Probabilistic Database Query Evaluation

CS267A - Fall 2018
Guy Van den Broeck
Query evaluation following the semantics directly is intractable!

\[ P(Q) = p_1 p_2 p_3 q_1 q_2 q_3 q_4 q_5 + p_1 (1-p_2) p_3 q_1 (1-q_2) q_3 (1-q_4) q_5 + (1-p_1)(1-p_2)p_3 q_1 (1-q_2) q_3 q_4 q_5 + \ldots \]

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{ Friend}(x,y) \]
Can we come up with a query evaluation algorithm that reasons efficiently at the first-order level?
Example Queries
Example Queries

Q = Scientist(John)
Example Queries

\[ Q = \exists x \text{ Scientist}(x) \]
Example Queries

\[ Q = \exists x \text{ Scientist}(x) \land \text{Handsome}(x) \]
Example Queries

\[ Q = \exists x \exists y \text{ Scientist}(x) \land \text{Coauthor}(x, y) \]
Probabilistic Query Evaluation

\[ Q = \exists x \exists y \text{Scientist}(x) \land \text{Coauthor}(x,y) \]

\[ P(Q) = 1 - \{ 1 - p_1 \cdot \left[ 1 - (1 - q_1) \cdot (1 - q_2) \right] \} \cdot \{ 1 - p_2 \cdot \left[ 1 - (1 - q_3) \cdot (1 - q_4) \cdot (1 - q_5) \right] \} \]

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

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>( q_5 )</td>
</tr>
</tbody>
</table>
Example Queries

\[ Q = \exists x \exists y \text{ Scientist}(x) \land \text{ Coauthor}(x,y) \lor \exists x \text{ Singer}(x) \land \text{ Handsome}(x) \]
Example Queries

\[ Q = \exists x \exists y \text{ Scientist}(x) \land \text{Coauthor}(x, y) \]
\[ \lor \exists x \text{ Scientist}(x) \land \text{Handsome}(x) \]
Example Queries

\[ Q_J = \forall x_1 \forall y_1 \forall x_2 \forall y_2 \ (S(x_1,y_1) \lor R(y_1) \lor S(x_2,y_2) \lor T(y_2)) \]
Query Eval

\[ Q_J = \forall x_1 \forall y_1 \forall x_2 \forall y_2 \ (S(x_1, y_1) \lor R(y_1) \lor S(x_2, y_2) \lor T(y_2)) \]

\[ = [\forall x_1 \forall y_1 S(x_1, y_1) \lor R(y_1)] \lor [\forall x_2 \forall y_2 S(x_2, y_2) \lor T(y_2)] \]

\[ P(Q_J) = P(Q_1) + P(Q_2) - P(Q_1 \land Q_2) \]

\[ Q_1 \land Q_2 = \forall y \ [\forall x_1 S(x_1, y) \lor R(y)] \land (\forall x_2 S(x_2, y)) \lor T(y)] \]

\[ = \forall y \ [\forall x S(x, y) \lor (R(y) \land T(y))] \]

\[ P(Q_1 \land Q_2) = \prod_{B \in \text{Domain}} P[\forall x. S(x, B) \lor (R(B) \land T(B))] = \ldots \text{etc} \]

Runtime = \( O(n^2) \).
Lifted Inference Rules

Preprocess $Q$ (omitted),
Then apply rules (some have preconditions; easy to check)

1. $P(\neg Q) = 1 - P(Q)$

2. $P(Q_1 \land Q_2) = P(Q_1) \cdot P(Q_2)$
   $P(Q_1 \lor Q_2) = 1 - (1 - P(Q_1))(1 - P(Q_2))$

3. $P(\exists z \ Q) = 1 - \prod_{A \in \text{Domain}} (1 - P(Q[A/z]))$
   $P(\forall z \ Q) = \prod_{A \in \text{Domain}} P(Q[A/z])$

4. $P(Q_1 \land Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \lor Q_2)$
   $P(Q_1 \lor Q_2) = P(Q_1) + P(Q_2) - P(Q_1 \land Q_2)$

Negation
Decomposable $\land, \lor$
Decomposable $\exists, \forall$
Inclusion/exclusion

[Suciu’11]
Algorithm 1 Lift\(^R\)\((Q, P)\), abbreviated by L\((Q, P)\)

**Input:** CNF Q and probabilistic tuples \(P\).

**Output:** The probability \(P_P(Q)\).

1. **Step 0** Base of Recursion
2. \(\text{if } Q \text{ is a single ground atom } t \text{ then} \)  
3. \(\text{if } \langle t : p \rangle \in P \text{ then return } p \text{ else return } 0\)

4. **Step 1** Rewriting of Query
5. Convert Q to union of CNFs: \(Q_{\text{UCNF}} = Q_1 \lor \ldots \lor Q_m\)

6. **Step 2** Decomposable Disjunction
7. \(\text{if } m > 1 \text{ and } Q_{\text{UCNF}} = Q_1 \lor Q_2 \text{ where } Q_1 \perp Q_2 \text{ then} \)  
8. \(q_1 \leftarrow L(Q_1, P|_{Q_1}) \text{ and } q_2 \leftarrow L(Q_2, P|_{Q_2})\)
9. \(\text{return } 1 - (1 - q_1) \cdot (1 - q_2)\)

10. **Step 3** Inclusion-Exclusion
11. \(\text{if } m > 1 \text{ but } Q_{\text{UCNF}} \text{ has no independent } Q_i \text{ then} \)  
12. \(\text{return } \sum_{s \subseteq m} (-1)^{|s|+1} \cdot L(\wedge_{i \in s} Q_i, P|_{\wedge_{i \in s} Q_i})\)
13. \(\triangleright \) (while performing cancellations)

14. **Step 4** Decomposable Conjunction
15. \(\text{if } Q = Q_1 \land Q_2 \text{ where } Q_1 \perp Q_2 \text{ then} \)  
16. \(\text{return } L(Q_1, P|_{Q_1}) \cdot L(Q_2, P|_{Q_2})\)

17. **Step 5** Decomposable Universal Quantifier
18. \(\text{if } Q \text{ has a separator variable } x \text{ then} \)  
19. \(\text{let } T \text{ be all constants as } x\text{-argument in } P\)
20. \(\text{return } \prod_{t \in T} L(Q[x/t], P|_{x=t})\)

21. **Step 6** Fail
Lifted Inference Algorithm: Step 0

1: **Step 0**  *Base of Recursion*
2: \[\text{if } Q \text{ is a single ground atom } t \text{ then}\]
3: \[\text{if } \langle t : p \rangle \in \mathcal{P} \text{ then return } p \text{ else return } 0\]
Lifted Inference Algorithm: Step 1

4: **Step 1** Rewriting of Query

5: Convert Q to union of CNFs: $Q_{UCNF} = Q_1 \lor ... \lor Q_m$
Lifted Inference Algorithm: Step 2

6: **Step 2 Decomposable Disjunction**

7: \[ \text{if } m > 1 \text{ and } Q_{\text{UCNF}} = Q_1 \lor Q_2 \text{ where } Q_1 \perp Q_2 \text{ then} \]

8: \[ q_1 \leftarrow L(Q_1, P|Q_1) \text{ and } q_2 \leftarrow L(Q_2, P|Q_2) \]

9: \[ \text{return } 1 - (1 - q_1) \cdot (1 - q_2) \]
Lifted Inference Algorithm: Step 3

10: **Step 3** *Inclusion-Exclusion*
11: if $m > 1$ but $Q_{UCNF}$ has no independent $Q_i$ then
12: return $\sum_{s \subseteq m} (-1)^{|s|+1} \cdot L(\land_{i \in s} Q_i, P|_{\land_{i \in s} Q_i})$
Lifted Inference Algorithm: Step 4

14: **Step 4** Decomposable Conjunction
15: \textbf{if} Q = Q_1 \land Q_2 \textbf{ where } Q_1 \perp Q_2 \textbf{ then}
16: \textbf{return} L(Q_1, \mathcal{P}|Q_1) \cdot L(Q_2, \mathcal{P}|Q_2)
Lifted Inference Algorithm: Step 5

17: **Step 5**  *Decomposable Universal Quantifier*
18:  if Q has a separator variable $x$ then
19:  let $T$ be all constants as $x$-argument in $\mathcal{P}$
20:  return $\prod_{t \in T} L(Q[x/t], \mathcal{P}|_{x=t})$
Lifted Inference Algorithm: Step 6

21: **Step 6** *Fail*

We say the query is not “liftable”
Probabilistic Query Evaluation

\[ Q = \exists x \exists y \text{Scientist}(x) \land \text{Coauthor}(x, y) \]

\[ P(Q) = 1 - \left\{ 1 - p_1 \left[ 1 - (1 - q_1)(1 - q_2) \right] \right\} \times \left\{ 1 - p_2 \left[ 1 - (1 - q_3)(1 - q_4)(1 - q_5) \right] \right\} \]

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

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>( q_1 )</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>( q_2 )</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>( q_3 )</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>( q_4 )</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>( q_5 )</td>
</tr>
</tbody>
</table>
An Example

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

\[ Q = \exists x \exists y \text{ Smoker}(x) \land \text{Friend}(x, y) \]

\[ \Pi_\Phi \]

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

\[ \text{Smoker}(x) \]

\[ \Pi_x \]

\[ \text{Friend}(x, y) \]

<table>
<thead>
<tr>
<th>x</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>p_1</td>
</tr>
<tr>
<td>B</td>
<td>p_2</td>
</tr>
<tr>
<td>C</td>
<td>p_3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>q_1</td>
</tr>
<tr>
<td>A</td>
<td>E</td>
<td>q_2</td>
</tr>
<tr>
<td>B</td>
<td>F</td>
<td>q_3</td>
</tr>
<tr>
<td>B</td>
<td>G</td>
<td>q_4</td>
</tr>
<tr>
<td>B</td>
<td>H</td>
<td>q_5</td>
</tr>
</tbody>
</table>
Example Queries (now as clause)

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

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

The decomposable \(\forall\)-rule:

$$P(\forall z Q) = \prod_{A \in \text{Domain}} P(Q[A/z])$$

... does not apply:

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

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

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

You reach Step 6 and fail! 😞
Two Questions

• **Question 1:** Are the lifted rules complete?
  - We know that they get stuck on some queries
    Some queries are not “liftable”.
  - Should we add more rules?

• **Question 2:** Are lifted rules stronger than grounded?
  - Advanced ground graphical model inference
  - Any advantage over grounded inference?
Hardness and Completeness
NP vs. #P

• Decision problems are a set of questions.
• For some subset the answer is “yes”.
• A problem is in NP iff
  – For each “yes” question, there exists a certificate
  – That certificate is small (polynomial in |question|)
  – There exists a verifier (an algorithm) that tells you whether the certificate guarantees that the answer should be “yes”.
  – That verifier is efficient to run.
NP vs. #$P$

#$P$ is taking a verifier and counting:

*How many certificates are there for each question?*
Model Counting

- Model = solution to a propositional logic formula $\Delta$
- Model counting = $\#\text{SAT}$

$\Delta = (\text{Rain} \Rightarrow \text{Cloudy})$

<table>
<thead>
<tr>
<th>Rain</th>
<th>Cloudy</th>
<th>Model?</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>Yes</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[Valiant] $\#P$-hard, even for 2CNF

$\#\text{SAT} = 3$
NP vs. #P

Decision Problems:
• SAT = Satisfiability Problem
• SAT is NP-complete [Cook’71]

Counting Problems:
• #SAT = model counting
• #SAT is #P-complete [Valiant’79]

Note: it would be wrong to say “#SAT is NP-complete”
Is this query hard?

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

How do I prove a problem is NP or #P hard?
Background: Positive Partitioned 2CNF

A PP2CNF is:

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

where $E$ = the edge set of a bipartite graph

$$F = (x_1 \lor y_1) \land (x_2 \lor y_1) \land (x_2 \lor y_3) \land (x_1 \lor y_3) \land (x_2 \lor y_2)$$

**Theorem:** $\#\text{PP2CNF}$ is $\#\text{P}$-hard [Provan’83]
Unliftable Clause

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

**Theorem.** Computing $P_D(H_0)$ is \#P-hard in the size of $D$

*Proof:* PP2CNF: $F = (X_{i1} \lor Y_{j1}) \land (X_{i2} \lor Y_{j2}) \land \ldots$ reduce $\#F$ to computing $P_D(H_0)$

By example:

$F = (X_1 \lor Y_1) \land (X_1 \lor Y_2) \land (X_2 \lor Y_2)$

$P_D(H_0) = P(F)$; hence $P_D(H_0)$ is \#P-hard

- **Smoker**
  - $x_1$: 0.5
  - $x_2$: 0.5

- **Friend**
  - $x_1$: 0.5
  - $x_2$: 0.0
  - $y_1$: 0.0
  - $y_2$: 0.0

- **Jogger**
  - $y_1$: 0.5
  - $y_2$: 0.5

Independent Project not possible:
For $A_1 \neq A_2$, $H_0[A_1/x]$ and $H_0[A_2/x]$ are dependent!
Details of Proof

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

Grounding over a domain of \{A,B,C\}

- Smoker(A) \lor \text{ Friend}(A,A) \lor \text{ Jogger}(A)
- Smoker(A) \lor \text{ Friend}(A,B) \lor \text{ Jogger}(B)
- Smoker(A) \lor \text{ Friend}(A,C) \lor \text{ Jogger}(C)

- Smoker(B) \lor \text{ Friend}(B,A) \lor \text{ Jogger}(A)
- Smoker(B) \lor \text{ Friend}(B,B) \lor \text{ Jogger}(B)
- Smoker(B) \lor \text{ Friend}(B,C) \lor \text{ Jogger}(C)

- Smoker(C) \lor \text{ Friend}(C,A) \lor \text{ Jogger}(A)
- Smoker(C) \lor \text{ Friend}(C,B) \lor \text{ Jogger}(B)
- Smoker(C) \lor \text{ Friend}(C,C) \lor \text{ Jogger}(C)
Details of Proof

Suppose I know that
• Friend(A,A)
• Friend(A,C)
• Friend(B,C)
• Friend(C,B)
are all true

That is, they have 100% probability
Details of Proof

Suppose I know that
- Friend(A,B)
- Friend(B,A)
- Friend(B,B)
- Friend(C,A)
- Friend(C,C)
are all false

That is, they have 0% probability
Details of Proof

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

Given the friendships we assumed before, query \( H_0 \) is true precisely when this CNF is true.

This CNF is a PP2CNF where Smoker atoms are \( X \)-variables and Jogger atoms are \( Y \)-variables.
Details of Proof

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

<table>
<thead>
<tr>
<th>( X_A \lor Y_B )</th>
<th>( X_B \lor Y_A )</th>
<th>( X_B \lor Y_B )</th>
<th>( X_C \lor Y_A )</th>
<th>( X_C \lor Y_C )</th>
</tr>
</thead>
</table>

This CNF is a PP2CNF where Smoker atoms are \( X \)-variables and Jogger atoms are \( Y \)-variables.

- Let \( K \) be the number of models of this PP2CNF (the \#PP2CNF count)
- This is also the number of worlds in which the query \( H_0 \) is true
- Assume that all (non-zero probability) possible worlds of the probabilistic database have exactly the same probability \( p \)

- Then \( P(H_0) = \sum_{\omega \models H_0} P(\omega) = \sum_{\omega \models H_0} p = K \cdot p \)
Details of Proof

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

- If we were able to compute \( P(H_0) \), then we could solve for \( K = P(H_0)/p \)
- Detail: to make all worlds have probability \( p \), we need to assign probability 0.5 to every tuple in Smoker and Jogger

<table>
<thead>
<tr>
<th>Smoker</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td></td>
</tr>
<tr>
<td>( x_2 )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Friend</th>
<th>( y )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>( Y )</td>
<td>( P )</td>
</tr>
<tr>
<td>( y_1 )</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>( y_2 )</td>
<td></td>
<td>0.5</td>
</tr>
</tbody>
</table>

- Computing \( K \) is a \#PP2CNF problem which is \#P-complete in general
- By choosing the Friendship relation we can construct a database corresponding to any arbitrary PP2CNF
- Hence computing \( P(H_0) \) is \#P-hard.
Unliftable Clause

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

**Theorem.** Computing \( P_D(H_0) \) is \#P-hard in the size of \( D \)

**Proof:** PP2CNF: \( F = (X_{i1} \lor Y_{j1}) \land (X_{i2} \lor Y_{j2}) \land \ldots \) reduce \#F to computing \( P_D(H_0) \)

By example:

\[ F = (X_1 \lor Y_1) \land (X_1 \lor Y_2) \land (X_2 \lor Y_2) \]

\[ P_D(H_0) = P(F); \] hence \( P_D(H_0) \) is \#P-hard

\begin{align*}
| X | P & | X | Y | P | \quad | X | Y | P | \\
|----|----|----|----|----|----|----|----|----|
| x_1 | 0.5 | x_1 | y_1 | 0 | x_1 | y_1 | 0 | \quad | y_1 | 0.5 | \\
| x_2 | 0.5 | x_1 | y_2 | 0 | x_2 | y_2 | 0 | \quad | y_2 | 0.5 |
\end{align*}
Are the Lifted Rules Complete?

You already know:

• Inference rules: PTIME data complexity
• There is a query with \#P-hard data complexity

What about all other cases?

[Dalvi and Suciu; JACM’11]
Hierarchical Queries

Fix $Q$; $\text{at}(x) =$ set of atoms (=literals) containing the variable $x$

**Definition** $Q$ is hierarchical if for all variables $x$, $y$:

$\text{at}(x) \subseteq \text{at}(y)$ or $\text{at}(x) \supseteq \text{at}(y)$ or $\text{at}(x) \cap \text{at}(y) = \emptyset$

**Hierarchical**

$Q = \forall x \forall y \forall z (S(x,y) \lor T(x,z))$

**Non-hierarchical**

$H_0 = \forall x \forall y (R(x) \lor S(x,y) \lor T(y))$
Special Case: No Repeated Symbols

a) What can we do with hierarchical queries?

b) What can we do with non-hierarchical queries?
The Small Dichotomy Theorem

[Dalvi&Suciu’04]

**Theorem** Let $Q$ be one CQ/clause, with no repeated symbols
- If $Q$ is hierarchical, then $P_{D}(Q)$ is in PTIME.
- If $Q$ is not hierarchical then $P_{D}(Q)$ is $\#P$-hard.

Checking “$Q$ is hierarchical” is in $AC^0$ (expression complexity)

[Dalvi&Suciu’12]

**Fact:** Any non-hierarchical $Q$ in UCQ is $\#P$-hard
Proof

Hierarchical $\rightarrow$ PTIME

Case 1:

Q = $\forall x \ldots$

$\forall$-Rule:

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

Non-hierarchical $\rightarrow$ #P-hard

Case 2:

Q = $Q_1 \lor Q_2$

$\lor$-Rule:

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

Reduction from $H_0$:

Q = $\ldots S(x, \ldots) \lor F(x,y,\ldots) \lor J(y,\ldots), \ldots$
Are the Lifted Rules Complete?

You already know:

• Inference rules: PTIME data complexity
• Some queries: \#P-hard data complexity
• Dichotomy for CQs/clauses with no repeated symbols.

Dichotomy Theorem for UCQ / Mon. CNF?
What about this query?

4: **Step 1** *Rewriting of Query*
5: Convert $Q$ to union of CNFs: $Q_{UCNF} = Q_1 \lor \ldots \lor Q_m$

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

$P(Q_W) = \ ?$
I/E and Cancellations

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

\[ P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) \]
\[ - P(Q_1 \land Q_2) - P(Q_2 \land Q_3) - P(Q_1 \land Q_3) \]
\[ + P(Q_1 \land Q_2 \land Q_3) \]
Background: \textbf{\#P-hard Queries $H_k$}

\begin{align*}
H_0 &= R(x) \lor S(x,y) \lor T(y) \\
H_1 &= [R(x_0) \lor S(x_0,y_0)] \land [S(x_1,y_1) \lor T(y_1)] \\
H_2 &= [R(x_0) \lor S_1(x_0,y_0)] \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \lor [S_2(x_2,y_2) \lor T(y_2)] \\
H_3 &= [R(x_0) \lor S_1(x_0,y_0)] \\
&\quad \land [S_1(x_1,y_1) \lor S_2(x_1,y_1)] \\
&\quad \land [S_2(x_2,y_2) \lor S_3(x_2,y_2)] \\
&\quad \land [S_3(x_3,y_3) \lor T(y_3)] \\
\ldots
\end{align*}

\textbf{Theorem.} Every query $H_k$ is \textbf{\#P-hard} \cite{DalviS12}
I/E and Cancellations

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

\[ \text{Also } = H_3 \]

\[ P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) + \]
\[ - P(Q_1 \land Q_2) - P(Q_2 \land Q_3) - P(Q_1 \land Q_3) \]
\[ + P(Q_1 \land Q_2 \land Q_3) \]

\[ = H_3 \text{ (}\#P\text{-hard }) \]

Need to cancel terms to compute the query in \textbf{PTIME}

[Suciu’11]
Cancellations?

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

• (EXTRA: The mathematical concept that explains which terms cancel out is the Mobius function on the implication lattice of the query)
Are the Lifted Rules Complete?

You already know:

- Inference rules: \textsc{PTIME} data complexity
- Some queries: \#P-hard data complexity

\textbf{Dichotomy Theorem} for UCQ / Mon. CNF

- If lifted rules succeed, then \textsc{PTIME} query
- If lifted rules fail, then query is \#P-hard (in the size of the database; data complexity)

Lifted rules are complete for UCQ!

[Dalvi and Suciu; JACM’11]
Side Note: Linear Data Complexity

\[ Q = \exists x \exists y \text{Scientist}(x) \land \text{Coauthor}(x,y) \]

\[ P(Q) = 1 - \prod_{A \in \text{Domain}} (1 - P(\text{Scientist}(A) \land \exists y \text{Coauthor}(A,y))) \]

\[ = 1 - (1 - P(\text{Scientist}(A) \land \exists y \text{Coauthor}(A,y))) \]
\[ \times (1 - P(\text{Scientist}(B) \land \exists y \text{Coauthor}(B,y))) \]
\[ \times (1 - P(\text{Scientist}(C) \land \exists y \text{Coauthor}(C,y))) \]
\[ \times (1 - P(\text{Scientist}(D) \land \exists y \text{Coauthor}(D,y))) \]
\[ \times (1 - P(\text{Scientist}(E) \land \exists y \text{Coauthor}(E,y))) \]
\[ \times (1 - P(\text{Scientist}(F) \land \exists y \text{Coauthor}(F,y))) \]
\[ \ldots \]

\[ \rightarrow \text{No supporting facts in database!} \]
\[ \rightarrow \text{Probability 0} \]
\[ \rightarrow \text{Ignore these sub-queries!} \]

Complexity linear time in database size!

[Ceylan’16]
What about negation?

• Negation not allowed in CQ/UCQ
• Negation common in AI knowledge bases
• Are existing lifted inference rules enough?

Example:

$$\text{Pr}(\forall x \forall y (R(x) \lor S(x, y)) \land (\neg S(x, y) \lor T(y)))$$
Two Questions

• **Question 1:** Are the lifted rules complete?
  – We know that they get stuck on some queries
  – Should we add more rules?

  **Complete for Unions of Conjunctive Queries (UCQ)**

• **Question 2:** Are lifted rules stronger than grounded?
  – Lifted rules can also be grounded
  – Any advantage over grounded inference?

  **Strictly stronger than DPLL-based algorithms (see book)**