$at(x) \not\subseteq at(y)$,
$at(x) \cap at(y) \neq \emptyset$
$at(x) \not\supseteq at(y)$

→

There exists atoms R, S, T:

$R \in at(x) - at(y)$,
$S \in at(x) \cap at(y)$,
$T \in at(y) - at(x)$

↓

Q = … R(x, …), S(x,y,…), T(y,…), …

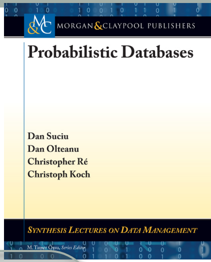
…and we use a reduction from $H_0$

# Summary of the Complexity

- Safe queries are in PTIME
- Unsafe queries are #P-complete

- Small Dichotomy Theorem: classifies every query in one of these two classes, but only applies to Conjunctive Queries without Self-joins

- Big Dichotomy Theorem: applies to all Unions of Conjunctive Queries – will discuss next




# Outline

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

Part 3

5. Extensional Evaluation — Chapter 4.1

Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions

# Overview

- Review: Unions of Conjunctive Queries, UCQ
- Four simple rules for evaluating queries Q
- Big Dichotomy Theorem:
  1. If the rules succeed → Q is safe → in PTIME
  2. If the rules fail → Q is unsafe → #P-complete
- Compare to the Small Dichotomy Theorem, which applies only to conjunctive queries w/o self-joins:
  - Case 1 holds precisely when Q is hierarchical
  - Case 2 holds precisely when Q is not hierarchical

# Review: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

$$Q(z) = \exists x_1 \exists t_1 \,(\text{Owner}(z,x_1) \land \text{Location}(x_1,t_1,\text{"Office444"})) \lor \exists x_2 \exists t_2 \,(\text{Owner}(z,x_2) \land \text{Location}(x_2,t_2,\text{"Hall7"}))$$

Same as:

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{"Office444"}) \lor \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{"Hall7"})$$

# Review: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

$$Q(z) = \exists x_1 \exists t_1 (\text{Owner}(z,x_1) \land \text{Location}(x_1,t_1,\text{"Office444"})) \lor \exists x_2 \exists t_2 (\text{Owner}(z,x_2) \land \text{Location}(x_2,t_2,\text{"Hall7"}))$$

Same as:

Union of conjunctive queries

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{"Office444"}) \lor \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{"Hall7"})$$

# Review: Unions of Conjunctive Queries

Owners of items in either "Office444" or "Hall7":

$$\textcolor{red}{Q}(z) = \exists x_1 \exists t_1\, (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{"Office444"})) \vee \exists x_2 \exists t_2\, (\text{Owner}(z,x_2) \wedge \text{Location}(x_2,t_2,\text{"Hall7"}))$$

Same as:

Union of conjunctive queries

$$\textcolor{red}{Q}(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{"Office444"}) \vee \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{"Hall7"})$$

Same as:

$$\textcolor{red}{Q}(z) = \text{Owner}(z,x) \wedge \exists t\, [\text{Location}(x,t,\text{"Office444"}) \vee \text{Location}(x,t,\text{"Hall7"})]$$

# Review: Unions of Conjunctive Queries

Owners of items in either "Office444" or "Hall7":

$$Q(z) = \exists x_1 \exists t_1 (Owner(z,x_1) \wedge Location(x_1,t_1,\text{"Office444"})) \vee \exists x_2 \exists t_2 (Owner(z,x_2) \wedge Location(x_2,t_2,\text{"Hall7"}))$$

Same as:

Union of conjunctive queries

$$Q(z) = Owner(z,x_1),Location(x_1,t_1,\text{"Office444"}) \vee Owner(z,x_2),Location(x_2,t_2,\text{"Hall7"})$$

Same as:

$$Q(z) = Owner(z,x) \wedge \exists t\, [Location(x,t,\text{"Office444"}) \vee Location(x,t,\text{"Hall7"})]$$

We will use these laws:

1. Distributivity law for $\vee$, $\wedge$
2. Commutativity law for $\exists$, $\vee$: $(\exists x\, P(x)) \vee (\exists y\, T(y)) = \exists z\, (P(z) \vee T(z))$

# Four Rules for Computing Query Probabilities

- Independent join
- Independent project
- Independent union
- Inclusion/exclusion

Rules apply to Boolean Queries only

# Rule 1: Independent Join

$$P(Q1 \wedge Q2) = P(Q1)P(Q2)$$

If Q1 and Q2 are independent
(meaning: no common atoms)

## Rule 1: Independent Join

$$P(Q1 \land Q2) = P(Q1)P(Q2)$$

If Q1 and Q2 are independent
(meaning: no common atoms)

## Rule 2: Independent Project

$$P(\exists z\, Q) = 1 - \Pi_{a \in \mathrm{Domain}} (1 - P(Q[a/z]))$$

If z is a "separator variable" in Q, meaning that for any constants a,b, Q[a/z] and Q[b/z] are independent

## Rule 1: Independent Join

$$P(Q1 \wedge Q2) = P(Q1)P(Q2)$$

If Q1 and Q2 are independent
(meaning: no common atoms)

## Rule 2: Independent Project

$$P(\exists z\, Q) = 1 - \Pi_{a \in \mathrm{Domain}} (1 - P(Q[a/z]))$$

If z is a "separator variable" in Q,
meaning that for any constants a,b,
Q[a/z] and Q[b/z] are independent

## Rule 3: Independent Union

$$P(Q1 \vee Q2) = 1 - (1 - P(Q1))(1 - P(Q2))$$

If Q1 and Q2 are independent
(meaning: no common atoms)

# Example

$$\boxed{Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)}$$

$$= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$$

# Example

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$

Commute $\exists$ with $\vee$

# Example

$$Q_U = R(x_1),S(x_1,y_1) \lor T(x_2),S(x_2,y_2)$$

$= \exists x_1 \exists y_1 R(x_1) \land S(x_1,y_1) \lor \exists x_2 \exists y_2 T(x_2) \land S(x_2,y_2)$

$$Q_U = \exists z\, [R(z) \land S(z,y_1) \lor T(z) \land S(z,y_2)]$$

Commute $\exists$ with $\lor$

$$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[R(a) \land \underline{S(a,y_1)} \lor T(a) \land \underline{S(a,y_2)})]$$

Independent project: for $a \neq b$, $Q_U[a/z]$ and $Q_U[b/z]$ are independent because atoms $R(a),S(a,y_1),T(a),S(a,y_2)$ are distinct from $R(b),S(b,y_1),T(b),S(b,y_2)$

# Example

$Q_U = R(x_1),S(x_1,y_1) \lor T(x_2),S(x_2,y_2)$

$= \exists x_1 \exists y_1 R(x_1) \land S(x_1,y_1) \lor \exists x_2 \exists y_2 T(x_2) \land S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \land S(z,y_1) \lor T(z) \land S(z,y_2)]$

Commute $\exists$ with $\lor$

$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[R(a) \land \underline{S(a,y_1)} \lor T(a) \land \underline{S(a,y_2)})]$

Independent project: for $a \neq b$, $Q_U[a/z]$ and $Q_U[b/z]$ are independent because atoms $R(a),S(a,y_1),T(a),S(a,y_2)$ are distinct from $R(b),S(b,y_1),T(b),S(b,y_2)$

$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[\underline{(R(a) \lor T(a))} \land \underline{\exists y. S(a,y)}]$

Distribute $\land$ over $\lor$

# Example

$$Q_U = R(x_1),S(x_1,y_1) \lor T(x_2),S(x_2,y_2)$$

$= \exists x_1 \exists y_1 R(x_1) \land S(x_1,y_1) \lor \exists x_2 \exists y_2 T(x_2) \land S(x_2,y_2)$

$$Q_U = \exists z\, [R(z) \land S(z,y_1) \lor T(z) \land S(z,y_2)]$$

Commute $\exists$ with $\lor$

$$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[R(a) \land \underline{S(a,y_1)} \lor T(a) \land \underline{S(a,y_2)})]$$

Independent project: for $a \neq b$, $Q_U[a/z]$ and $Q_U[b/z]$ are independent because atoms $R(a),S(a,y_1),T(a),S(a,y_2)$ are distinct from $R(b),S(b,y_1),T(b),S(b,y_2)$

$$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[\underline{(R(a) \lor T(a))} \land \underline{\exists y.\, S(a,y)}]$$

Distribute $\land$ over $\lor$

$$P(Q_U) = 1 - \Pi_{a \in \text{Domain}} (1 - P[R(a) \lor T(a)]\, P[\exists y.\, S(a,y)]$$

Independent join

# Example

$\boxed{Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)}$ $= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$\boxed{Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]}$ Commute $\exists$ with $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[R(a) \wedge \underline{S(a,y_1)} \vee T(a) \wedge \underline{S(a,y_2)})]$

Independent project: for $a \neq b$, $Q_U[a/z]$ and $Q_U[b/z]$ are independent because atoms $R(a),S(a,y_1),T(a),S(a,y_2)$ are distinct from $R(b),S(b,y_1),T(b),S(b,y_2)$

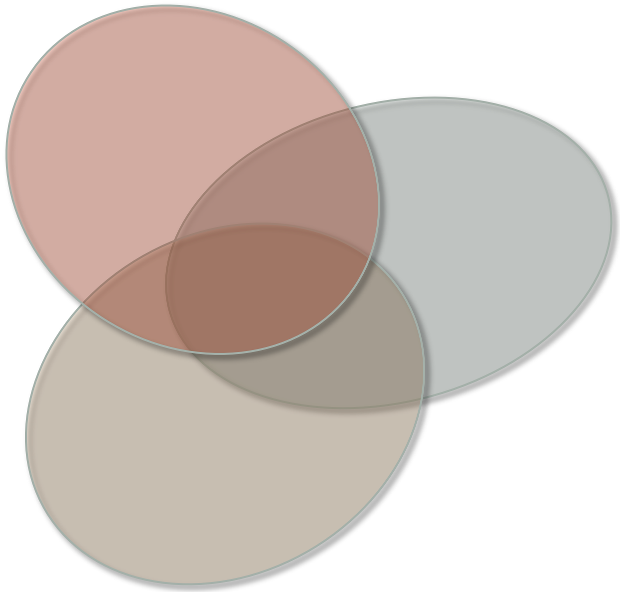
$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[\underline{(R(a) \vee T(a))} \wedge \underline{\exists y.\ S(a,y)}]$ Distribute $\wedge$ over $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[R(a) \vee T(a)]\ P[\exists y.\ S(a,y)]$ Independent join

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - (1-(1-P[R(a)])(1-P[T(a)]))\ (1-\Pi_{b \in Domain} (1 - P[S(a,b)])))$

# Rule 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$



Note: this is the dual of the more popular formula:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

# Example

$\boxed{Q_J = R(x_1),S(x_1,y_1),\ T(x_2),S(x_2,y_2)}$

$= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \ \wedge \ [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

# Example

$\boxed{Q_J = R(x_1),S(x_1,y_1),\ T(x_2),S(x_2,y_2)}$ $= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \ \wedge\ [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

$\boxed{Q_J = Q_1 \wedge Q_2}$ where $\boxed{Q_1 = R(x_1),S(x_1,y_1)}$ $\boxed{Q_2 = T(x_2),S(x_2,y_2)}$

# Example

$\boxed{Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)}$ $= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \wedge [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

$\boxed{Q_J = Q_1 \wedge Q_2}$ where $\boxed{Q_1 = R(x_1),S(x_1,y_1)}$ $\boxed{Q_2 = T(x_2),S(x_2,y_2)}$

$$P(Q_J) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$$

$Q_1$ = a hierarchical conjunctive query w/o self-joins

$Q_2$ = similar

$Q_1 \vee Q_2 = Q_U$, which have see a couple of slides ago

# Lesson 3

We need unions in order to handle self-joins!

- Conjunctive Queries = not a “natural” class of queries for Probabilistic DBs

- Unions of Conjunctive Queries = the “natural” class of queries

# Unsafe Queries – When the Rules Fail

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

# Unsafe Queries – When the Rules Fail

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

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

$= \exists z\ [R(z) \wedge S(z,y_0) \vee S(x_1,z) \wedge T(z)]$

Unlike $Q_U$, here z occurs on different positions in S and we cannot apply Independent Project

# Unsafe Queries – When the Rules Fail

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

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

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

# Unsafe Queries – When the Rules Fail

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

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

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

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

. . .

# Unsafe Queries – When the Rules Fail

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

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

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

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

. . .

**Theorem.** Each query $H_k$ is #P-hard

The proof is in [Dalvi&S, JACM’2012]

# The Amazing Queries $H_k$

$H_k$ is #P-hard.
But if we drop any one conjunctive query, then it is in PTIME

$$H_3 = R(x_0),S_1(x_0,y_0) \vee \cancel{S_1(x_1,y_1),S_2(x_1,y_1)} \vee S_2(x_2,y_2),S_3(x_2,y_2) \vee S_3(x_3,y_3),T(y_3)$$

Independent union

$$= \exists z\, [S_2(x_2,z),S_3(x_2,z) \vee S_3(x_3,z),T(z)]$$
$$= \exists z\, [\exists x_3\, S_3(x_3,z)] \wedge [(\exists x_2 S_2(x_2,z)) \vee T(z)]$$
$$= \text{etc}$$

# Where We Are

- We have seen examples of unsafe queries: $H_k$
  - But if a query Q has $H_k$ as a subquery, it is not necessarily unsafe
- When the four rules succeed, then Q is safe
  - But inclusion/exclusion is insufficient: need to replace with Mobius inversion formula

We will discuss these issues
then state the Big Dichotomy Theorem

# A Safe Query with $H_1$ as Subquery

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

# A Safe Query with $H_1$ as Subquery

Disconnected query

$$Q_V = \underbrace{R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2)}_{= H_1 \text{ (unsafe!)}} \vee R(x_3),T(y_3)$$

# A Safe Query with $H_1$ as Subquery

Disconnected query

DNF

$$Q_V = \underbrace{R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2)}_{= H_1 \text{ (unsafe!)}} \vee R(x_3),T(y_3)$$

CNF

$$Q_V = [S(x_2,y_2),T(y_2) \vee R(x_3)] \wedge [R(x_1),S(x_1,y_1) \vee T(y_3)]$$

# A Safe Query with $H_1$ as Subquery

DNF

= $H_1$ (unsafe!)

Disconnected query

$$Q_V = \underbrace{R(x_1),S(x_1,y_1) \lor S(x_2,y_2),T(y_2)}_{= H_1 \text{ (unsafe!)}} \lor R(x_3),T(y_3)$$

CNF

$$Q_V = [S(x_2,y_2),T(y_2) \lor R(x_3)] \land [R(x_1),S(x_1,y_1) \lor T(y_3)]$$

Inclusion/exclusion:

PTIME !

$$P(Q_V) = P(q_1 \land q_2) = P(q_1) + P(q_2) - P(q_1 \lor q_2)$$

$= R(x_3) \lor T(y_3)$

# Inclusion/Exclusion is Insufficient

$$
\begin{aligned}
Q_W = & [R(x_0),S_1(x_0,y_0) \lor S_2(x_2,y_2),S_3(x_2,y_2)] \land && \text{/* Q1 */} \\
& [R(x_0),S_1(x_0,y_0) \lor S_3(x_3,y_3),T(y_3)] \land && \text{/* Q2 */} \\
& [S_1(x_1,y_1),S_2(x_1,y_1) \lor S_3(x_3,y_3),T(y_3)] && \text{/* Q3 */}
\end{aligned}
$$

# Inclusion/Exclusion is Insufficient

$$
\begin{aligned}
Q_W = {} & [R(x_0),S_1(x_0,y_0) \vee S_2(x_2,y_2),S_3(x_2,y_2)] \wedge && \text{/* Q1 */} \\
& [R(x_0),S_1(x_0,y_0) \vee S_3(x_3,y_3),T(y_3)] \wedge && \text{/* Q2 */} \\
& [S_1(x_1,y_1),S_2(x_1,y_1) \vee S_3(x_3,y_3),T(y_3)] && \text{/* Q3 */}
\end{aligned}
$$

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

$= H_3$ (hard !)

Also $= H_3$

# Inclusion/Exclusion is Insufficient

$$
\begin{aligned}
Q_W = {} & [R(x_0),S_1(x_0,y_0) \vee S_2(x_2,y_2),S_3(x_2,y_2)] \wedge && \text{/* Q1 */} \\
& [R(x_0),S_1(x_0,y_0) \vee S_3(x_3,y_3),T(y_3)] \wedge && \text{/* Q2 */} \\
& [S_1(x_1,y_1),S_2(x_1,y_1) \vee S_3(x_3,y_3),T(y_3)] && \text{/* Q3 */}
\end{aligned}
$$

PTIME

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

$= H_3$ (hard !)

Also $= H_3$

#P-hard

# Inclusion/Exclusion is Insufficient

$$
\begin{aligned}
Q_W = {} & [R(x_0),S_1(x_0,y_0) \vee S_2(x_2,y_2),S_3(x_2,y_2)] \wedge && /^*\ Q1\ ^*/ \\
& [R(x_0),S_1(x_0,y_0) \vee S_3(x_3,y_3),T(y_3)] \wedge && /^*\ Q2\ ^*/ \\
& [S_1(x_1,y_1),S_2(x_1,y_1) \vee S_3(x_3,y_3),T(y_3)] && /^*\ Q3\ ^*/
\end{aligned}
$$

PTIME

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

PTIME

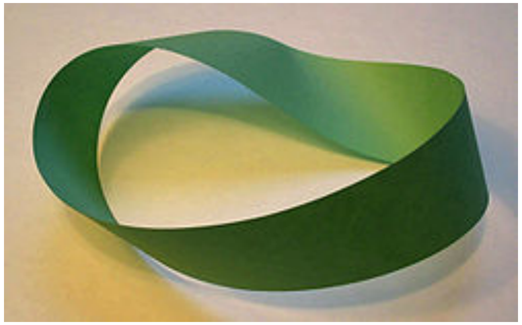
$= H_3$ (hard !)

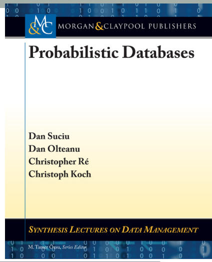
Also $= H_3$

#P-hard

# August Ferdinand Möbius 1790-1868

- Möbius strip
- Möbius function μ in number theory
- Generalized to lattices [Stanley'97,Rota'09]
- And now to queries !

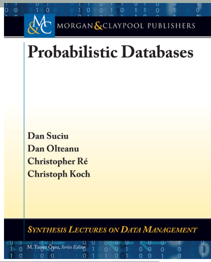





# The CNF Lattice

See formal definition in the book.

**Definition**. The CNF lattice of Q = Q1 ∧ Q2 ∧ … is:



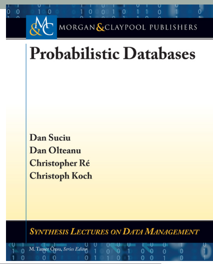

# The CNF Lattice

See formal definition in the book.

**Definition**. The CNF lattice of Q = Q1 ∧ Q2 ∧ … is:

## Example

$$
\begin{aligned}
Q_W = \; & [R(x_0),S_1(x_0,y_0) \;\vee\; S_2(x_2,y_2),S_3(x_2,y_2)] \;\wedge && \text{/* Q1 */}\\
& [R(x_0),S_1(x_0,y_0) \;\vee\; S_3(x_3,y_3),T(y_3)] \;\wedge && \text{/* Q2 */}\\
& [S_1(x_1,y_1),S_2(x_1,y_1) \;\vee\; S_3(x_3,y_3),T(y_3)] && \text{/* Q3 */}
\end{aligned}
$$



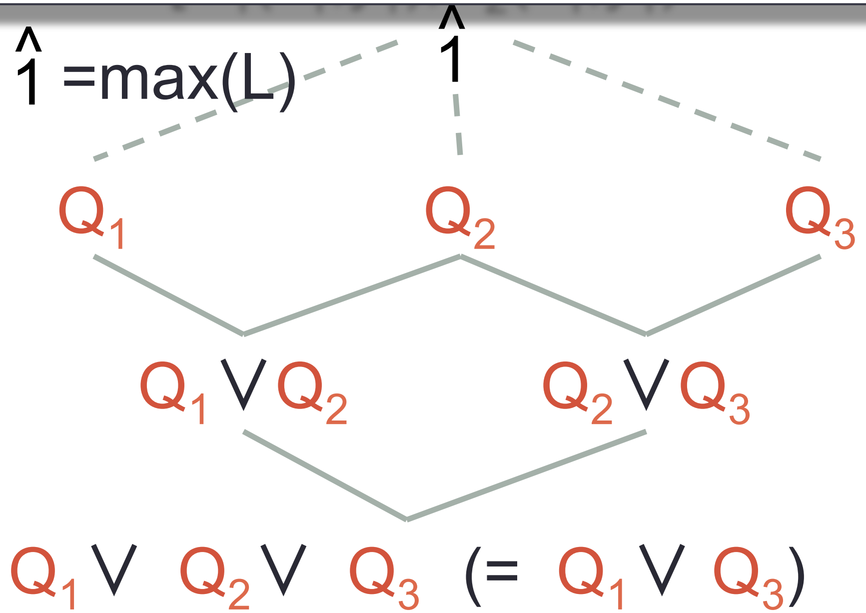

# The CNF Lattice

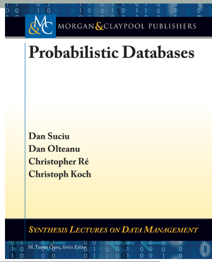
See formal definition in the book.

**Definition**. The CNF lattice of $Q = Q1 \wedge Q2 \wedge \ldots$ is:

## Example

$$
\begin{array}{llll}
Q_W = & [R(x_0),S_1(x_0,y_0) \vee & S_2(x_2,y_2),S_3(x_2,y_2)] \wedge & \text{/* Q1 */} \\
 & [R(x_0),S_1(x_0,y_0) \vee & S_3(x_3,y_3),T(y_3)] \wedge & \text{/* Q2 */} \\
 & [S_1(x_1,y_1),S_2(x_1,y_1) \vee & S_3(x_3,y_3),T(y_3)] & \text{/* Q3 */}
\end{array}
$$






# The CNF Lattice

See formal definition in the book.

**Definition**. The CNF lattice of $Q = Q1 \wedge Q2 \wedge \ldots$ is:

## Example

$$
\begin{aligned}
Q_W = {} & [R(x_0),S_1(x_0,y_0) \vee S_2(x_2,y_2),S_3(x_2,y_2)] \wedge && \text{/* Q1 */} \\
& [R(x_0),S_1(x_0,y_0) \vee S_3(x_3,y_3),T(y_3)] \wedge && \text{/* Q2 */} \\
& [S_1(x_1,y_1),S_2(x_1,y_1) \vee S_3(x_3,y_3),T(y_3)] && \text{/* Q3 */}
\end{aligned}
$$

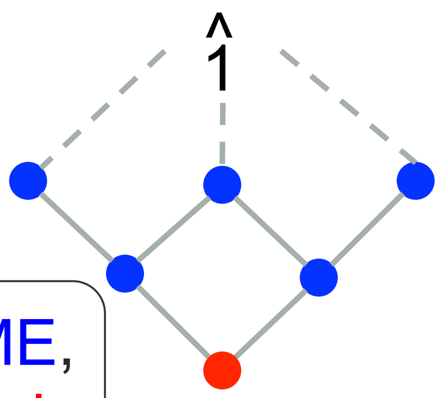
$\hat{1}$ =max(L)

$\hat{1}$

$Q_1$ $Q_2$ $Q_3$

$Q_1 \vee Q_2$ $Q_2 \vee Q_3$

$Q_1 \vee Q_2 \vee Q_3$ $(= Q_1 \vee Q_3)$




Nodes • in PTIME,
Nodes • #P hard.

# The Möbius' Function

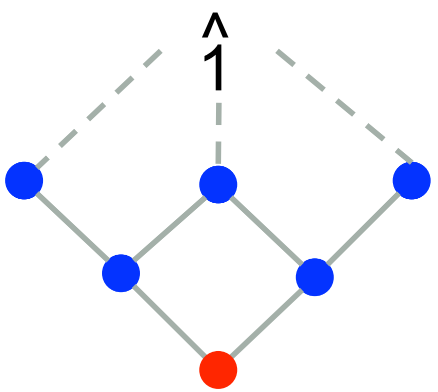
**Def**. The Möbius function:

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

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

**Möbius' Inversion Formula**:

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



# The Möbius' Function

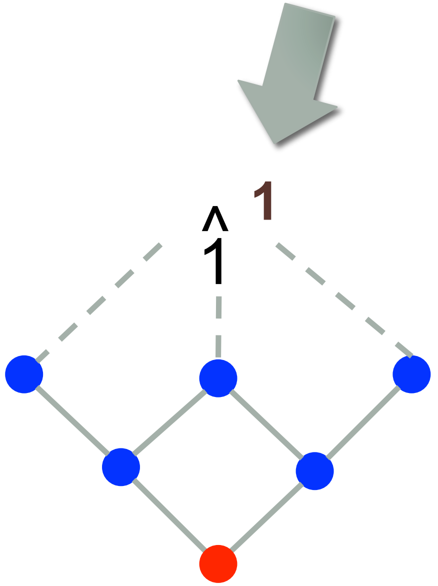
**Def**. The Möbius function:

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

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

**Möbius' Inversion Formula**:

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



# The Möbius' Function

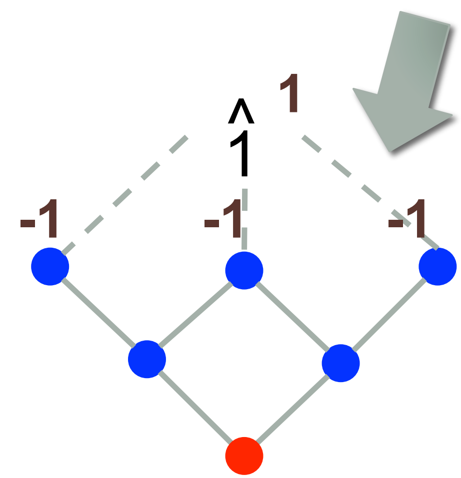
**Def**. The Möbius function:

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

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

**Möbius' Inversion Formula**:

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



# The Möbius' Function

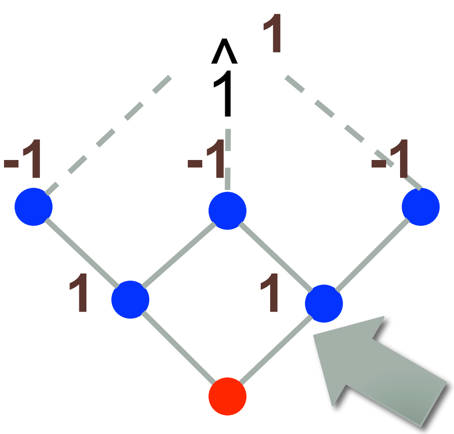
**Def**. The Möbius function:

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

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

**Möbius' Inversion Formula**:

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



# The Möbius' Function

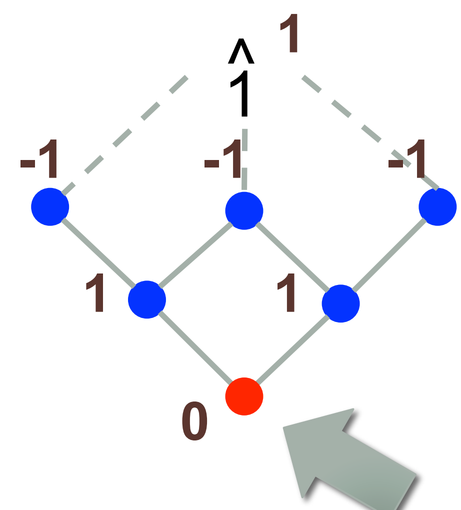
**Def**. The Möbius function:

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

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

**Möbius' Inversion Formula**:

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

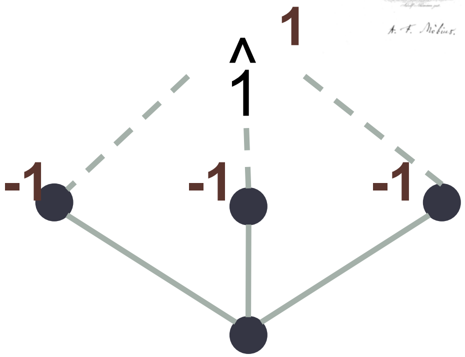


# The Möbius' Function


**Def.** The Möbius function:

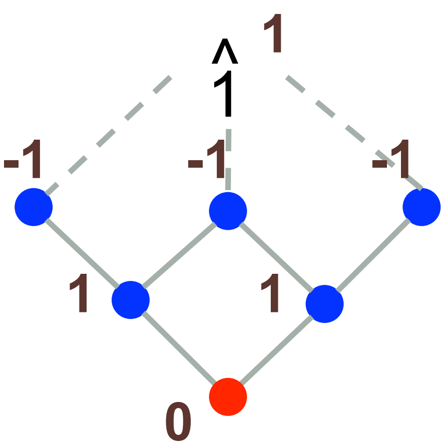
$\mu(\hat{1}, \hat{1}) = 1$

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




**Möbius' Inversion Formula**:

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

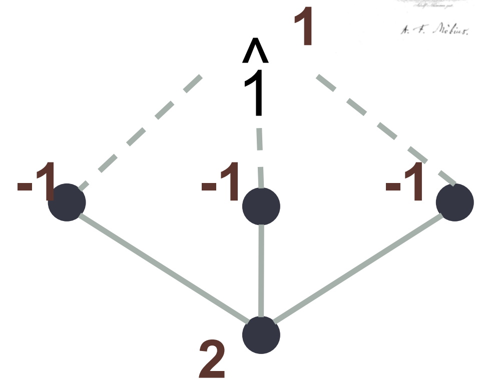


# The Möbius' Function

**Def.** The Möbius function:

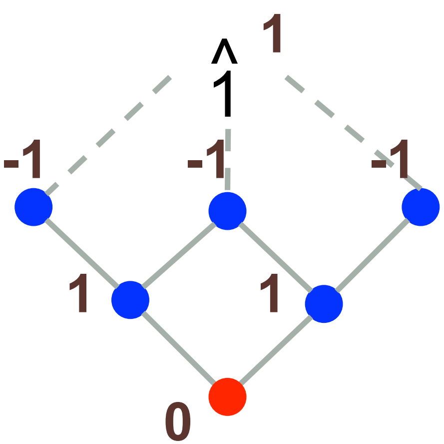
$\mu(\hat{1}, \hat{1}) = 1$

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




**Möbius' Inversion Formula:**

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

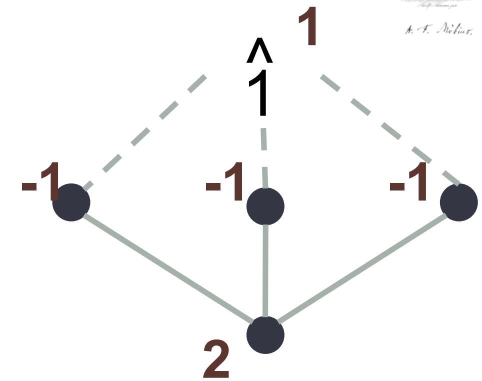


# The Möbius' Function

**Def.** The Möbius function:

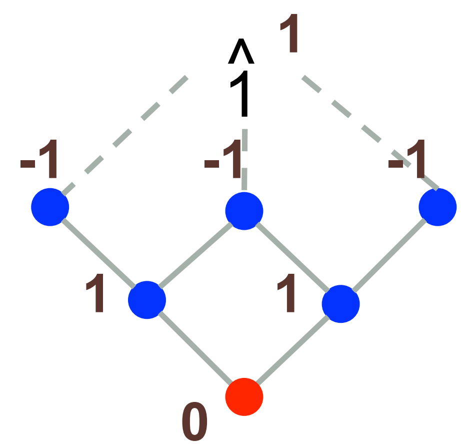
$\mu(\hat{1}, \hat{1}) = 1$

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




**Möbius' Inversion Formula**:

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




**<u>New Rule</u>**

Inclusion/Exclusion

→ Mobius' Inversion Formula

# The Big Dichotomy Theorem

**Dichotomy Theorem** Fix a UCQ query Q.
1. If rules terminates, then P(Q) is in PTIME
2. If rules fail, then P(Q) is #P-complete

The proof is in [Dalvi&S, JACM'2012]

Dichotomy into PTIME/#P-complete based on "syntax" where "syntax" includes the Mobius function !

# Lesson 5

- Four simple rules are all we need to compute query probabilities in PTIME:
  - Independent join
  - Independent project
  - Independent union
  - ~~Inclusion/Exclusion~~ → Mobius inversion formula

- Inclusion/exclusion is not used in modern model counting systems! It is specific to probabilistic databases

# Representation Theorem

Do we really need the lattice and Mobius function?

Yes! For every lattice on can construct a query Q s.t.:

- Q is in PTIME if $\mu=0$
- Q is #P-complete if $\mu\neq 0$

- This suggests that using the Mobius function is unavoidable in Probabilistic Databases

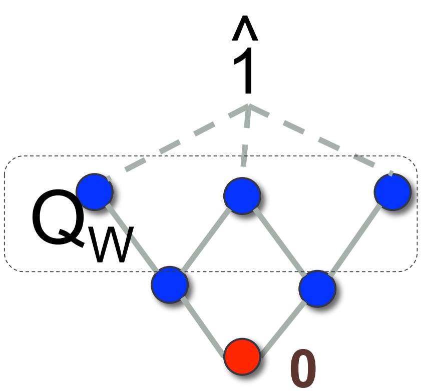
# Representation Theorem

**THEOREM** Every lattice L is the CNF lattice of a query Q, s.t.
- The query at $\hat{0}$ (= min(L)) is hard for #P
- All other queries are in PTIME

Q is in PTIME iff $\mu(\hat{0}, \hat{1})=0$ !

Examples:

PTIME !

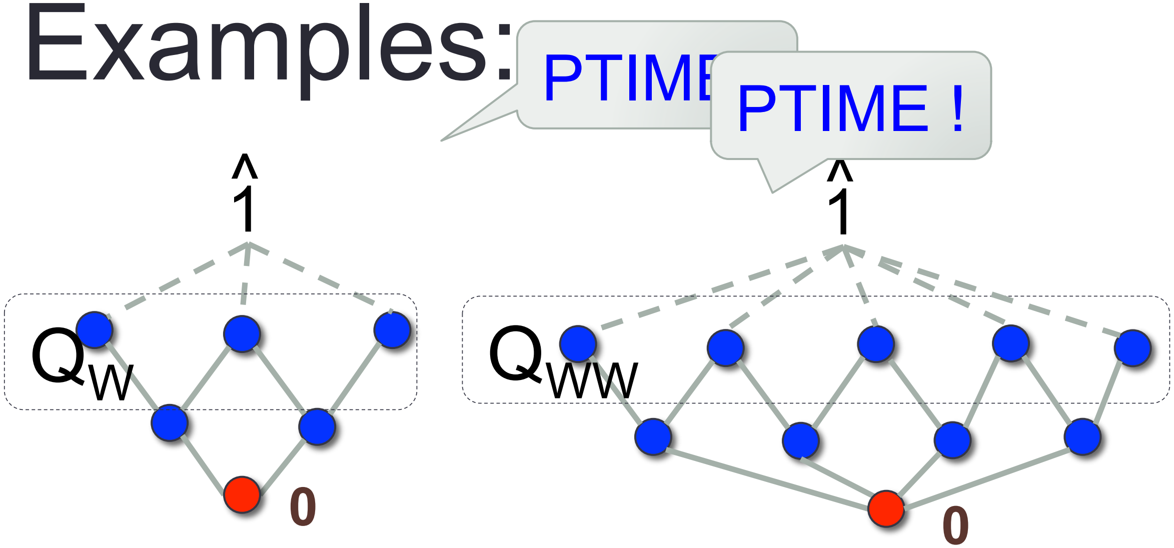


# Representation Theorem

**THEOREM** Every lattice L is the CNF lattice of a query Q, s.t.
- The query at $\hat{0}$ (= min(L)) is hard for #P
- All other queries are in PTIME

Q is in PTIME iff $\mu(\hat{0}, \hat{1})=0$ !

Examples:

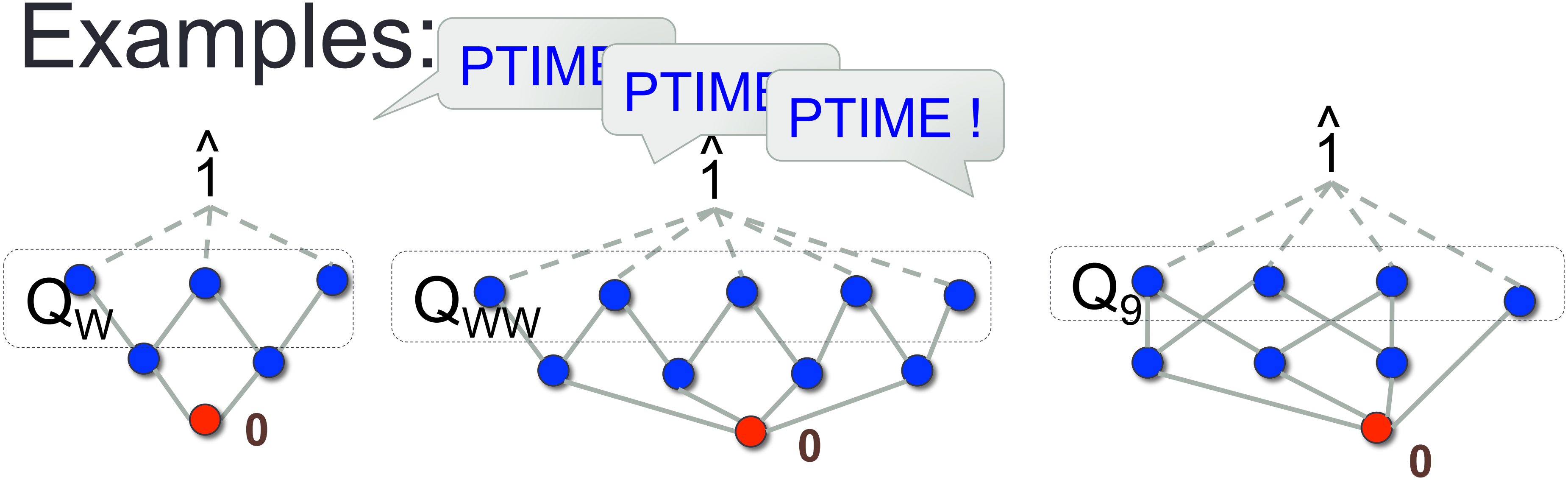


# Representation Theorem

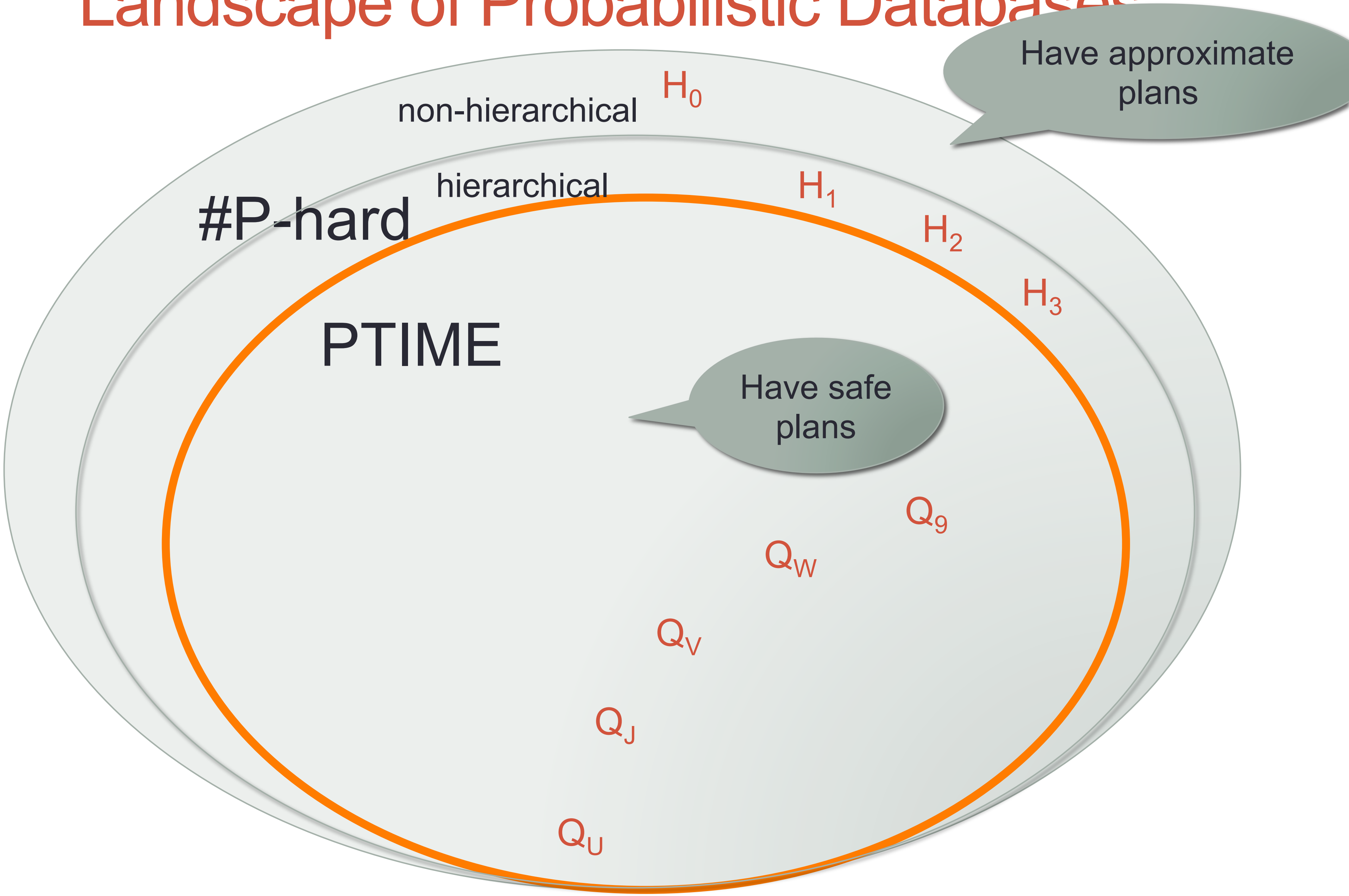
**THEOREM** Every lattice L is the CNF lattice of a query Q, s.t.
- The query at $\hat{0}$ (= min(L)) is hard for #P
- All other queries are in PTIME

Q is in PTIME iff $\mu(\hat{0}, \hat{1})=0$ !

Examples:



# Landscape of Probabilistic Databases



# Extensional Plans for UCQ

- Recall extensional operators for Conjunctive Queries w/o self-joins
  - Independent join: $\bowtie$
  - Independent projection $\Pi$
  - Selection $\sigma$
- Now we need two more operators:
  - Independent union: $\cup^{i}$
  - Mobius sum: $\Sigma^{\mu1,\mu2,\mu3}$

# Independent-Union and Mobius-Sum

| A | P |
|---|---|
| a1 | p1 |
| a2 | 1-(1-p2)(1-q2) |
| a3 | 1-(1-p3)(1-q3) |
| a4 | q4 |

```
SELECT 1.0 -
   (1.0 - (CASE
            WHEN R.p IS null THEN 0
            ELSE R.p END))*
   (1.0 - (CASE
            WHEN S.p IS null THEN 0
            ELSE S.p END))
FROM R full outer join S on r.x=s.x;
```

$\cup^{i}$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

T(A)

| A | P |
|---|---|
| a2 | q2 |
| a3 | q3 |
| a4 | q4 |

# Independent-Union and Mobius-Sum

| A | P |
|---|---|
| a1 | p1 |
| a2 | 1-(1-p2)(1-q2) |
| a3 | 1-(1-p3)(1-q3) |
| a4 | q4 |

```
SELECT 1.0 -
   (1.0 - (CASE
            WHEN R.p IS null THEN 0
            ELSE R.p END))*
   (1.0 - (CASE
            WHEN S.p IS null THEN 0
            ELSE S.p END))
FROM R full outer join S on r.x=s.x;
```

| A | P |
|---|---|
| a1 | μ1*p1+μ3*s1 |
| a2 | μ1*p2+μ3*q2+μ3*s2 |
| a3 | μ1*p3+μ3*q2+μ3*s3 |
| a4 | μ3*q4 |

```
SELECT …
-- long query
--  here
```

$\cup^{i}$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

T(A)

| A | P |
|---|---|
| a2 | q2 |
| a3 | q3 |
| a4 | q4 |

$\Sigma_{A}^{\mu1,\mu2,\mu3}$

R(A)

| A | P |
|---|---|
| a1 | p1 |
| a2 | p2 |
| a3 | p3 |

T(A)

| A | P |
|---|---|
| a2 | q2 |
| a3 | q3 |
| a4 | q4 |

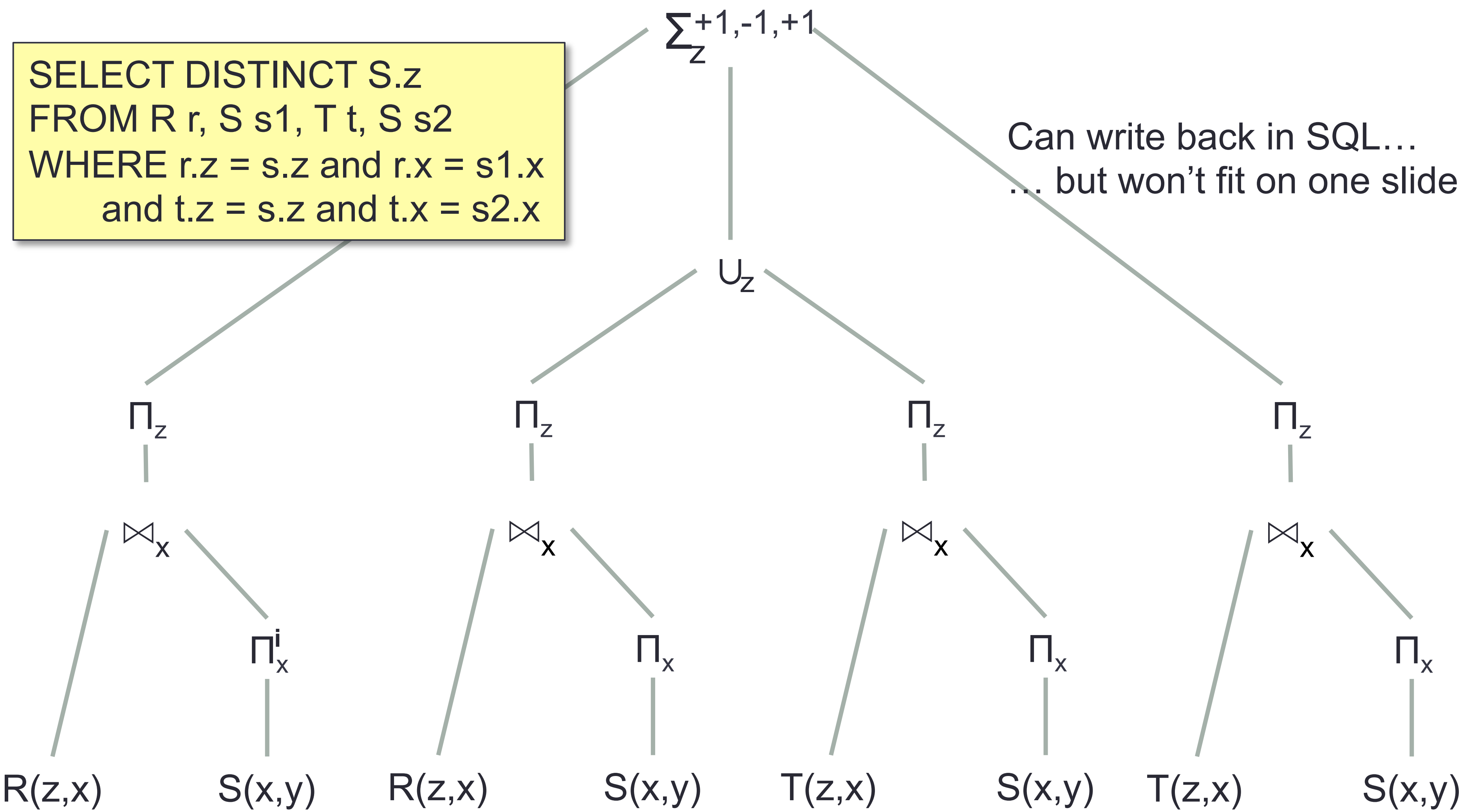
U(A)

| A | P |
|---|---|
| a1 | s1 |
| a3 | s3 |

# Extensional Plans for UCQ

```
SELECT DISTINCT S.z
FROM R r, S s1, T t, S s2
WHERE r.z = s.z and r.x = s1.x
      and t.z = s.z and t.x = s2.x
```

Can write back in SQL…
… but won't fit on one slide

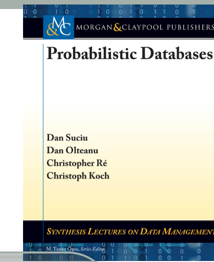


# Summary: Extensional Query Evaluation

- Four rules can evaluate all queries that are in PTIME
- Actually, a fifth rule is needed (ranking), see book

- Big Dichotomy Theorem:
  - If the rules succeed → query is safe → in PTIME
  - If the rules fail → query is unsafe → #P-complete

- Inclusion/exclusion is specific to probabilistic databases, not used by modern model counters: will discuss next.




# Outline

Part 1

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

Part 2

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

Part 3

5. Extensional Evaluation — Chapter 4.1

Part 4

6. Intensional Evaluation — Chapter 5
7. Conclusions

# Review: Model Counting

- Model Counting Problem: given F, compute #F
- Probability Computation Problem: given F, compute P(F)

- #P-complete

- Exact solvers – based on the DPLL procedure
- Approximate solvers – will not discuss in this tutorial

- Question: what performance guarantees offer DPLL-based algorithm for query evaluating on probabilistic databases?

# Background: DPLL

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

Based on Davis, Putnam, Logemann, Loveland, from the 60s; [Gomes et al., 2009]

Modern model counting systems are based on DPLL

- `c2d` [Huang and Darwiche, 2007], `Dsharp` [Muise et al., 2012]

Usually F is in CNF, then, add this simple heuristics:
favor a variable X that occurs only (un-)negated.

We will apply DPLL to positive DNF formulas F

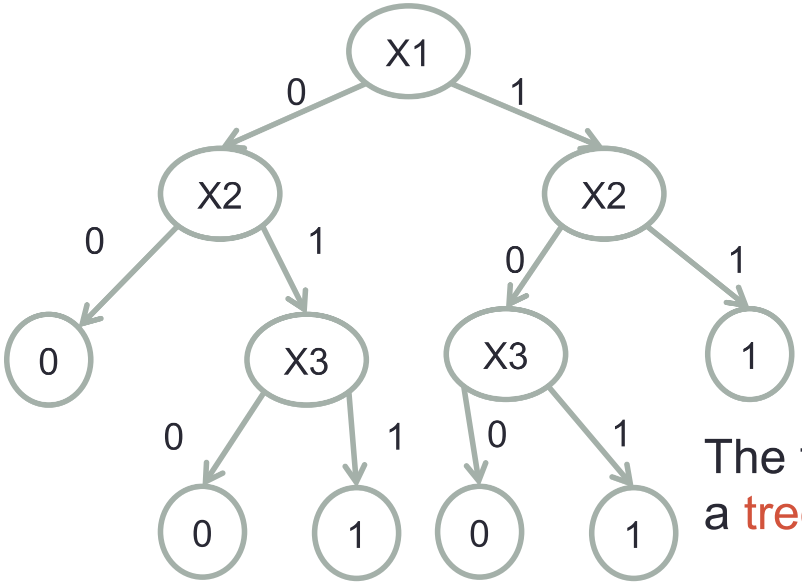
# The trace of DPLL

- The trace of a DPLL algorithm on a Boolean function F is a Read-Once-Branching-Program (RBOC), also called a Free Binary Decision Diagram (FBDD)
- The trace is used in:
  - Knowledge compilation: constructs the trace explicitly
  - Lower bounds for DPLL-based algorithms: we do this next

# The trace of DPLL

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

$F = X_1X_2 \vee X_1X_3 \vee X_2X_3$




The trace is a tree FBDD

# Background: DPLL with Caching

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

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

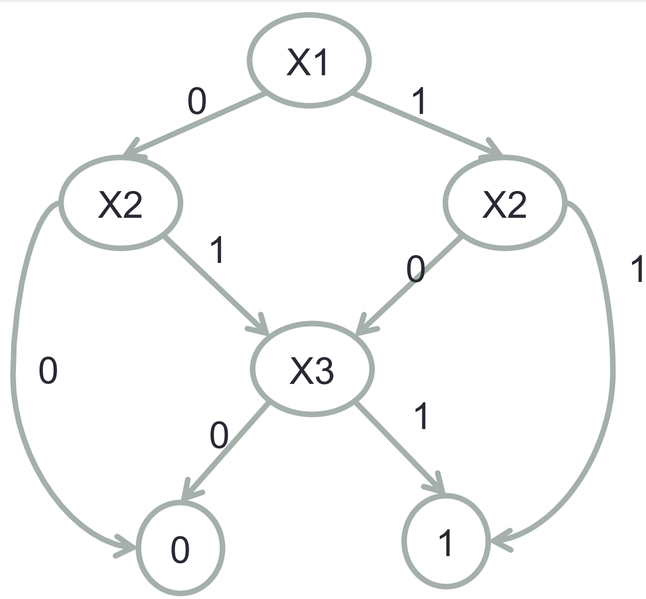
All modern model counting systems extend DPLL with Caching

# The trace of DPLL with Caching

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

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

$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$

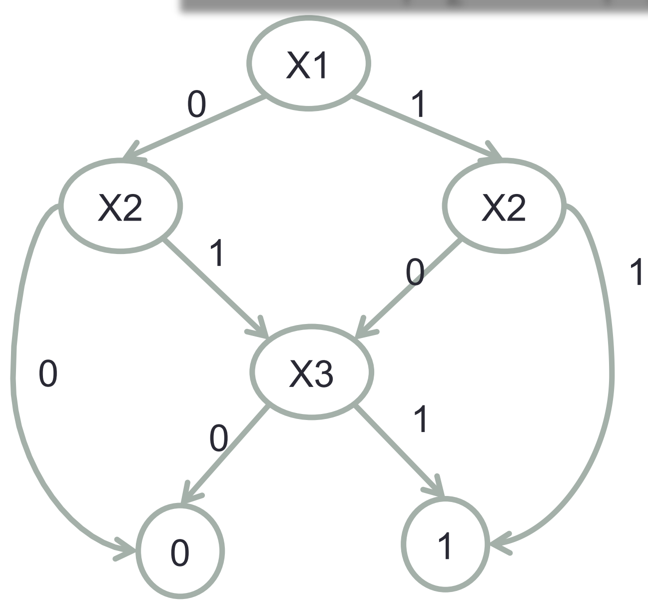



The trace is an FBDD

# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$

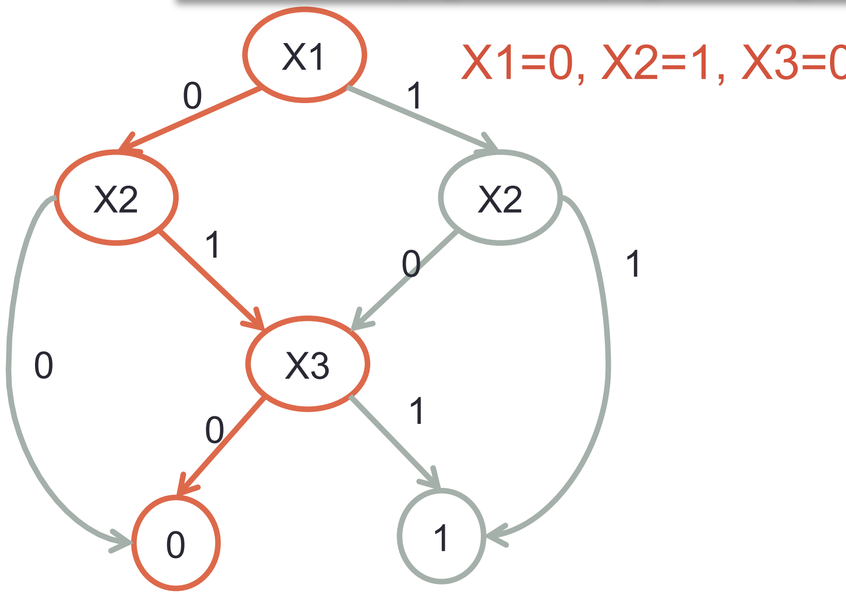


# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$

- Computing F with the FBDD:
  - If $X_i = 0$ follow the 0 edge
  - If $X_i = 1$ follow the 1 edge

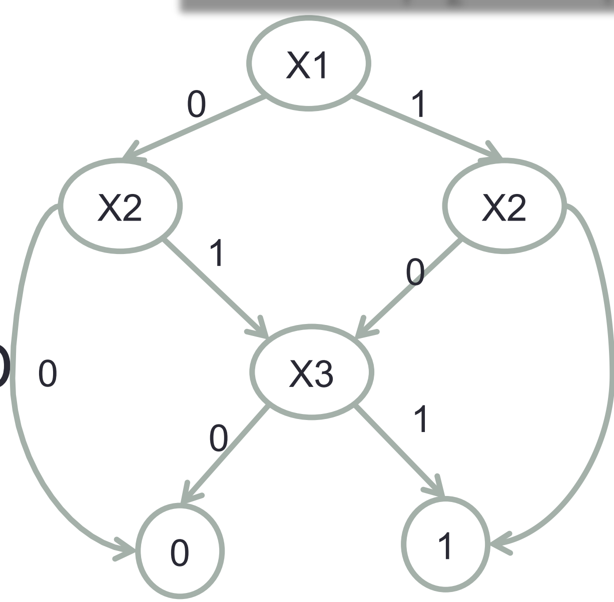


# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$F = X_1X_2 \vee X_1X_3 \vee X_2X_3$

- Computing F with the FBDD:
  - If $X_i = 0$ follow the 0 edge
  - If $X_i = 1$ follow the 1 edge
- Computing P(F) with the FBDD
  - Dynamic programming bottom-up




P(X1)=0.1
P(X2)=0.2
P(X3)=0.3

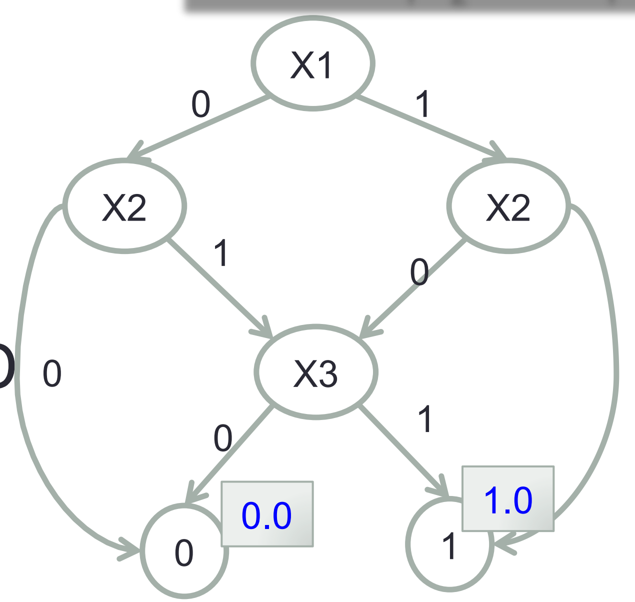
# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$

- Computing F with the FBDD:
  - If $X_i = 0$ follow the 0 edge
  - If $X_i = 1$ follow the 1 edge

- Computing P(F) with the FBDD
  - Dynamic programming bottom-up



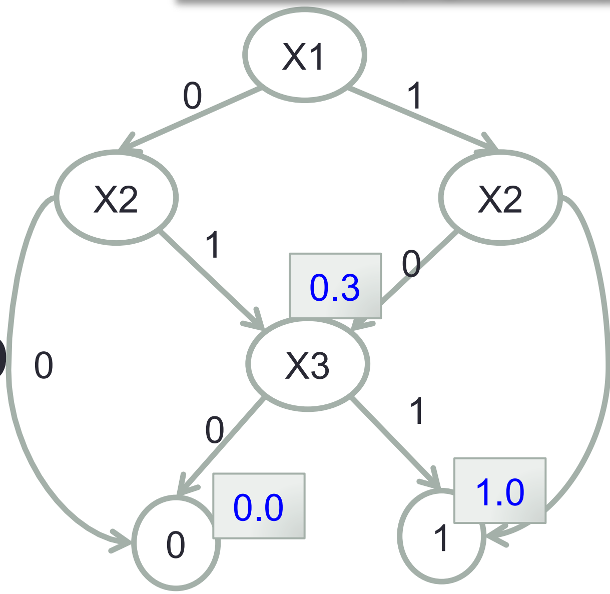

P(X1)=0.1
P(X2)=0.2
P(X3)=0.3

# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$$

- Computing F with the FBDD:
  - If $X_i = 0$ follow the 0 edge
  - If $X_i = 1$ follow the 1 edge
- Computing P(F) with the FBDD
  - Dynamic programming bottom-up



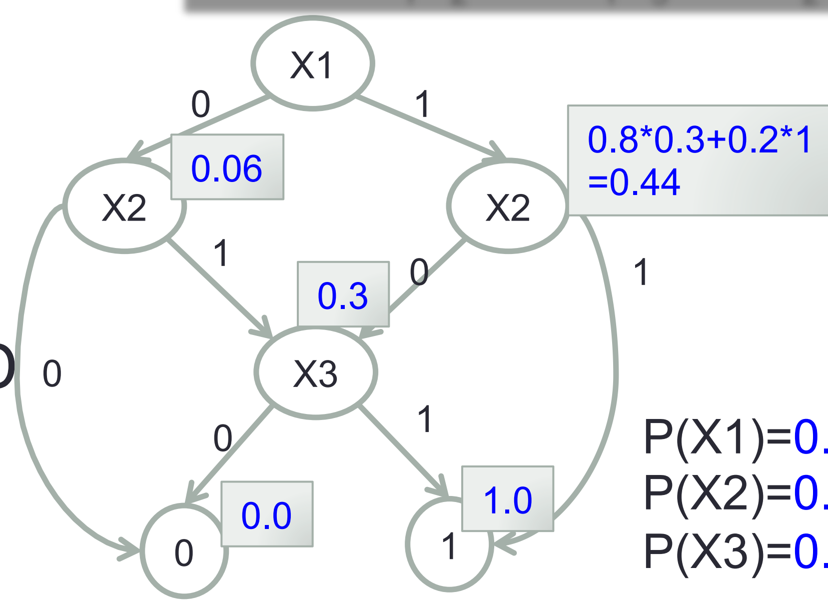

$P(X1)=0.1$
$P(X2)=0.2$
$P(X3)=0.3$

# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$$F = X_1X_2 \vee X_1X_3 \vee X_2X_3$$

- Computing F with the FBDD:
  - If $X_i = 0$ follow the 0 edge
  - If $X_i = 1$ follow the 1 edge
- Computing P(F) with the FBDD
  - Dynamic programming bottom-up



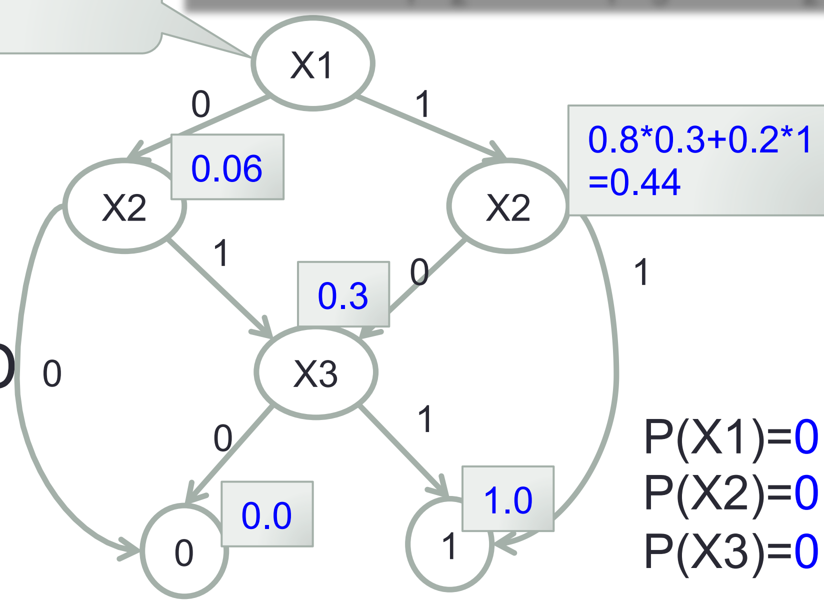

P(X1)=0.1
P(X2)=0.2
P(X3)=0.3

# Background: FBDD

- An FBDD (Free Binary Decision Diagram) is a graph s.t.:
  - Sink node are labeled 0 or 1
  - Internal nodes are labeled with a variable $X_i$, and have two outgoing edges, labeled 0 and 1
  - Every root-to-sink path visits each variable $X_i$ at most once

$F = X_1X_2 \lor X_1X_3 \lor X_2X_3$

- Computing F with the FBDD:
  - If $X_i$ = 0 follow the 0 edge
  - If $X_i$ = 1 follow the 1 edge
- Computing P(F) with the FBDD
  - Dynamic programming bottom-up




P(X1)=0.1
P(X2)=0.2
P(X3)=0.3

# Background:DPLL w Fixed Variable Order

- The order in which the DPLL procedure processes the Boolean variables X has dramatic impact on performance
- Heuristics: choose a fixed variable order Π: $X_1$, $X_2$, …; process variables always in this order
- An OBDD (Ordered Binary Decision Diagram) is an FBDD where every path from the root to a leaf node visits the variables in the same order Π

# Background:DPLL w Fixed Variable Order

Process variables in predefined order $X_1, X_2, \ldots$

// basic DPLL:

Function P(F):

  if F = false then return 0

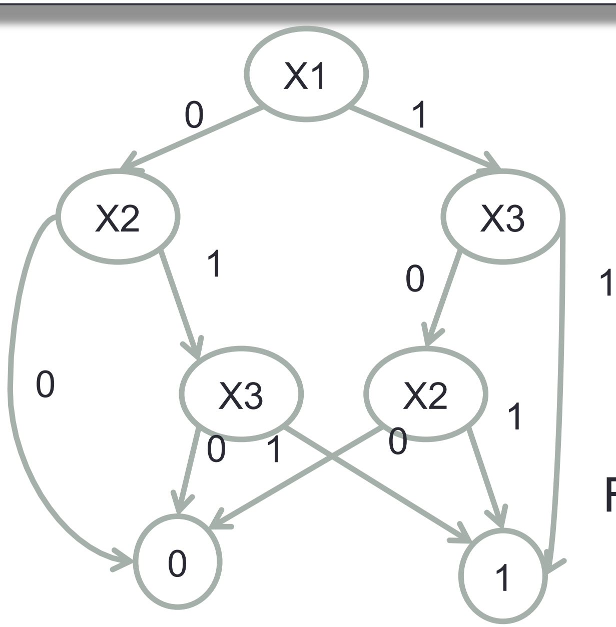
  if F = true then return 1

  select a variable X, return $(1\text{-}P(X)) \times P(F_{X=0}) + P(X) \times P(F_{X=1})$

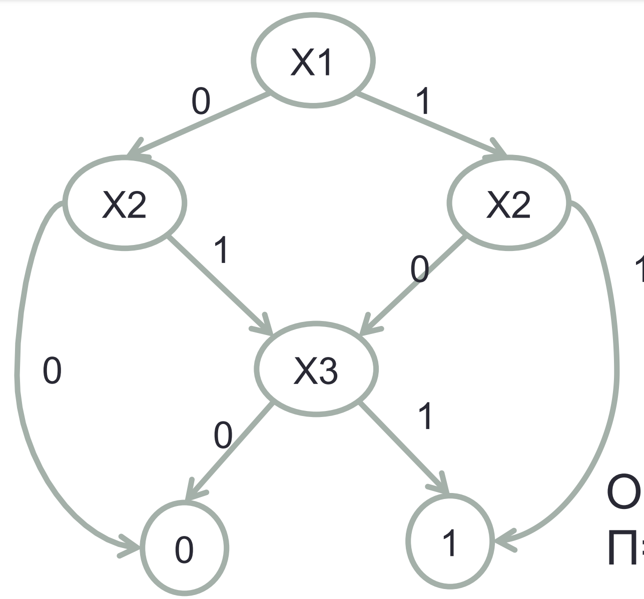
// DPLL with caching:

Cache F and P(F);

look it up before computing




FBDD




OBDD

Π=X1,X2,X3

# Background: DPLL w/ Components

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

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

// DPLL with components:
  if $F = F_1 \wedge F_2$
    and $F_1$, $F_2$ have no common variables
  then return $P(F_1) \times P(F_2)$

Most modern model counting systems implement components

The trace of a DPLL w/ caching and components is called a "decision-DNNF".

**Theorem**: [Beame'13] Every decision-DNNF for a positive k-DNF formula can be converted into an FBDD of size $O(N^k)$

# Summary of DPLL

| **DPLL Variant** | **Trace** |
|---|---|
| DPLL with caching and fixed variable order | OBDD |
| DPLL with caching | FBDD |
| DPLL with caching and components | FBDD<br>(for queries) |
| … a different model counting formalism | d-DNNF |

# Intentional Query Evaluation

Query $Q$ + database $D$ → lineage expression $F_Q$

Compute $P(F)$ using a general model counting system

Question: for which $Q$ does the system run in PTIME?

- The size of the trace gives a lower bound on the running time of DPLL-based algorithm

# Example

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q = R(x), S(x,y)

R

| x | |
|---|---|
| a1 | X1 |
| a2 | X2 |
| a3 | X3 |

S

| x | y | |
|---|---|---|
| a1 | b1 | Y1 |
| a1 | b2 | Y2 |
| a2 | b3 | Y3 |
| a2 | b4 | Y4 |
| a2 | b5 | Y5 |

$F_Q$ = X1 Y1 ∨ X1 Y2 ∨ X2 Y3 ∨ X2 Y4 ∨ X2 Y5

Study the size of OBDD, FBDD, etc for the formula $F_Q$

# 1. Read-Once Boolean Formulas

A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

$$(X \vee Z) \wedge (Y \vee U)$$

# 1. Read-Once Boolean Formulas

A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

$(X \vee Z) \wedge (Y \vee U)$

- P(F) can be computed in linear time:

$$P(F_1 \wedge F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \vee F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

# 1. Read-Once Boolean Formulas

A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

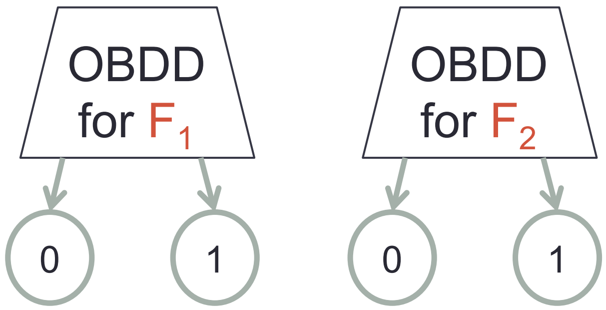
$(X \vee Z) \wedge (Y \vee U)$

- P(F) can be computed in linear time:

$$P(F_1 \wedge F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \vee F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

- F has an OBDD with n nodes, where n = # of variables

$F_1 \wedge F_2$ $F_1 \vee F_2$



# 1. Read-Once Boolean Formulas

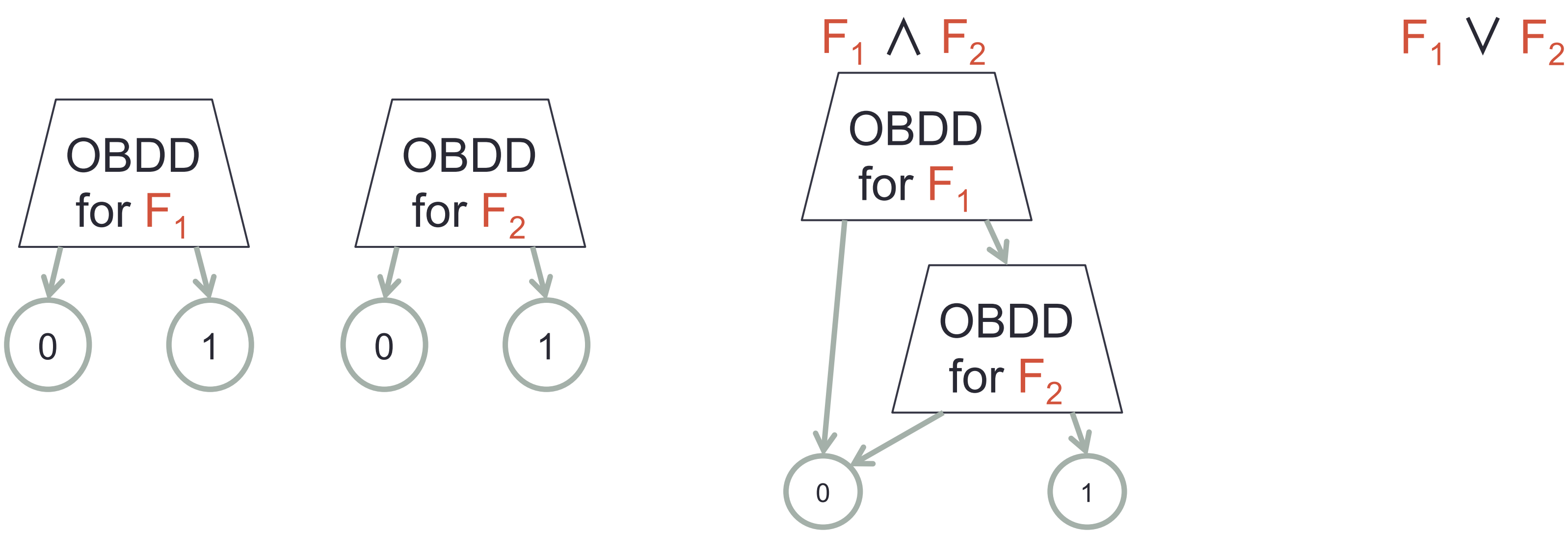
A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

- P(F) can be computed in linear time:

$(X \lor Z) \land (Y \lor U)$

$$P(F_1 \land F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \lor F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

- F has an OBDD with n nodes, where n = # of variables



# 1. Read-Once Boolean Formulas

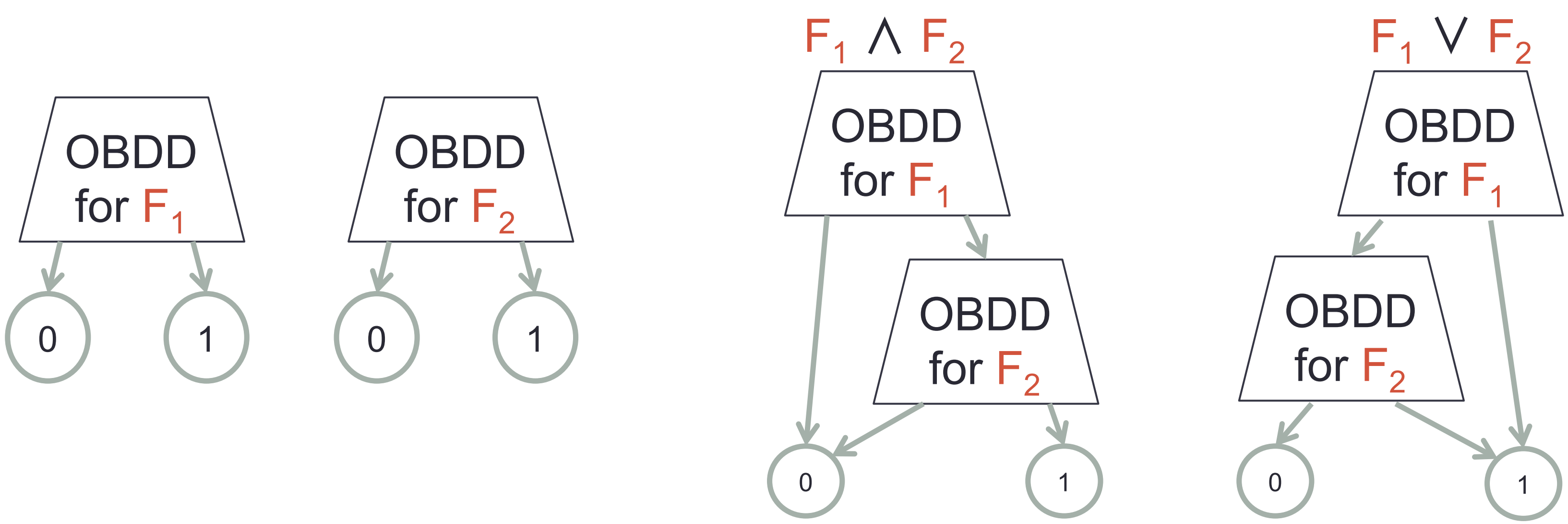
A Boolean formula F is called read-once if it can be written such that every Boolean variable occurs only once

$(X \vee Z) \wedge (Y \vee U)$

- P(F) can be computed in linear time:

$$P(F_1 \wedge F_2) = P(F_1) \times P(F_2)$$
$$P(F_1 \vee F_2) = 1 - (1 - P(F_1)) \times (1 - P(F_2))$$

- F has an OBDD with n nodes, where n = # of variables



# 1. Read-Once Example

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q = R(x), S(x,y)

R

| x | |
|---|---|
| a1 | X1 |
| a2 | X2 |
| a3 | X3 |

S

| x | y | |
|---|---|---|
| a1 | b1 | Y1 |
| a1 | b2 | Y2 |
| a2 | b3 | Y3 |
| a2 | b4 | Y4 |
| a2 | b5 | Y5 |

$F_Q$ = X1 Y1 V X1 Y2 V X2 Y3 V X2 Y4 V X2 Y5

= X1 (Y1 V Y2) V X2 (Y3 V Y4 V Y5)

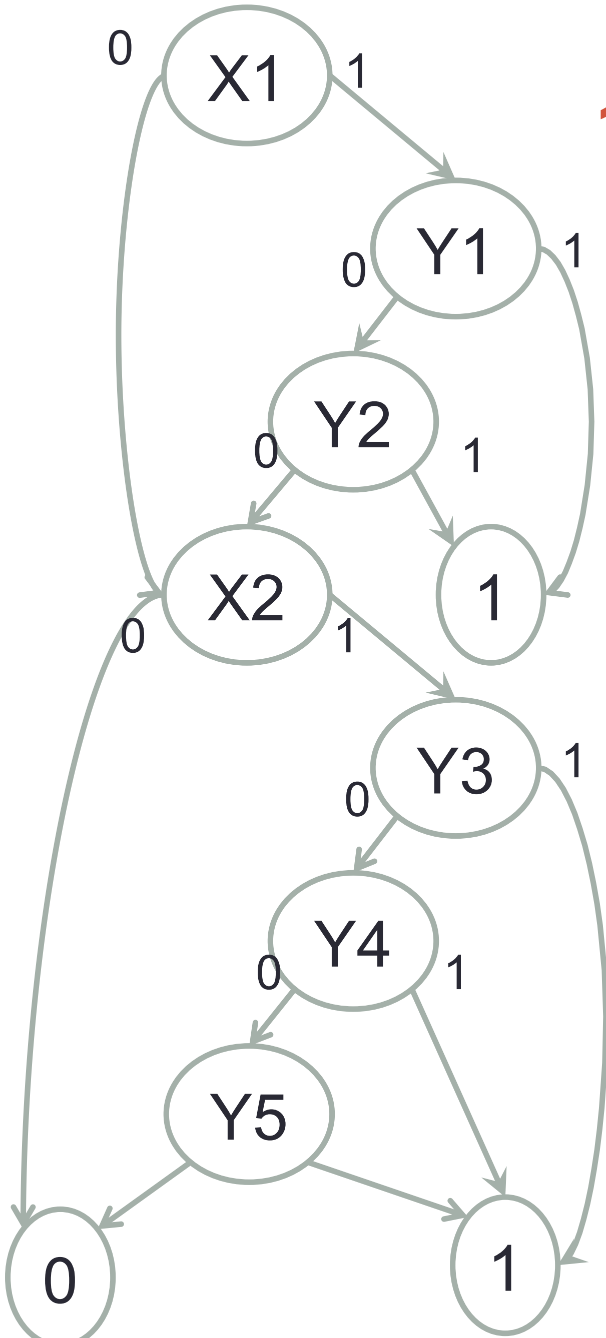
Read-once

# 1. Read-Once Example

$Q = R(x), S(x,y)$

$F_Q = X1\ (Y1 \lor Y2) \lor X2\ (Y3 \lor Y4 \lor Y5)$

Read-once to OBDD




Note: every Boolean variable occurs only once in the OBDD

The size of the OBDD is linear in the size of the database D

# 1. Read-Once Queries

**Theorem** For any query Q, the following are equivalent:

- ∀ D, the lineage $F_Q$ is read-once
- Q can be written such that it is both hierarchical, and every symbols occurs once.

Read-once:

Q = R(x), S(x,y)

… all hierarchical, conjunctive queries w/o self-joins

$Q_U = R(x_1), S(x_1,y_1) \vee T(x_2), S(x_2,y_2)$

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

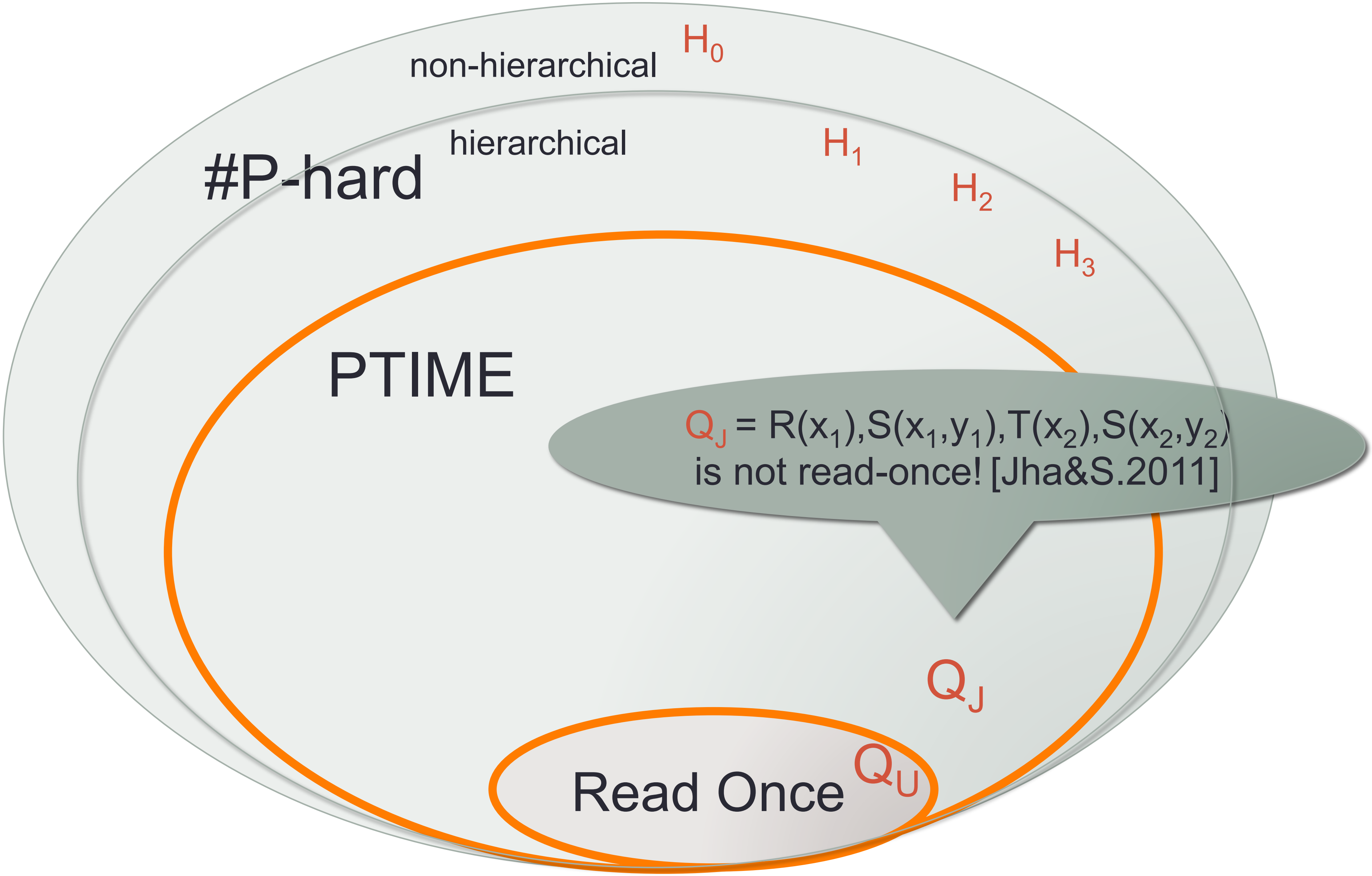
…

Not read-once (but still PTIME):

$Q_J = R(x_1), S(x_1,y_1), T(x_2), S(x_2,y_2)$

…

# Landscape of Probabilistic Databases



# 2. OBDD

OBDD Synthesis [Wegener'2000]

Let Π be an order of the Boolean variables: $X_1, X_2, \ldots, X_k, \ldots, X_n$.

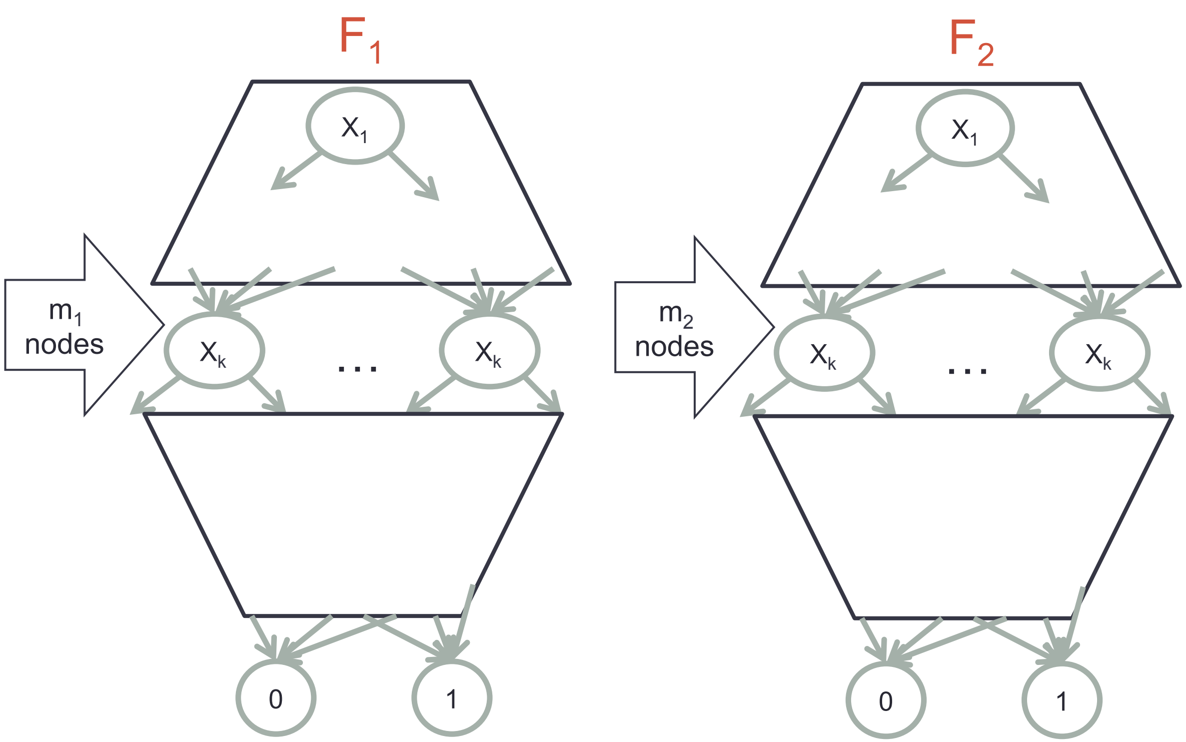
- If the Π-OBDD for $F_1$ has size $N_1$, and the Π-OBDD for $F_2$ has size $N_2$ then Π-OBDD for $F_1 \wedge F_2$ and for $F_1 \vee F_2$ has has size $\leq N_1N_2$

# 2. OBDD

OBDD Synthesis [Wegener'2000]

Let Π be an order of the Boolean variables: $X_1, X_2, \ldots, X_k, \ldots, X_n$.

- If the Π-OBDD for $F_1$ has size $N_1$, and the Π-OBDD for $F_2$ has size $N_2$ then Π-OBDD for $F_1 \wedge F_2$ and for $F_1 \vee F_2$ has has size $\leq N_1 N_2$

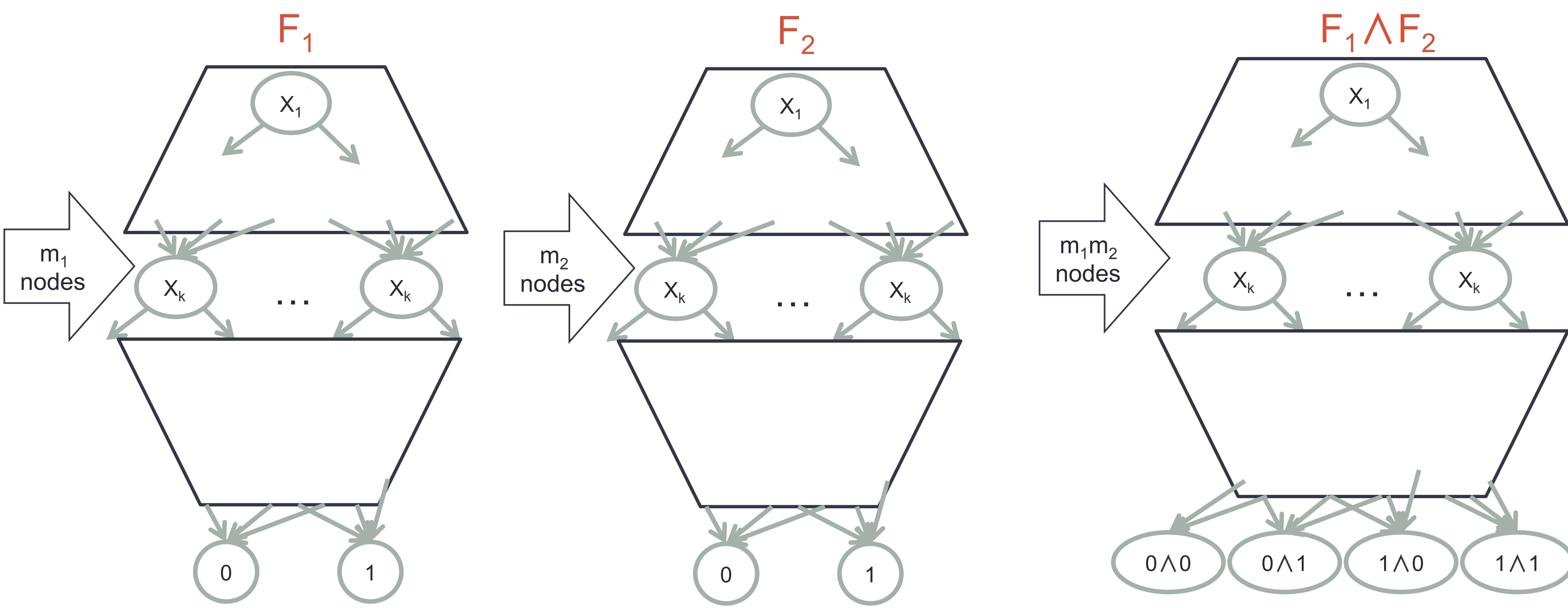


# 2. OBDD

OBDD Synthesis [Wegener'2000]

Let Π be an order of the Boolean variables: $X_1, X_2, \ldots, X_k, \ldots, X_n$.

- If the Π-OBDD for $F_1$ has size $N_1$, and the Π-OBDD for $F_2$ has size $N_2$ then Π-OBDD for $F_1 \wedge F_2$ and for $F_1 \vee F_2$ has has size $\leq N_1N_2$



$Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)$

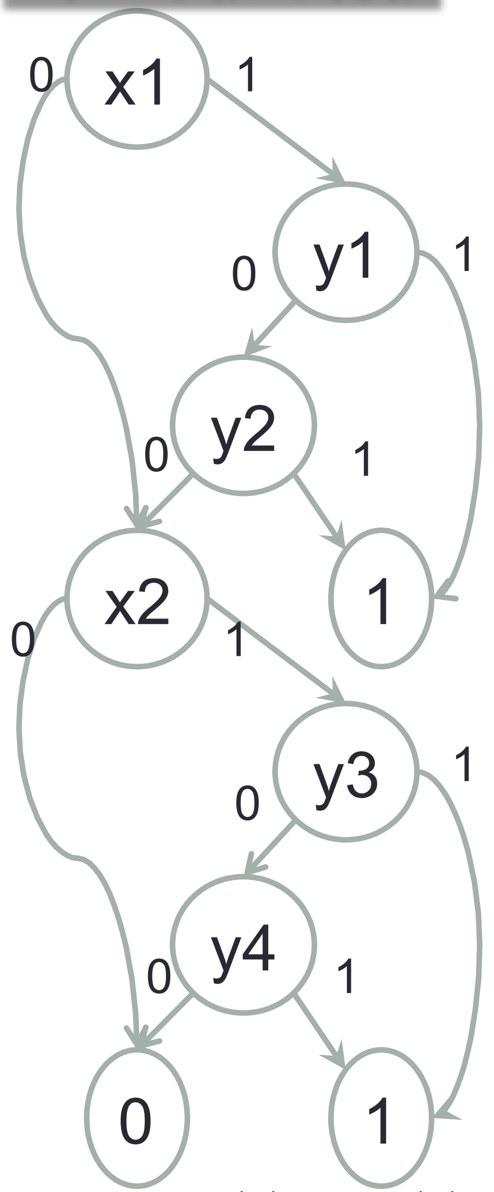
# 2. OBDD Example

$Q_1 = R(x_1),S(x_1,y_1)$ $\wedge$ $Q_2 = T(x_2),S(x_2,y_2)$ $=$ $Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)$

# 2. OBDD Example

$Q_1 = R(x_1),S(x_1,y_1)$ $\wedge$ $Q_2 = T(x_2),S(x_2,y_2)$ $=$ $Q_J = R(x_1),S(x_1,y_1),\ T(x_2),S(x_2,y_2)$

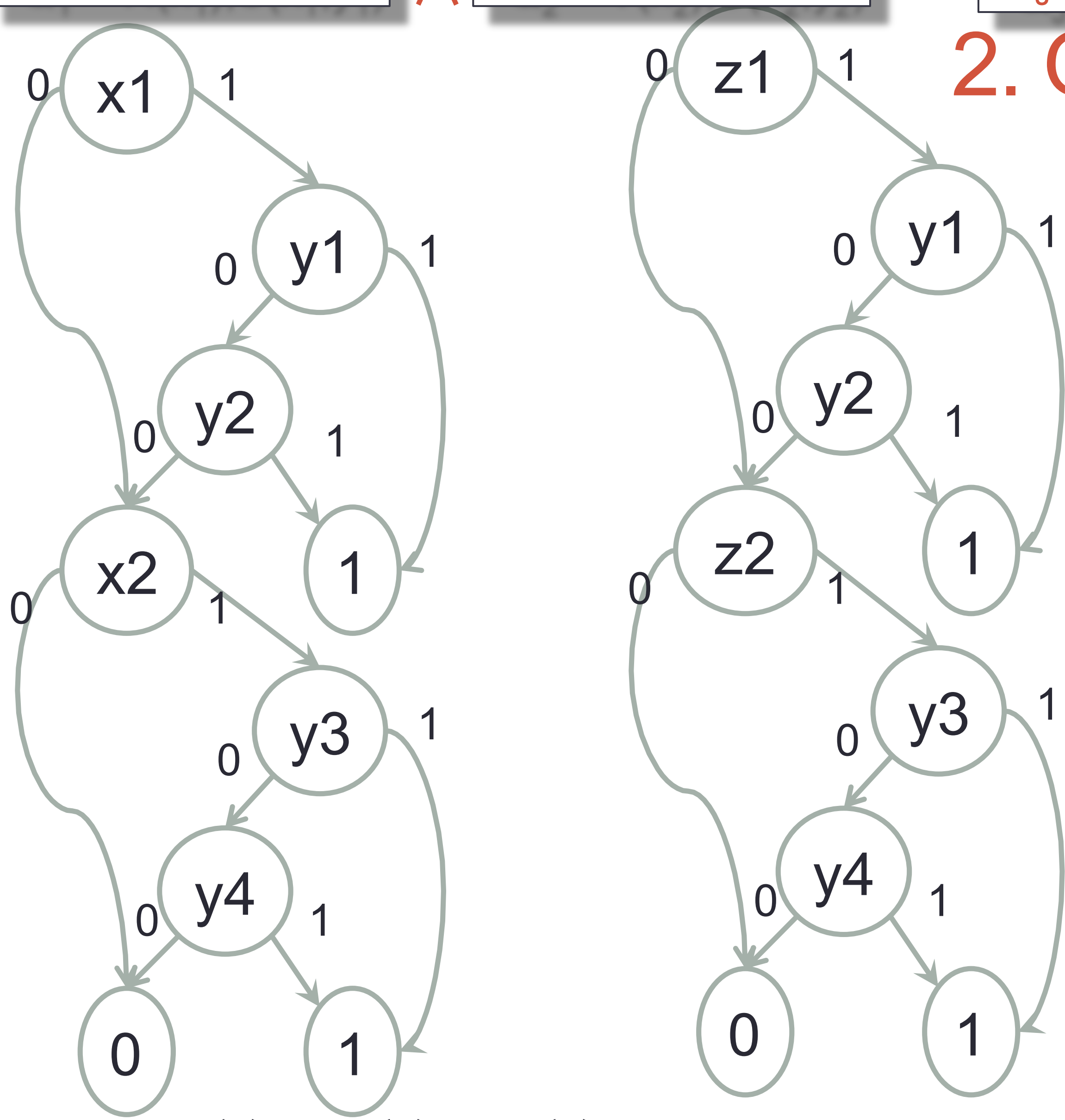
# 2. OBDD Example




$F_1 = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$

$Q_1 = R(x_1),S(x_1,y_1)$ $\wedge$ $Q_2 = T(x_2),S(x_2,y_2)$ = $Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)$

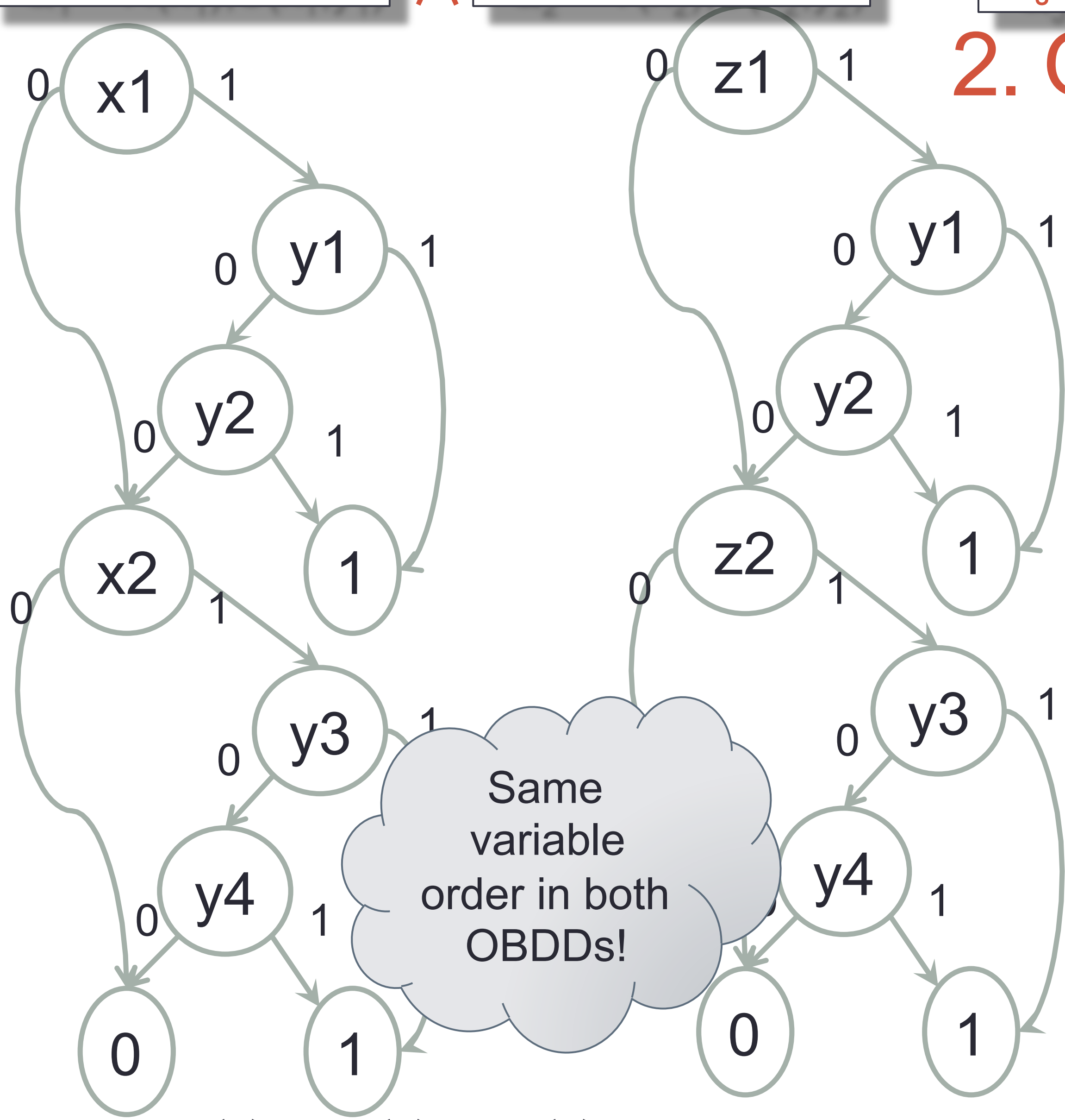
# 2. OBDD Example




$F_1 = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$ $F_2 = Z_1Y_1 \vee Z_1Y_2 \vee Z_2Y_3 \vee Z_2Y_4$

$Q_1 = R(x_1),S(x_1,y_1)$ $\wedge$ $Q_2 = T(x_2),S(x_2,y_2)$ $=$ $Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)$

# 2. OBDD Example

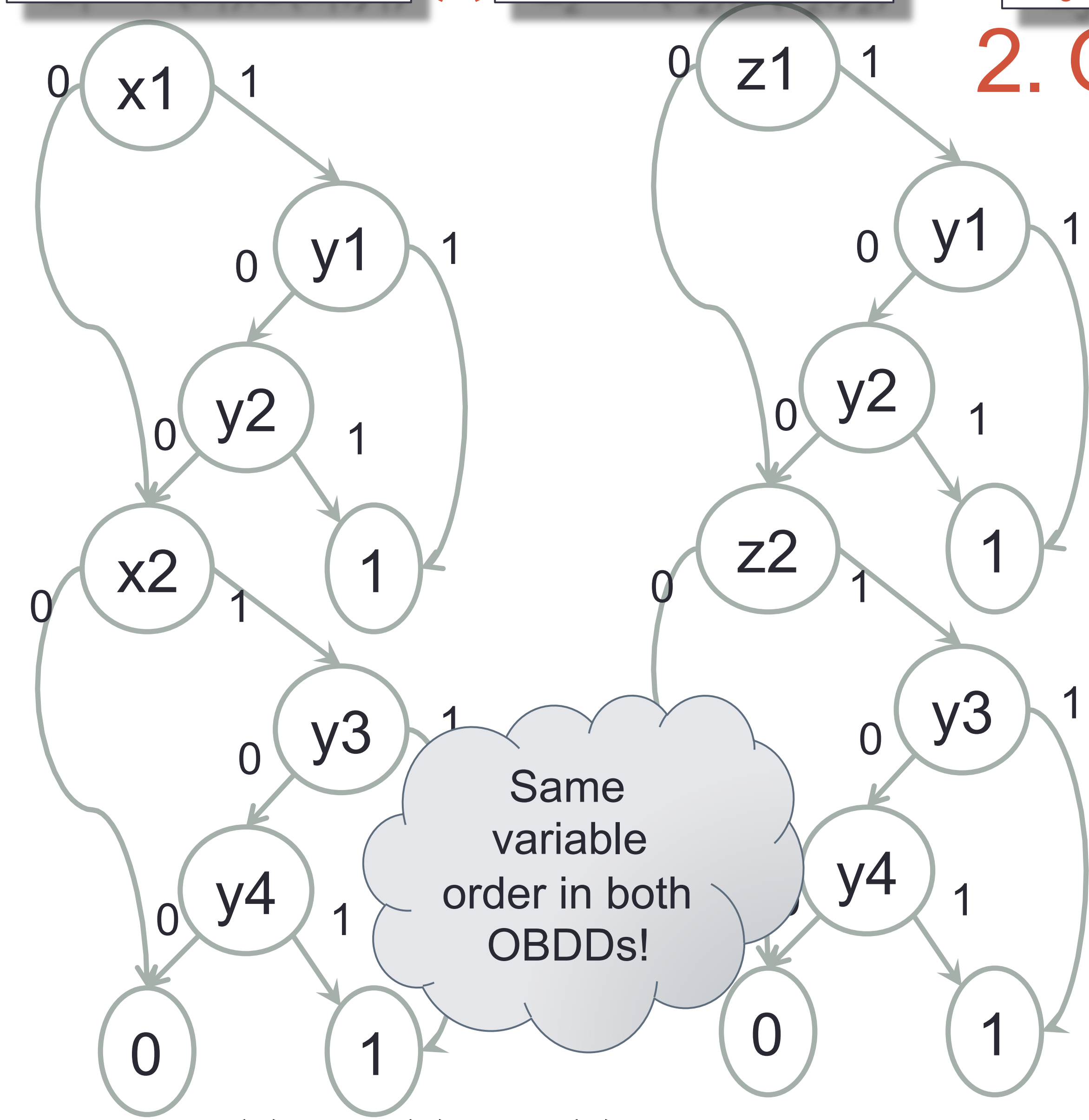



$F_1 = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$

$F_2 = Z_1Y_1 \vee Z_1Y_2 \vee Z_2Y_3 \vee Z_2Y_4$

$Q_1 = R(x_1),S(x_1,y_1)$ $\wedge$ $Q_2 = T(x_2),S(x_2,y_2)$ $=$ $Q_J = R(x_1),S(x_1,y_1), T(x_2),S(x_2,y_2)$

# 2. OBDD Example




= Efficient OBDD for $Q_J = Q_1 \wedge Q_2$

$F_1 = X_1Y_1 \vee X_1Y_2 \vee X_2Y_3 \vee X_2Y_4$

$F_2 = Z_1Y_1 \vee Z_1Y_2 \vee Z_2Y_3 \vee Z_2Y_4$

# 2. Queries with Efficient OBDD

**Theorem** For any query Q the following are equivalent:
- $\forall$ D, the lineage $F_Q$ has an OBDD of size $|D|^{O(1)}$
- Q is inversion-free (hierarchical + same hierarchy order for each symbol; see book)

Inversion-free

$Q_J = R(x_1), S(x_1,y_1), T(x_2), S(x_2,y_2)$

Same order in both S:
$at(x_1) \subset at(y_1)$
$at(x_2) \subset at(y_2)$

…

Has inversion (but still PTIME):

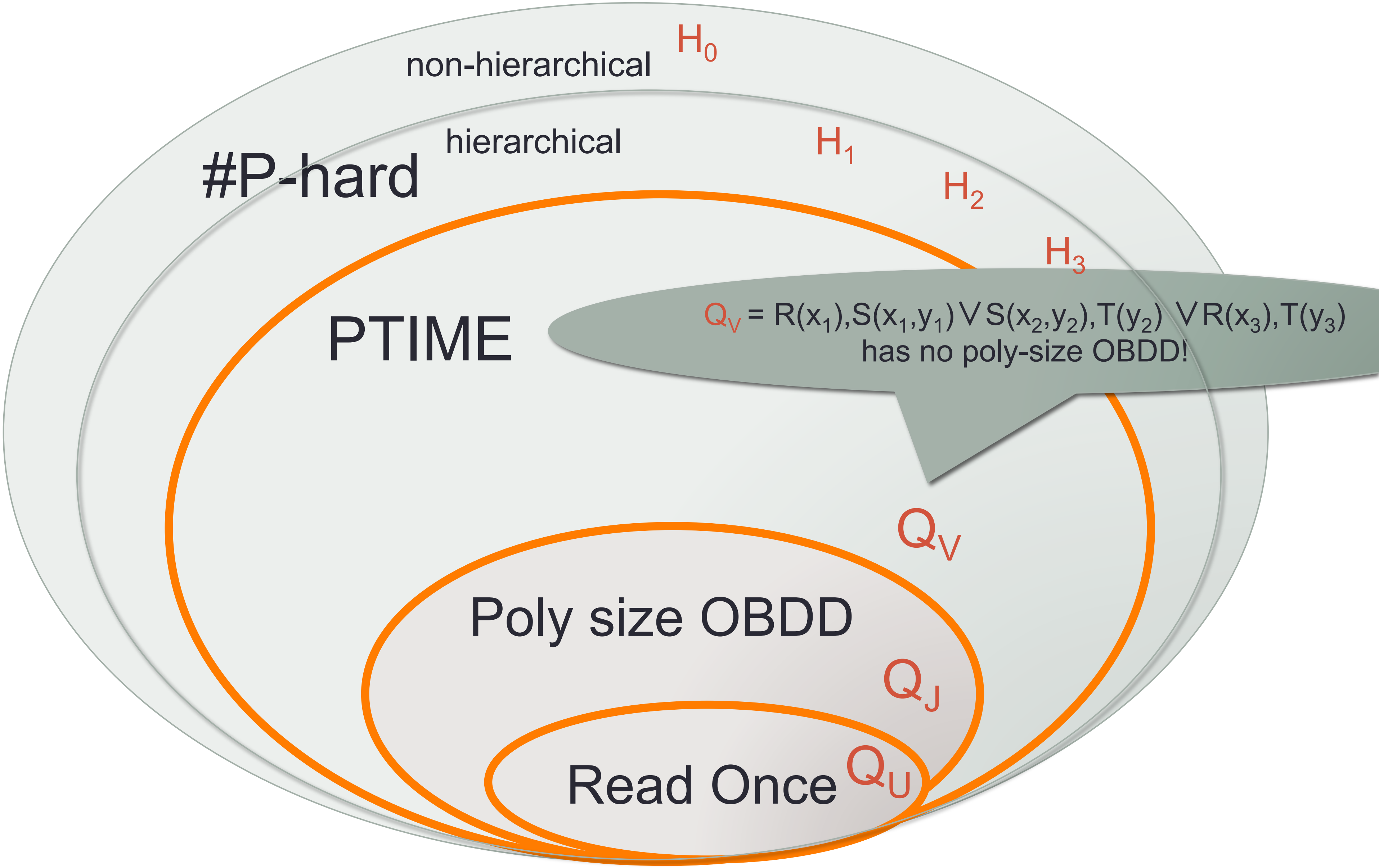
$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$

In S: $at(x_1) \supset at(y_1)$

In S: $at(x_2) \subset at(y_2)$

…

# Landscape of Probabilistic Databases



# 3. FBDD

- Some queries have inversion (hence no efficient OBDD), but have an efficient FBDD
- We only know of examples of such queries, no complete characterization

# 3. FBDD

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

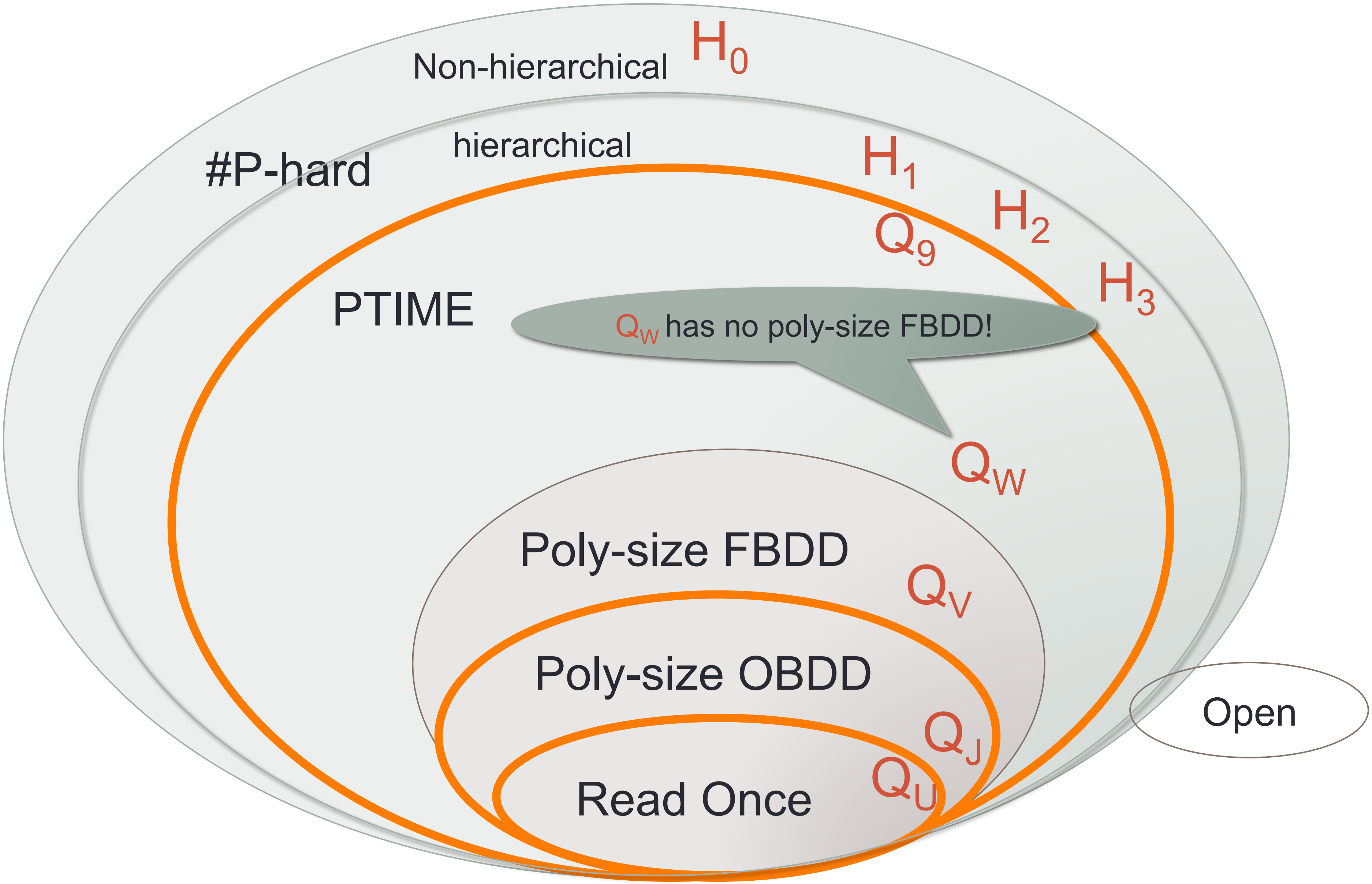
**Fact**: (The lineage of ) $Q_V$ has an FBDD whose size is linear in |D|

$$\begin{array}{llll} Q_W = & [R(x_0),S_1(x_0,y_0) \vee S_2(x_2,y_2),S_3(x_2,y_2)] \wedge & & /^*\ Q1\ ^*/ \\ & [R(x_0),S_1(x_0,y_0) \vee S_3(x_3,y_3),T(y_3)] \wedge & & /^*\ Q2\ ^*/ \\ & [S_1(x_1,y_1),S_2(x_1,y_1) \vee S_3(x_3,y_3),T(y_3)] & & /^*\ Q3\ ^*/ \end{array}$$

**Theorem**: Any FBDD for (the lineage of ) $Q_W$ has size exponential in |D|

See the book for both results.

# Landscape of Probabilistic Databases



# 4. d-DNNF

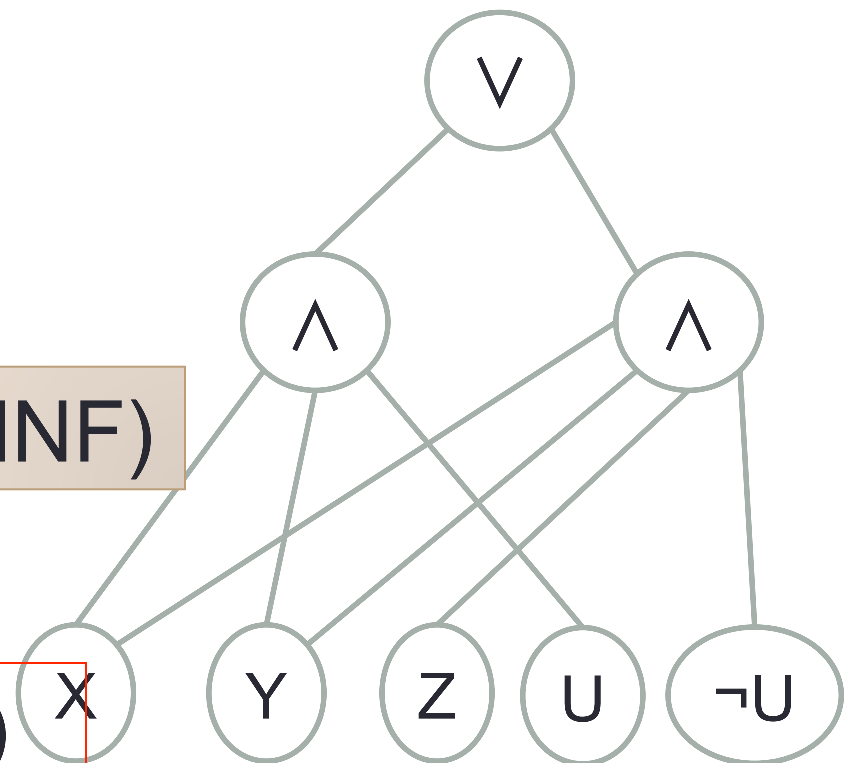
[Darwiche'2000]

**d-DNNF** = expression DAG where
$\wedge$ are "Decomposable"
$\vee$ are "Deterministic"
All negations at the leaves

P(F) in PTIME in size(d-DNNF)

$$P(F_1 \wedge F_2) = P(F_1) * P(F_2)$$
$$P(F_1 \vee F_2) = P(F_1) + P(F_2)$$

$X\,Y\,U \vee X\,Y\,Z$



# 4. d-DNNF
[Darwiche'2000]

**Fact 1.** $Q_W$ has an efficient d-DNNF

Proof. Condition on $h_{30}$:

$Q_W = (h_{30} \lor h_{32}) \land (h_{30} \lor h_{33}) \land (h_{31} \lor h_{33})$

$$= [(\neg h_{30}) \land h_{32} \land h_{33} \land (h_{31} \lor h_{33})] \lor [h_{30} \land (h_{31} \lor h_{33})]$$

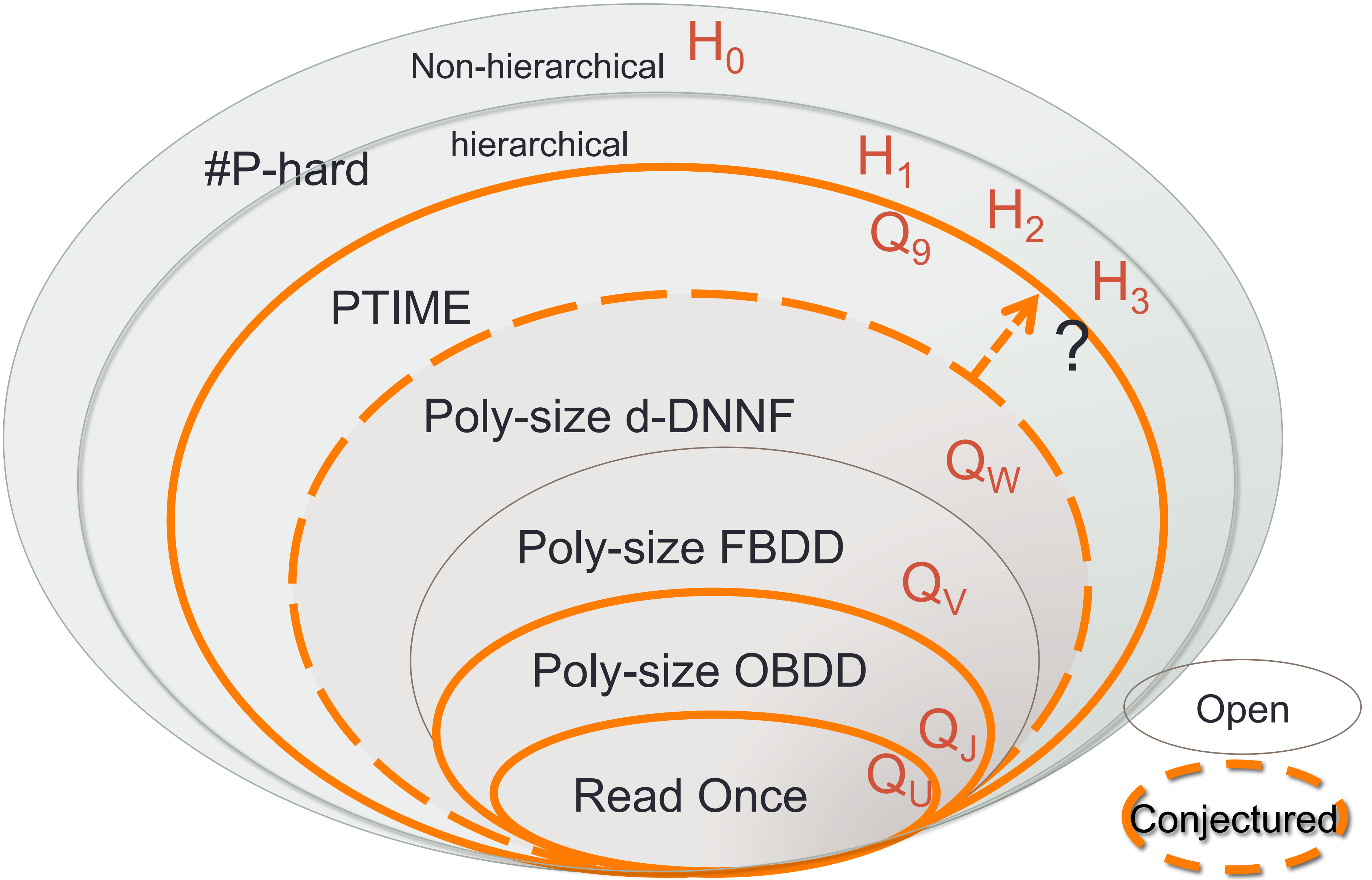
$$= \underbrace{[(\neg h_{30}) \land h_{32} \land h_{33}]}_{\text{Inversion-free}} \underbrace{\lor}_{\text{Deterministic}} \underbrace{[h_{30} \land (h_{31} \lor h_{33})]}_{\text{Inversion-free}}$$

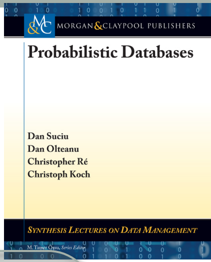
**Fact 2.** $\neg Q_W$ has an efficient d-DNNF

See book

**Conjecture.** $Q_9$ has as no efficient d-DNNF

# Landscape of Probabilistic Databases






# Outline

**Part 1**

1. Motivating Applications
2. The Probabilistic Data Model — Chapter 2

**Part 2**

3. Extensional Query Plans — Chapter 4.2
4. The Complexity of Query Evaluation — Chapter 3

**Part 3**

5. Extensional Evaluation — Chapter 4.1

**Part 4**

6. Intensional Evaluation — Chapter 5
7. Conclusions

# Summary (1/2)

- There are many applications that require storage/ management of *uncertain* data
  - Retain data that is not absolutely certain
  - Retain more than one alternative way to clean
- Probabilities are application specific
  - All we care about is that “bigger is better”
- Queries have precise semantics
  - Important for query optimization
  - “Bigger probability” means “more certain answer”
- “Stop worrying about probabilities and start asking queries”

# Summary (2/2)

- Extensional query evaluation:
  - Advantage: can use out of the box DBMS
  - Disadvantage: can't handle unsafe queries (but can still give upper/ lower bounds on probabilities)
  - *"You don't need a probabilistic database management system to manage probabilistic data"*

- Intensional query evaluation:
  - Advantage: can use out of the box model counting system
  - Disadvantage: requires expensive "lineage computation" step; none of the model counting approaches seems complete for SPJU queries.

# Open Problems

- How do we compute the "hard" queries ?
- Extensions to:
  - Full FO: $\neg$ $\forall$
  - Independent AND disjoint tuples
  - Uniform probabilistic structures (MLNs)
- Which queries admit efficient FBDDs? Efficient d-DNNFs?

# Thank You !

http://www.cs.washington.edu/homes/suciu/