# AN OVERVIEW OF PROBABILISTIC DATABASES

Dan Suciu
University of Washington

# Probabilistic Databases

- Traditional Data: deterministic
  - Accounting, inventory, …
- New Data: uncertain/probabilistic
  - Big data analytics
  - Information extraction, fuzzy object matching
  - Sensor data, moving objects
  - Scientific data

# The Landscape of ProbDBs

## Early days

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

Core challenge:
Query Evaluation
(=Probabilistic Inference)

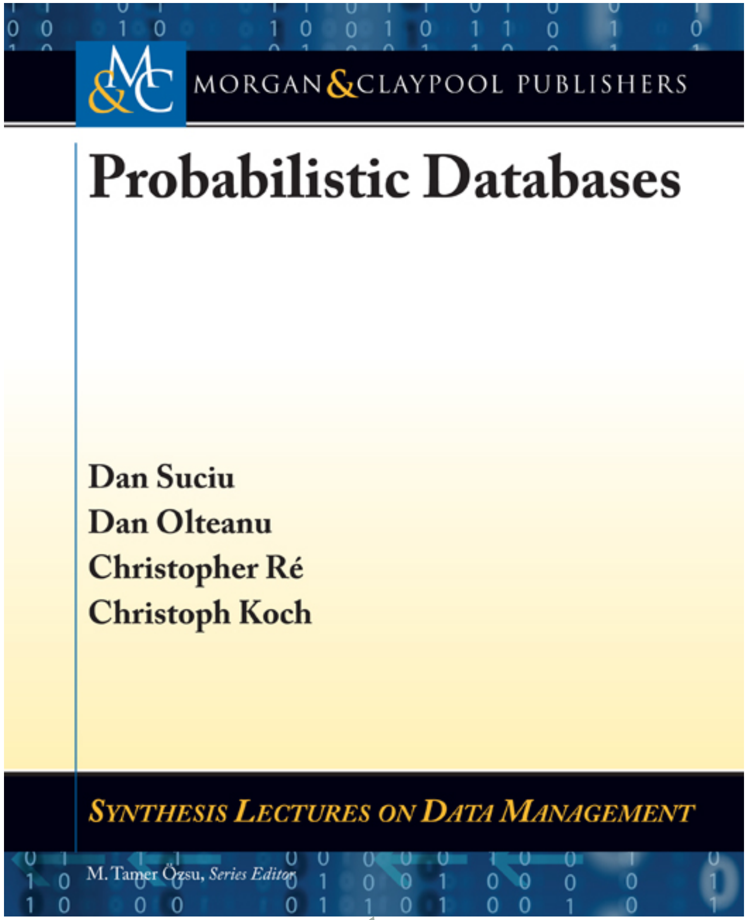
## Recent work

- 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

# This Talk: Query Evaluation

This talk is based on:

- Dalvi, S. *Efficient query evaluation on probabilistic databases.* VLDB'04
- Dalvi,S. *The Dichotomy of Probabilistic Inference for UCQ*, JACM'12
- Dalvi, Schnaitter, S.: *Computing query probability with incidence algebras.* PODS'10
- Jha, S.: Probabilistic Databases with MarkoViews, VLDB'2012
- Jha, S.: Query Compilation, ICDT'2013




…and "the book"

# Outline

- Introduction
- Motivation and Background
- Extensional Query Evaluation
- Intensional Query Evaluation
- Conclusions

# Probabilistic Database = Data+Probability

Friends

| Name1 | Name2 | |
|---|---|---|
| Alice | Bob | .7 |
| Alice | Carol | .4 |
| Carol | Fred | .2 |
| Bob | Carol | .2 |
| Alice | Fred | .5 |
| … | | |

```
-- find Alice and Carol's
-- common friends

SELECT DISTINCT y.name2
FROM Friends x, Friends y
WHERE x.name1 = 'Alice'
    and x.name2 = y.name2
    and y.name1 = 'Carol'
```

| | |
|---|---|
| Fred | .10 |
| Kurt | .03 |
| … | |

Tuples: have probabilities

Queries: SQL

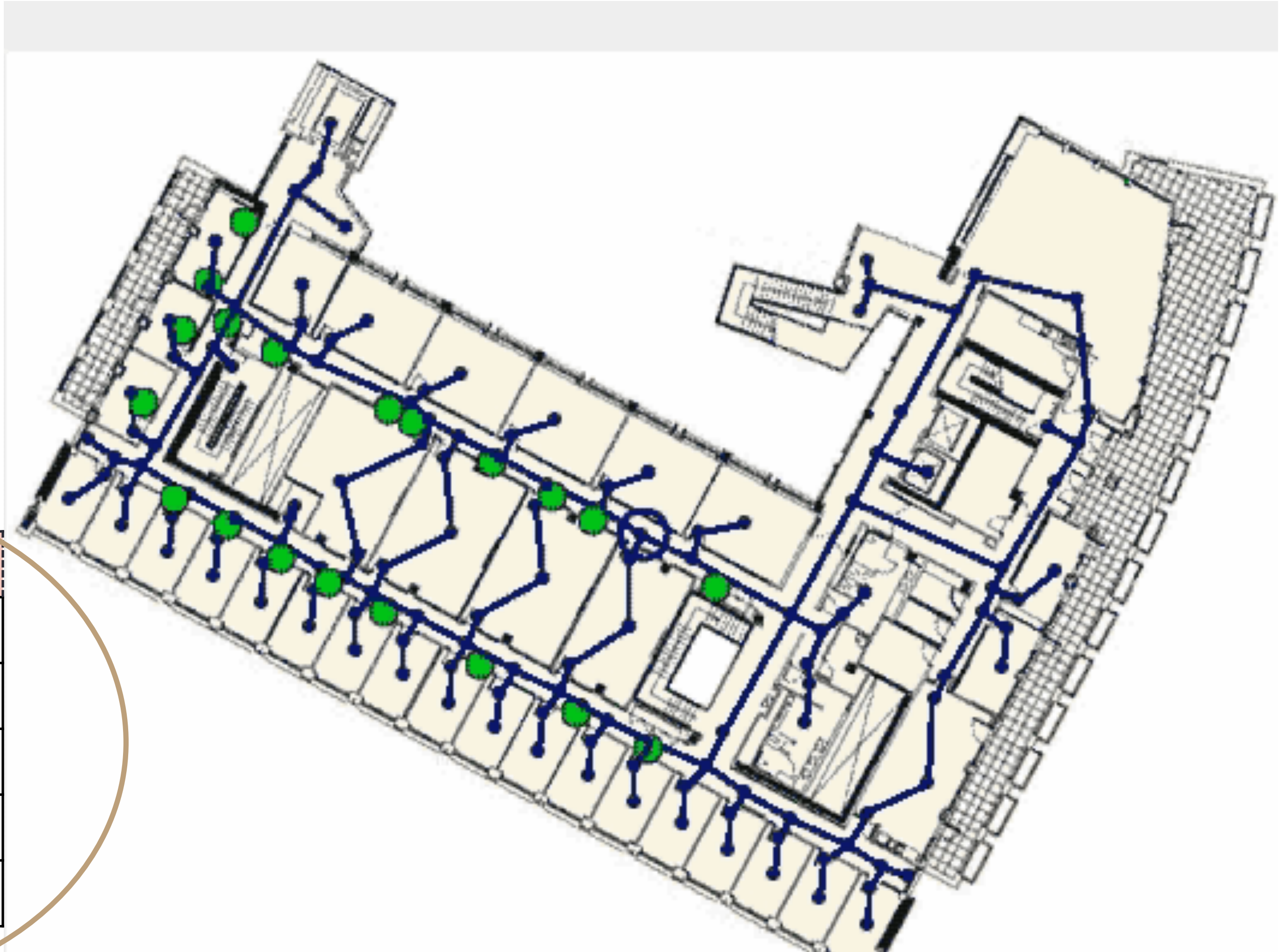
Answers: have probabilities

Balazinska, Borriello, Letchner, Re, Welbourne

# Example1: RFID Data

Standard Deterministic DB

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



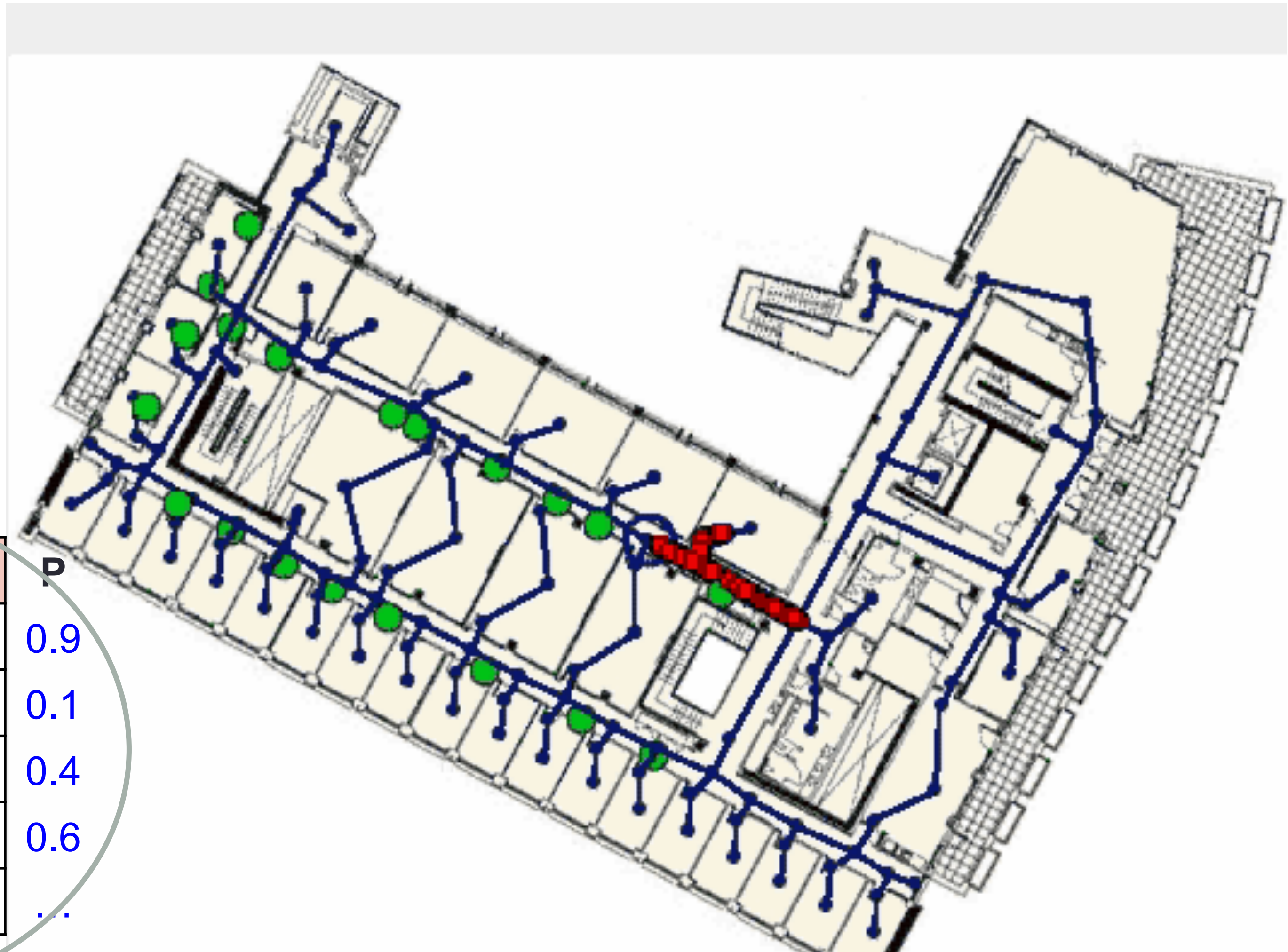
Complex Query: "Who brought a laptop to the meeting ?"

Balazinska, Borriello, Letchner, Re, Welbourne

# Example1: RFID Data

Probabilistic DB

| Tag | Loc | Time | P |
|---|---|---|---|
| Joe | Hall4 | 3 | 0.9 |
| | Office3 | 3 | 0.1 |
| Joe | Hall4 | 4 | 0.4 |
| | Office3 | 4 | 0.6 |
| Bob | …. | … | … |



Complex Query: "Who brought a laptop to the meeting ?"

[Gupta&Sarawagi'2006]

# Ex 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

[Stoyanovich'2011]

# Ex 3: Modeling Missing Data

| id | age | edu | inc | nw |
|---|---|---|---|---|
| $t_1$ | 20 | HS | ? | ? |
| $t_2$ | 20 | BS | 50K | 100K |
| $t_3$ | 20 | ? | 50K | ? |
| $t_4$ | 20 | HS | 100K | 500K |
| $t_5$ | 20 | ? | ? | ? |
| $t_6$ | 20 | HS | 50K | 100K |
| $t_7$ | 20 | HS | 50K | 500K |
| $t_8$ | ? | HS | ? | ? |
| $t_9$ | 30 | BS | 100K | 100K |
| $t_{10}$ | 30 | ? | 100K | ? |
| $t_{11}$ | 30 | HS | ? | ? |
| $t_{12}$ | 30 | MS | ? | ? |
| $t_{13}$ | 40 | BS | 100K | 100K |
| $t_{14}$ | 40 | HS | ? | ? |
| $t_{15}$ | 40 | BS | 50K | 500K |
| $t_{16}$ | 40 | HS | ? | 500K |
| $t_{17}$ | 40 | HS | 100K | 500K |

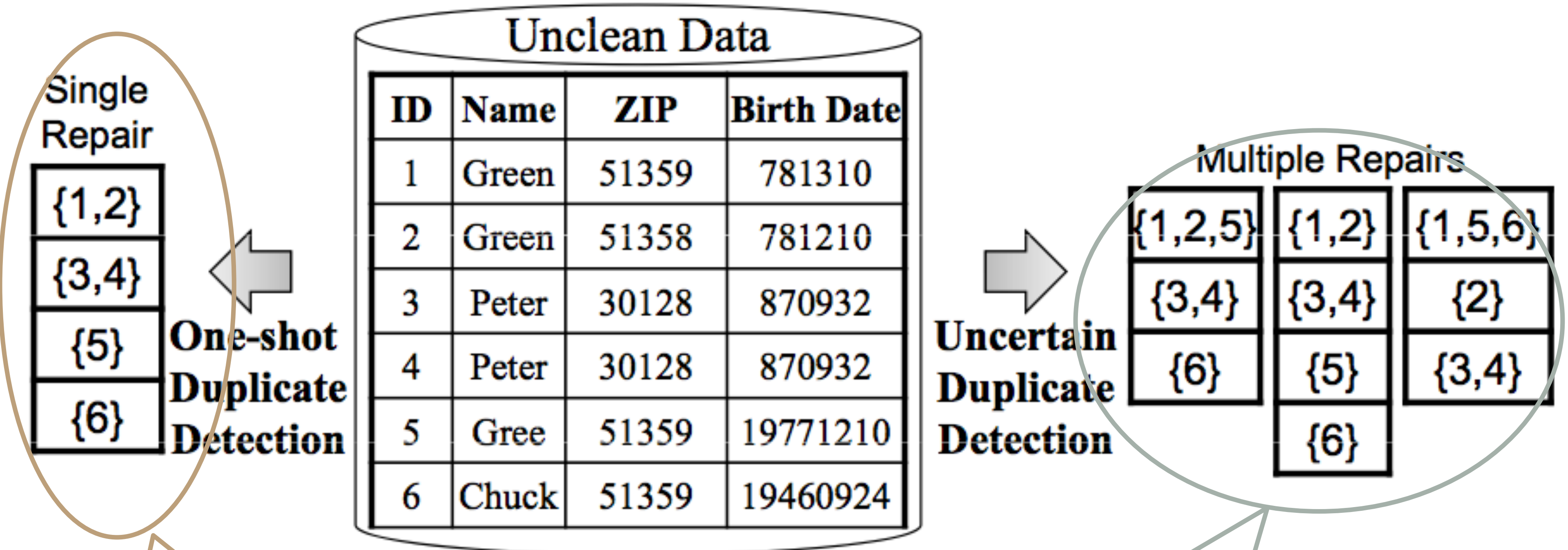
Standard DB: NULL

Probabilistic DB:
distribution on possible values

| id | age | edu | inc | nw | prob |
|---|---|---|---|---|---|
| $t_{12}.1$ | 30 | MS | 50K | 100K | 0.30 |
| $t_{12}.2$ | 30 | MS | 50K | 500K | 0.45 |
| $t_{12}.3$ | 30 | MS | 100K | 100K | 0.10 |
| $t_{12}.4$ | 30 | MS | 100K | 500K | 0.15 |

[Beskales'2009]

# Ex 4: Data Cleaning



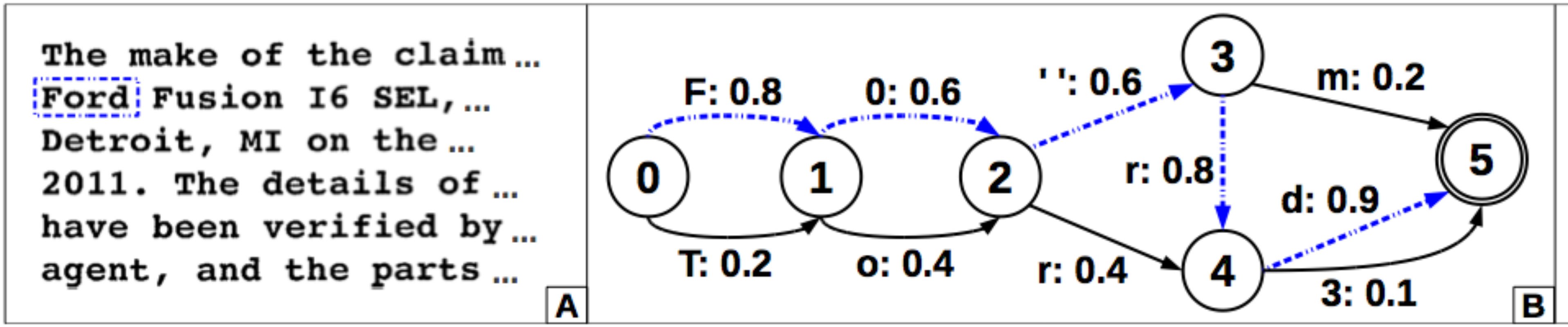

| ID | Name | ZIP | Birth Date |
|---|---|---|---|
| 1 | Green | 51359 | 781310 |
| 2 | Green | 51358 | 781210 |
| 3 | Peter | 30128 | 870932 |
| 4 | Peter | 30128 | 870932 |
| 5 | Gree | 51359 | 19771210 |
| 6 | Chuck | 51359 | 19460924 |

Standard DB
cleaning data means choosing one possible repair

Probabilistic DB
keep many/all possible repairs

[Kumar&Re'2012]

# Ex 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

# Terminology

- Types of uncertainty:
  - Tuple uncertainty = tuple is present/absent ← this talk
  - Attribute uncertainty = attribute value is a pdf
- Types of probability distrubtion
  - Independent tuples ← this talk
  - Disjoint/independent tuples
  - Correlations: Markov/Bayesian Networks, Markov Logic Networks
- Types of queries
  - SQL SPJU, or Unions of Conjunctive Queries ← this talk
  - Extensions with explicit reference to probabilities

# Tuple-Independent Databases

Every tuple t in D = independent random variable

R(x) = "x was at the meeting"

S(x,y) = "x carried laptop y"

R(A)

| A | |
|---|---|
| a1 | .4 |
| a2 | .2 |
| a3 | .5 |

S(A,B)

| A | B | |
|---|---|---|
| a1 | b1 | .7 |
| a2 | b1 | .4 |
| a2 | b2 | .2 |
| a2 | b3 | .2 |
| a3 | B2 | .5 |

# SELECT-PROJECT-JOIN-UNION

Did anyone bring a laptop to the meeting ?

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

$$Q = \exists x. \exists y. R(x) \wedge S(x,y) \equiv Q = R(x), S(x,y)$$

Answer: P(Q) = 0.35

# The Query Evaluation Problem

Given database D, query Q

Compute P(Q)

- Q is *small*, the database D is *huge*
- Related to *model counting*, and *probabilistic inference*, in AI and model checking; #P-hard in general
- Novel approach possible in probabilistic databases

# Outline

- Introduction
- Motivation and Background
- Extensional Query Evaluation
- Intensional Query Evaluation
- Conclusions

# Extensional Query Evaluation

Relational operators are modified to handle probabilities:

1. Join
2. Projection w/ duplicate elimination
3. Union
4. Inclusion/Exclusion
5. Selection

# An Example

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

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

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

S(A,B)

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

R(A)

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

# 1. Join operator

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

Running on a standard RDBMS

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

$\bowtie$

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 |

# 2. Projection w/ duplicate elimination

| A | P |
|---|---|
| a1 | 1 - (1-p1)*(1-p2) |
| a2 | 1 - (1-p3)*(1-p4)*(1-p5) |

$\Pi_A$

S(A,B)

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

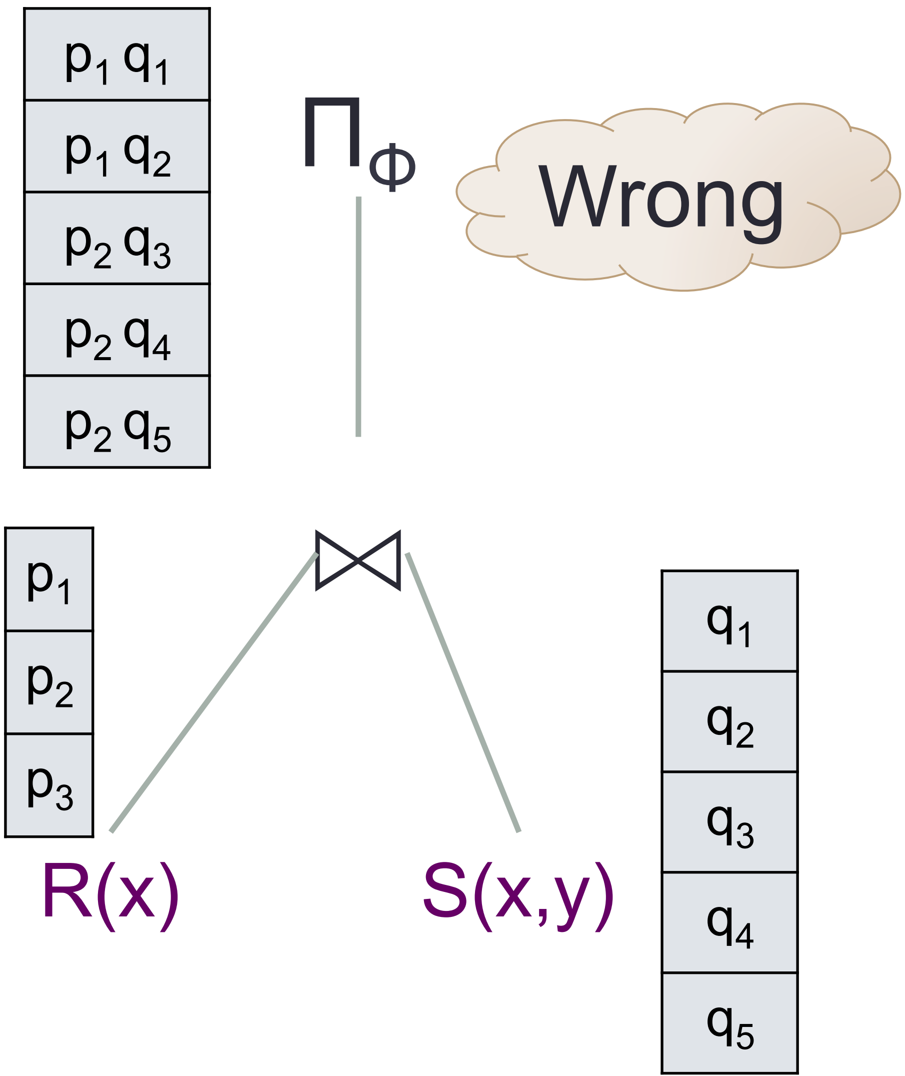
Running on a standard RDBMS
(need to write the aggregate prod):

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

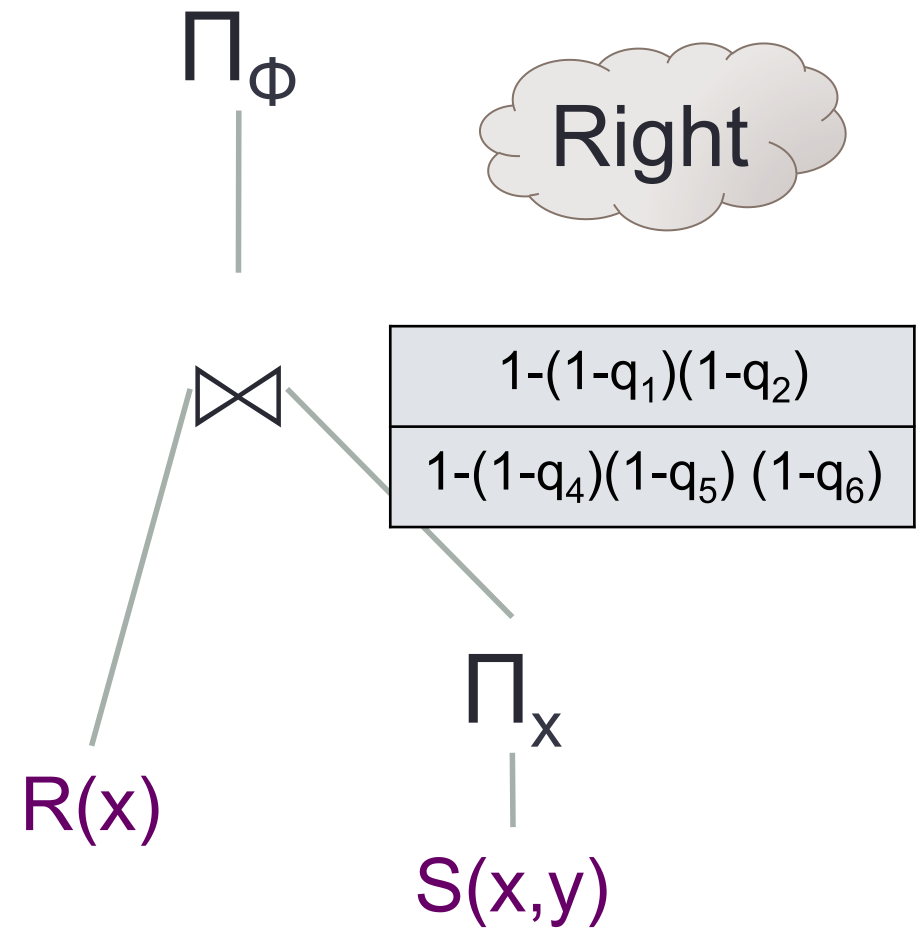
# $Q = R(x), S(x,y)$

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

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




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

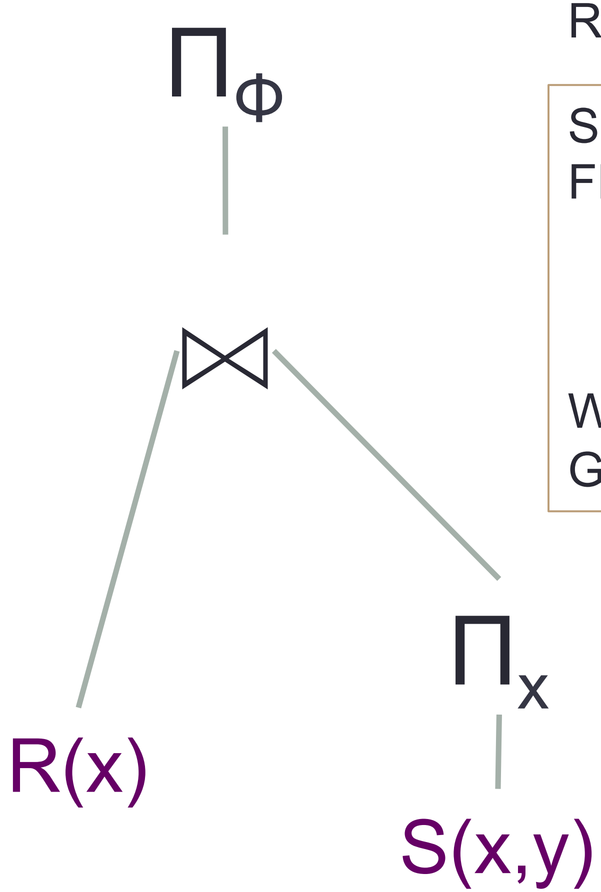


# Lesson 1

- Plans that are equivalent may be different when interpreted as extensional plans
- A correct extensional plan is called a *safe plan*
- Need to find a safe plan!

# Using a Standard RDBMS

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




Running on a standard RDBMS

```
SELECT 'yes' as z, 1.0-prod(1.0 – R.P * Temp.P) as p
FROM R,
          (SELECT S.x, 1.0-prod(1.0 - S.P) as P
           FROM   S
           GROUP BY S.x) Temp
WHERE R.x = Temp.x
GROUP BY 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!)

# 3. Union

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

∪

R(A)

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

T(A)

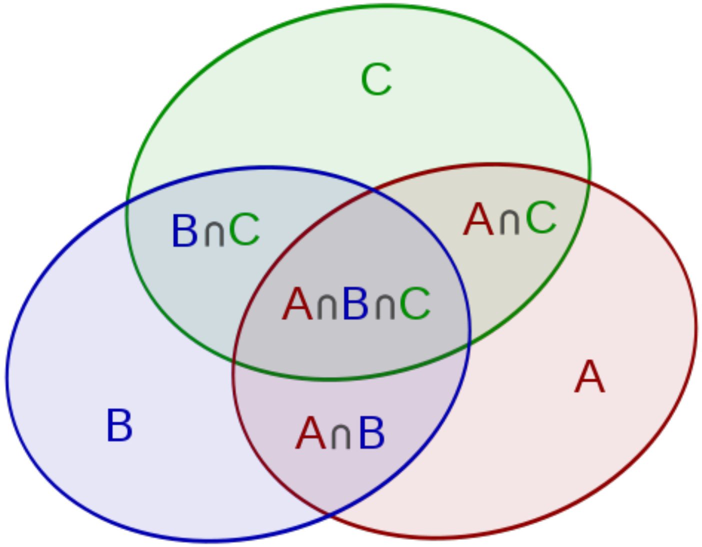
| A | P |
|---|---|
| a2 | q2 |
| a3 | q3 |
| a4 | q4 |

Running on a standard RDBMS

```
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;
```

# 4 Inclusion-exclusion Formula

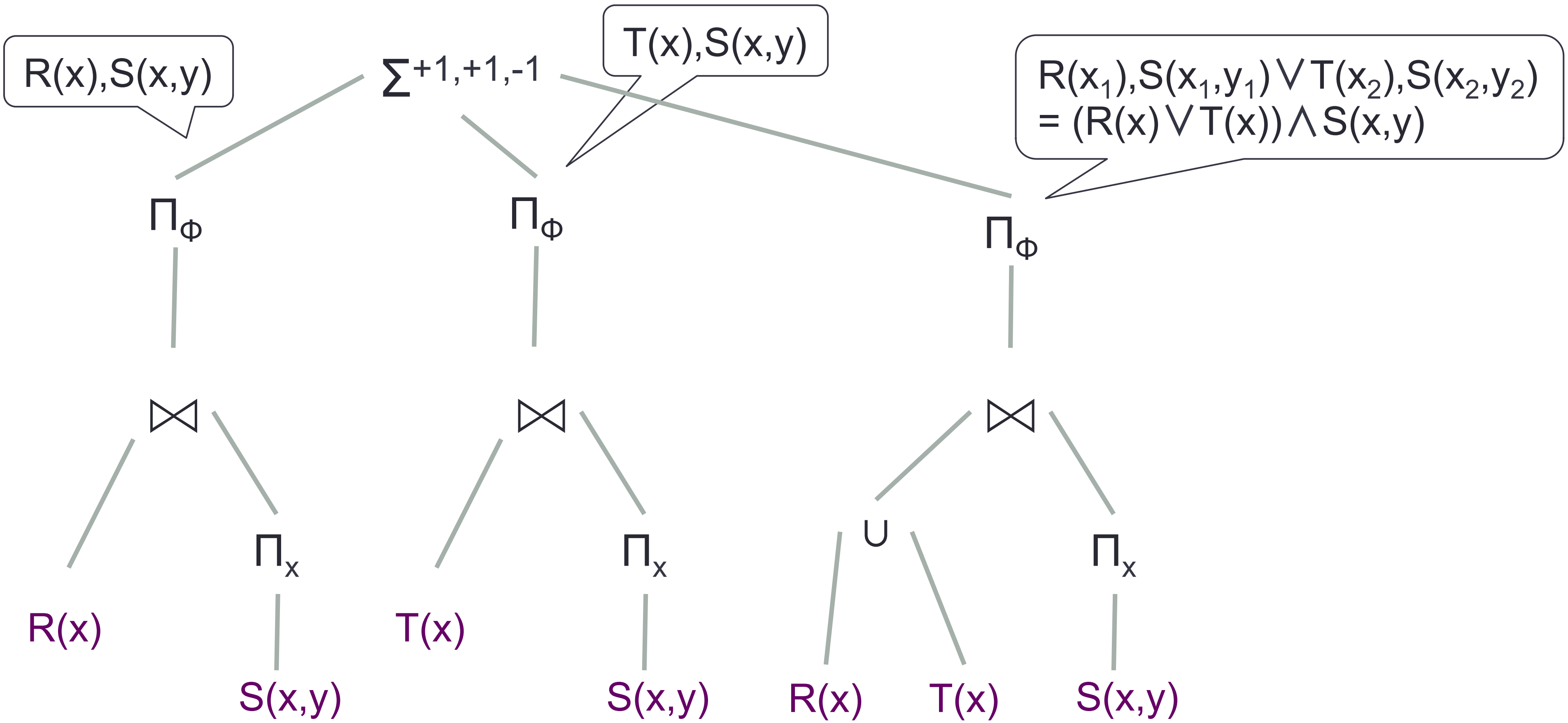
$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$



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

```
SELECT DISTINCT 'yes'
FROM R r, S s1, T t, S s2
WHERE r.x = s1.x
      and t.x = s2.x
```

$$P(Q_J) = P(q_1, q_2) = P(q_1) + P(q_2) - P(q_1 \vee q_2)$$



# Lesson 3

We need unions in order to handle self-joins!

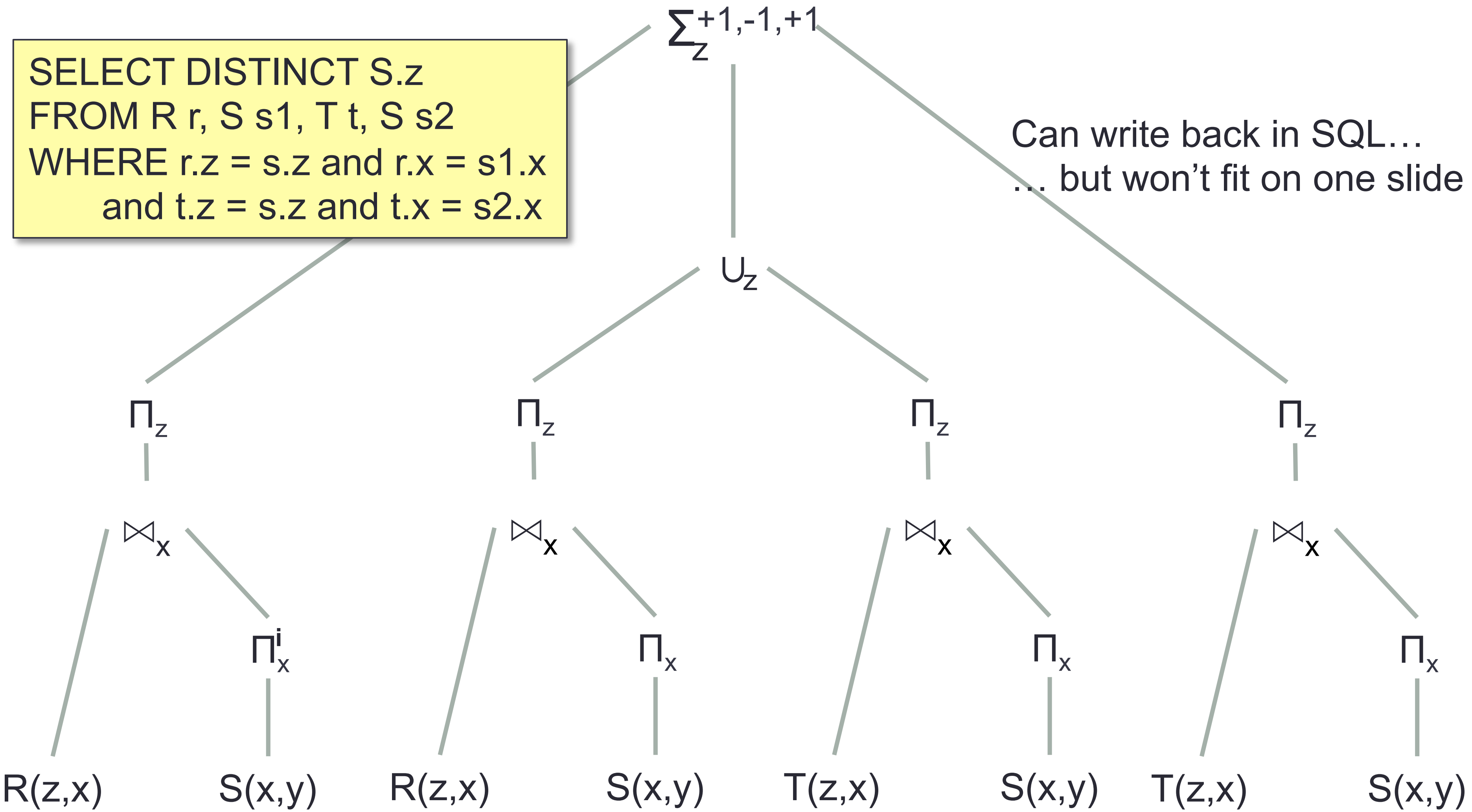
- SPJ = not a “natural” class of queries for probDB
- SPJU = the “natural” class of queries

We need the inclusion/exclusion formula

- Today’s probabilistic inference systems do NOT use inclusion/exclusion; they cannot have guaranteed running time bounds on such queries
- More on this later…

# Putting it Together

```
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

# When It All Fails

R

| B | P |
|---|---|
| x1 | p1 |
| x2 | p2 |

S

| B | C |
|---|---|
| x1 | y1 |
| x1 | y2 |
| x2 | y2 |

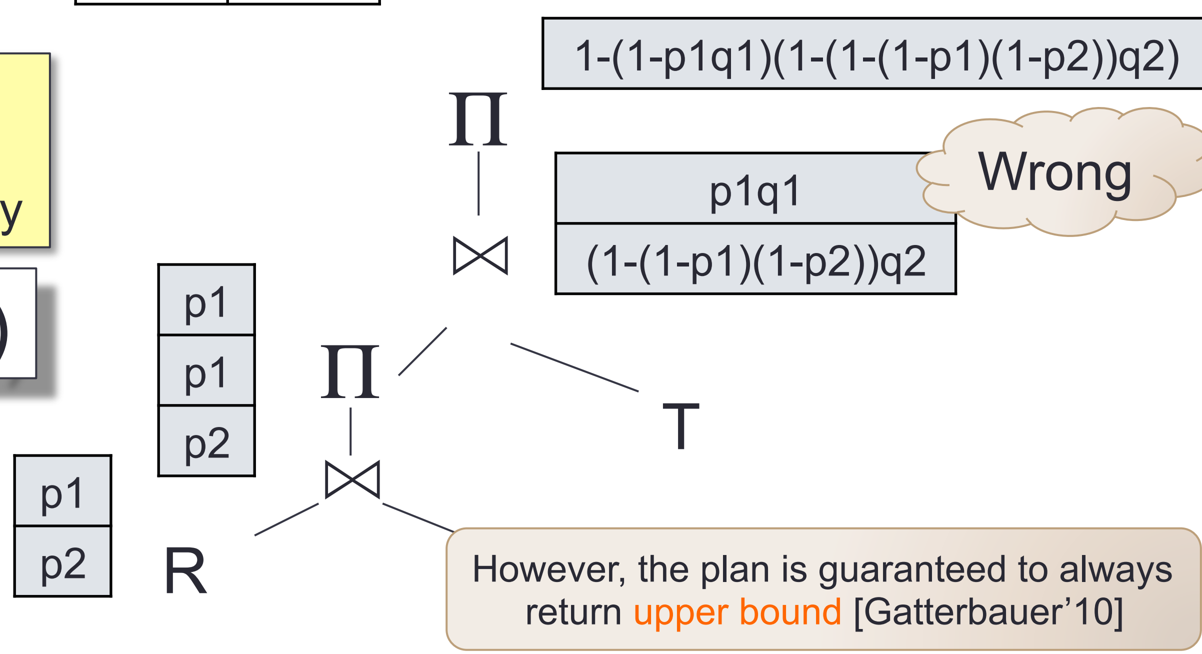
T

| C | 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



## PTIME Queries

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

R(x, y), S(y), T('a', y)

R(x), S(x, y), T(y), U(u, y), W('a', u)

. . .

## #P-Hard Queries

hd1 = R(x), S(x, y), T(y)

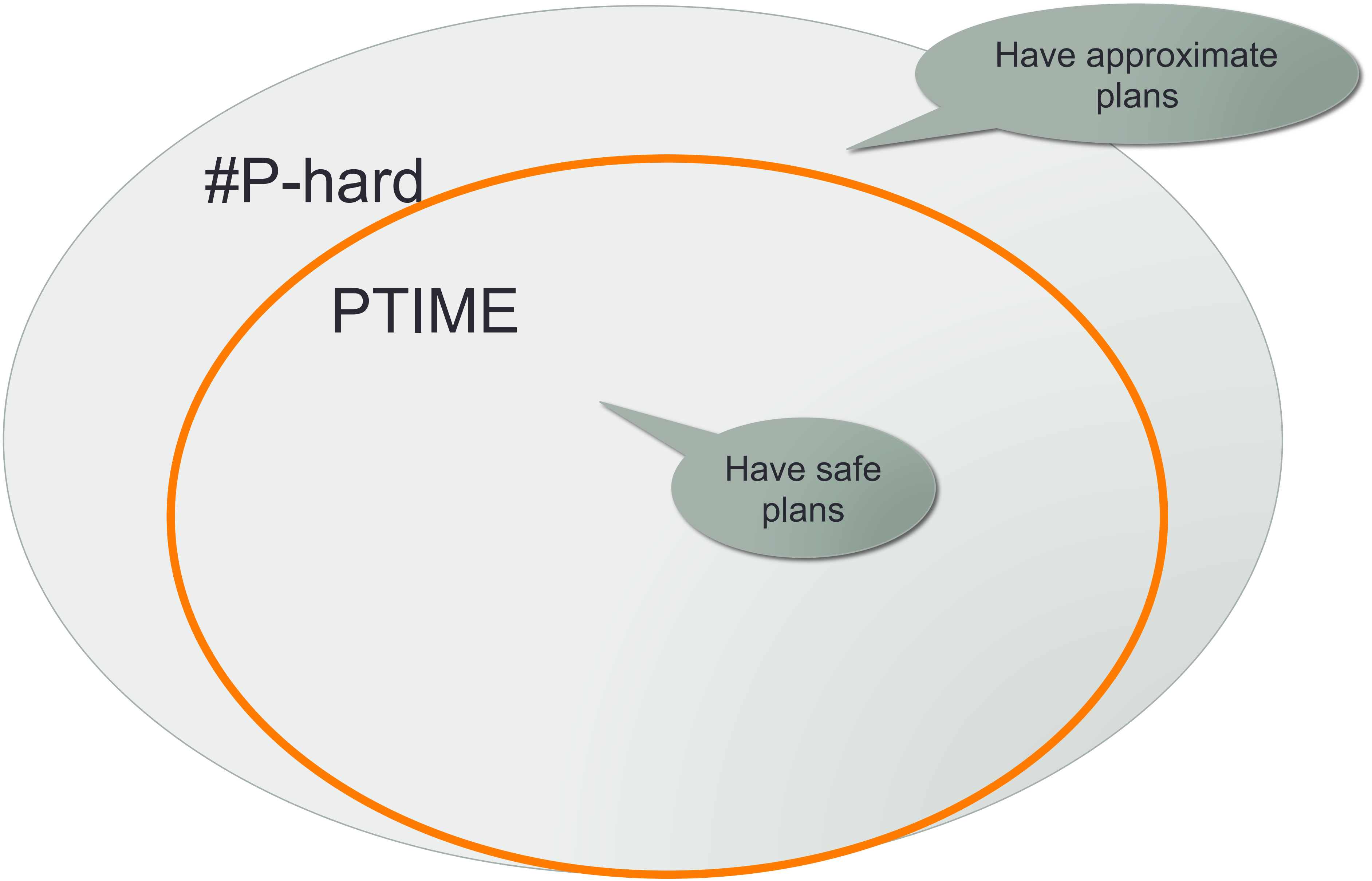
hd2 = R(x,y), S(y)

hd3 = R(x,y), S(x,y)

. . .

**Theorem**. Every SPJU query has safe plan, or #P-hard.

# Landscape of SPJU Queries so far…



# Summary: Extensional Query Evaluation

- Relational operators are modified to handle probabilities:
  - Join, projection, union, inclusion/exclusion, selection
- *Safe* plan =computes probabilities correctly
- Every safe plan can converted into standard SQL and run on your favorite RDBMS
- Some cool facts along the way:
  - Inclusion/exclusion formula
  - Using unions to handle self-joins

# Outline

- Introduction
- Motivation and Background
- Extensional Query Evaluation
- Intensional Query Evaluation
- Conclusions

# Background: Model Counting

Let F be a Boolean formula over Boolean variables $X_1, X_2, \ldots$

#F = number of satisfying assignments

Model counting problem: compute #F.

Valiant'79: Model counting is #P-hard

P(F) denotes the probability, assuming $X_1, X_2, \ldots$ are set to true independently, with known probabilities

Factoid: if $P(X_1) = P(X_2) = \ldots = ½$ then $P(F) = \#F / 2^n$

$F = X\,Y \vee Y\,Z \vee X\,Z$

$\#F = 4$

$$
\begin{aligned}
P(F) = {} & P(X)\times P(Y) \\
& + P(X)\times P(Z) \\
& + P(X)\times P(Z) \\
& - 2\times P(X)\times P(Y)\times P(Z)
\end{aligned}
$$

# Background: DPLL

Modern model counting systems:

- `c2d` [Huang and Darwiche, 2007], `Dsharp` [Muise et al., 2012]
- Based on Davis, Putnam, Logemann, Loveland, see [Gomes et al., 2009]

// 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 $P(F_1) \times P(F_2)$

# Intentional Query Evaluation

Query $Q$ + database $D$ = *lineage expression* $F$

- Boolean variables $X_1$, $X_2$, … correspond to tuples $t_1$, $t_2$, …
- The lineage $F$ says when "$Q$ is true"

Compute $P(F)$ using a DPLL-style system

- `c2d` or `Dsharp` or …

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

# 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 [Golumbic'2004]

P(F) can be computed in linear time:

$$P(F_1 \wedge F_2) = P(F_1) \times P(F_2)$$

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

Modify the model counter to check for read-once formulas [Sen'2010], [Roy'2011]

# 1. Read-Once Boolean Formulas

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

SELECT DISTINCT 'yes'
FROM R, S WHERE R.x = S.x

R

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

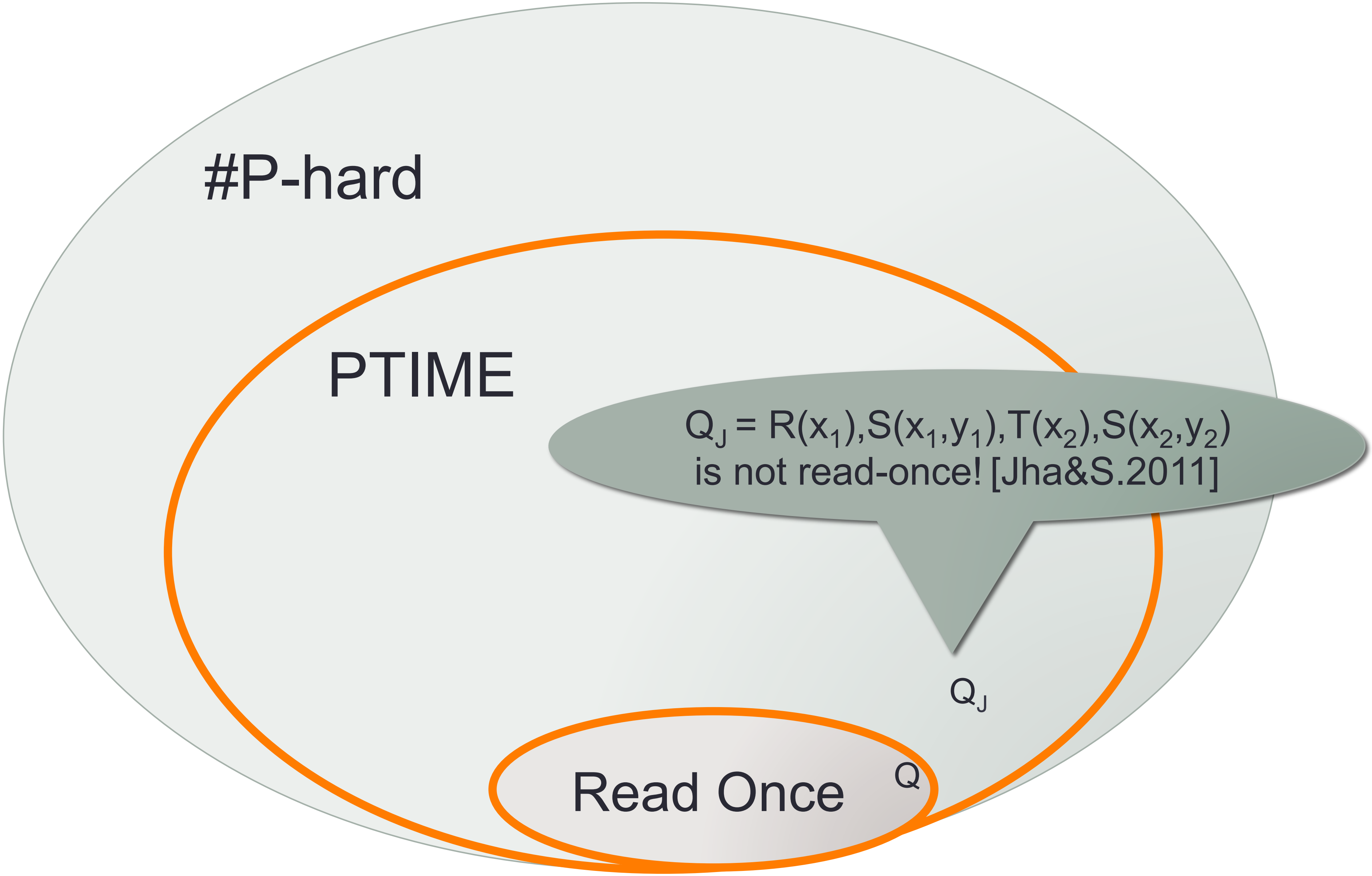
S

| A | B | |
|---|---|---|
| a1 | b1 | Y1 |
| a1 | b1 | Y2 |
| a2 | b2 | Y3 |
| a2 | b3 | Y4 |
| a2 | B2 | Y5 |

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

Read-once

# Landscape of SPJU



# 2. Ordered Binary Decision Diagrams

Branching program, BP = a rooted DAG where:

- Each internal node tests a Boolean variable X and has two outgoing edges labeled 0 and 1
- Each sink node is labeled 0 or 1
- Every path tests every variable X only once

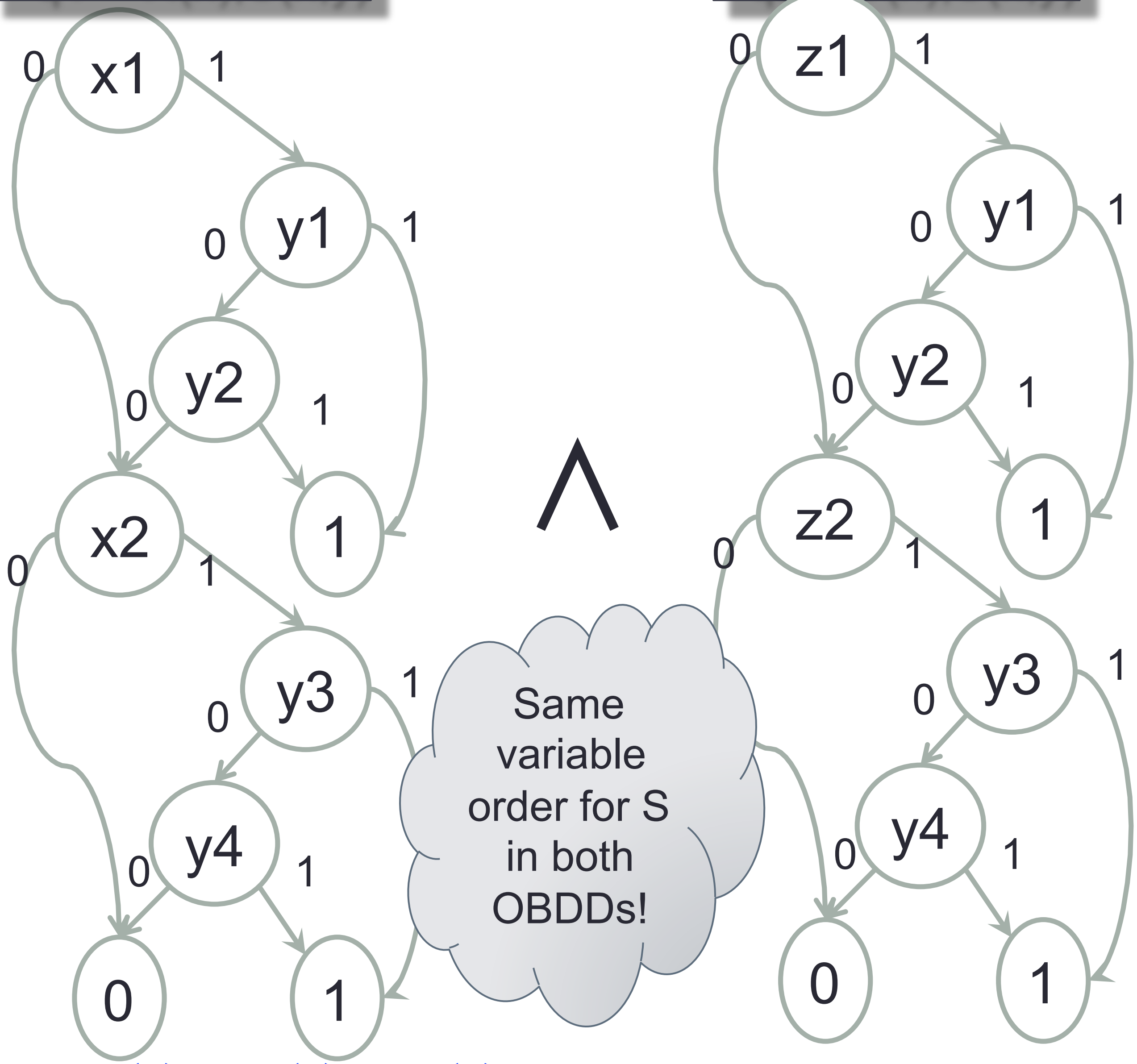
OBDD = a BP where:

- Every path tests the variables in the same order

Fact: any DPLL-with-caching procedure that tests the variables in the same order has a trace that is an OBDD q1 = R(x),S(x,y)

q2 = T(x),S(x,y)

$Q_J = q1,q2$



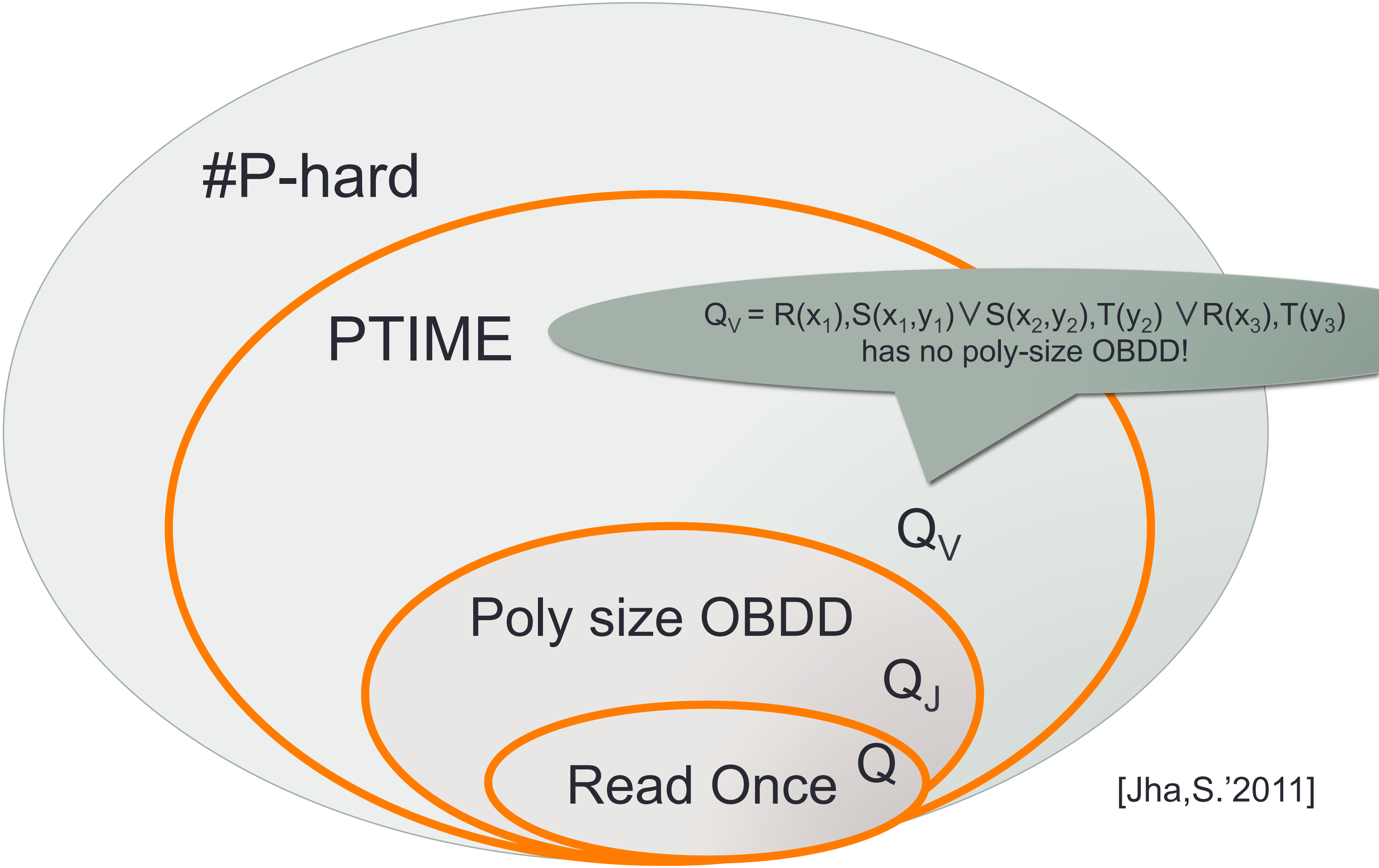

=

Efficient OBDD synthesis is possible when they use the same variable order [Wegener'2000]

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

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

# Landscape of SPJU

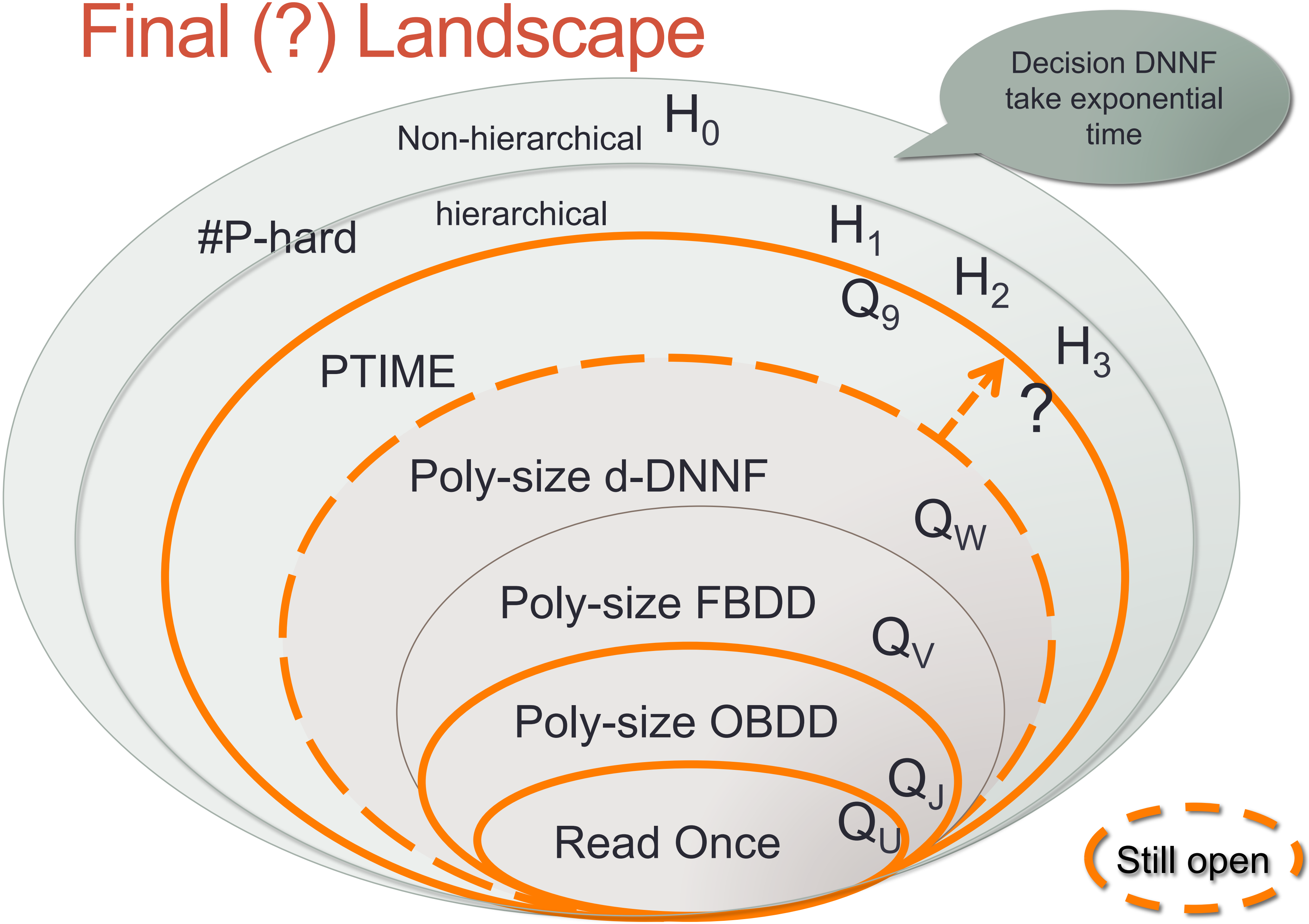



[Jha,S.'2011]

# 3. Other Types of DPLL Traces

- FBDD = binary decision diagrams where different paths may test variables in different orders
- Decision-DNNF's = [equivalent to] binary decision diagrams extended with independent $\wedge$ [Darwiche'2007]
- d-DNNF's = even more powerful… [Darwiche'2000]

# Final (?) Landscape



# Outline

- Introduction
- Motivation and Background
- Extensional Query Evaluation
- Intensional Query Evaluation
- 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.

# Thank You !

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