On the Optimal Approximation of Queries Using Tractable Propositional Languages

ICDT 2011, Uppsala

Robert Fink and Dan Olteanu

Computing Laboratory, University of Oxford

March 2011

Outline

Outline

Motivation for query approximation

- Approximate query evaluation in probabilistic databases
  $\rightarrow$ Exact query evaluation is #P-hard already for simple queries.

- Approximate explanations of query answers in provenance databases
  $\rightarrow$ Full explanations may have large size.

- Sampling-based approximation for query evaluation in relational databases
  $\rightarrow$ For aggregation queries in very large databases

Outline

Annotated databases

- Tuples are annotated with event („lineage“) expressions
- Here: Annotation with elements of the Boolean semi-ring

R

A	E
1	x_1
2	x_2

S

A	B	E
1	1	$\top$
1	2	$\top$
2	2	$\top$

T

B	E
1	y_1
2	y_2

- Queries map annotated databases to annotated databases. In particular, for every query, one can construct a propositional formula Φ that reflects the computation of the query answer.

$Q(A, B) \leftarrow R(A), S(A, B), T(B)$

A	B	E
1	1	$x_1 y_1$
1	2	$x_1 y_2$
2	2	$x_2 y_2$

$Q \leftarrow R(A), S(A, B), T(B)$

	E
()	$x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$

Sandwich-bounds for event formulas

R

A	E
1	x_1
2	x_2

S

A	B	E
1	1	$\top$
1	2	$\top$
2	2	$\top$

T

B	E
1	y_1
2	y_2

$$Q \leftarrow R(A), S(A, B), T(B)$$

$$\Phi = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$$

- Find formulas Φ_L, Φ_U such that $\Phi_L \models \Phi \models \Phi_U$
- If Φ_L, Φ_U have „nicer“ properties than Φ, then they provide convenient lower and upper bounds for Φ
- For example, bound formulas in which every variable symbol occurs only once: $\Phi_L = x_1(y_1 \vee y_2)$, $\Phi_U = x_1 \vee x_2 y_2$

Application to provenance databases

R

A	E
1	x_1
2	x_2

S

A	B	E
1	1	$\top$
1	2	$\top$
2	2	$\top$

T

B	E
1	y_1
2	y_2

$$Q \leftarrow R(A), S(A, B), T(B)$$

$$\Phi = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$$

$$x_1(y_1 \vee y_2) \models x_1 y_1 \vee x_1 y_2 \vee x_2 y_2 \models x_1 \vee x_2 y_2$$

- Lower bounds represent correct, yet not necessarily complete explanations
- Upper bounds represent complete, yet not necessarily correct explanations
- Idea: Choose bound formulas that admit small representation

Application to probabilistic databases

R

A	E
1	x_1
2	x_2

S

A	B	E
1	1	$\top$
1	2	$\top$
2	2	$\top$

T

B	E
1	y_1
2	y_2

$$Q \leftarrow R(A), S(A, B), T(B)$$

$$\Phi = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$$

- Possible world semantics (database instances D, interpretations I):

$$P(Q) \stackrel{def}{=} \sum_{D:Q(D) \text{ is true}} P(D) = \sum_{I:I \models \Phi} P(I) \stackrel{def}{=} P(\Phi)$$

- Probability computation for general propositional formulas is #P-hard
- Model bounds imply probability bounds:

$$\Phi_L \models \Phi \models \Phi_U \qquad \Rightarrow \qquad P(\Phi_L) \leq P(\Phi) \leq P(\Phi_U)$$

- Idea: Choose bound formulas from a language that admits efficient probability computation

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
2. How to define optimality of bounds?
3. How to compute optimal bounds efficiently?

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas or their DNF restrictions have size linear in the number of variables (and hence the size of the database) and admit linear time probability computation.
   - The event of every tractable conjunctive query without self-joins is equivalent to a read-once formula that can be computed in polynomial time.
   - More expressive languages? It is NP-hard to decide whether a formula has an equivalent read-2 formula. For read-3 formulas, probability computation is #P-hard.
2. How to define optimality of bounds?
3. How to compute optimal bounds efficiently?

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas
2. How to define optimality of bounds?
3. How to compute optimal bounds efficiently?

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas
2. How to define optimality of bounds?
   - Let $\mathcal{L}'$ and $\mathcal{L}$ be two languages of propositional formulas and $\Phi \in \mathcal{L}$. Formula $\Phi_L \in \mathcal{L}'$ is a *lower bound for* Φ *with respect to* $\mathcal{L}'$, if

$$\Phi_L \models \Phi \qquad \text{(i.e. } \mathcal{M}(\Phi_L) \subseteq \mathcal{M}(\Phi)\text{)}.$$

     If in addition there is no formula $\Phi'_L \in \mathcal{L}'$ such that

$$\mathcal{M}(\Phi_L) \subset \mathcal{M}(\Phi'_L) \subseteq \mathcal{M}(\Phi)$$

     then Φ_L is a *greatest lower bound for* Φ *with respect to* $\mathcal{L}'$. Least upper bounds are defined analogously.
3. How to compute optimal bounds efficiently?

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas
2. How to define optimality of bounds?
   - Greatest lower bounds and least upper bounds w.r.t. a language
3. How to compute optimal bounds efficiently?

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas
2. How to define optimality of bounds?
   - Greatest lower bounds and least upper bounds w.r.t. a language
3. How to compute optimal bounds efficiently?
   - Semantic definition is not very useful
   - Seek equivalent syntactic definitions of optimal bounds
   - Find algorithms to compute those bounds

Key challenges for model-based query approximation

1. Which languages of propositional formulas are useful?
   - Read-once formulas
2. How to define optimality of bounds?
   - Greatest lower bounds and least upper bounds w.r.t. a language
3. How to compute optimal bounds efficiently?
   - Seek equivalent syntactic characterisation of optimal bounds

Outline

Syntactic characterisation of optimal iDNF lower bounds

- iDNF = class of read-once DNF formulas
- Consider monotone/unate input formulas, since non-trivial approximation of general formulas is NP-hard
- Starting point: Generic characterisation of lower bounds:
  Φ_L is a lower bound of Φ if and only if Φ_L is obtainable by removing clauses from Φ or adding literals to its clauses.
- Example: $\Phi = x_1y_1 \vee x_1y_2 \vee x_2y_2$
  Lower bounds: x_1y_1, $x_1y_1 \vee x_2y_2$, $x_1y_1y_2$, ...

- Syntactic characterisation of optimal lower iDNF bounds:
  1. (*Lower bound*) Φ_L contains a subset of the clauses of Φ
  2. (*Maximality*) No further clause from Φ can be added to Φ_L

Syntactic characterisation of optimal iDNF lower bounds

- iDNF = class of read-once DNF formulas
- Consider monotone/unate input formulas, since non-trivial approximation of general formulas is NP-hard
- Starting point: Generic characterisation of lower bounds:
  Φ_L is a lower bound of Φ if and only if Φ_L is obtainable by removing clauses from Φ or adding literals to its clauses.
- Example: $\Phi = x_1y_1 \vee x_1y_2 \vee x_2y_2$
  Lower bounds: x_1y_1, $x_1y_1 \vee x_2y_2$, $x_1y_1y_2$, ...
  Optimal iDNF lower bounds: x_1y_2, $x_1y_1 \vee x_2y_2$
  Non-iDNF lower bounds: $x_1y_1 \vee x_1y_2$, ...
  Non-optimal iDNF lower bounds: x_1y_1, x_2y_2, ...
- Syntactic characterisation of optimal lower iDNF bounds:
  1. (*Lower bound*) Φ_L contains a subset of the clauses of Φ
  2. (*Maximality*) No further clause from Φ can be added to Φ_L

Syntactic characterisation of optimal iDNF lower bounds

- Theorem: The semantic and syntactic characterisations of optimal iDNF lower bounds are equivalent.
- How many optimal lower bounds exist for a given formula? Exponentially many!

$$\Phi = (x_1 y_1 \vee x_1 y_2) \vee \cdots \vee (x_n y_{2n-1} \vee x_n y_{2n})$$

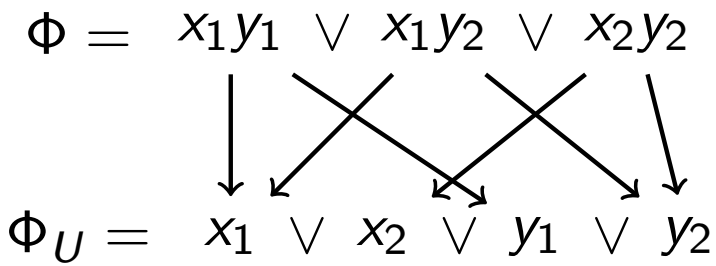
  has $3n$ variables, $2n$ clauses and 2^n iDNF greatest lower bounds.
- Polynomial enumeration of all optimal lower bounds is thus not possible. Next best thing: **Polynomial delay**
- Optimal lower bounds correspond to maximal independent sets in the clause dependency graph of the input formula
- There exist algorithms for polynomial-delay enumeration of maximal independet sets (e.g. Johnson et al, 1998)

Syntactic characterisation of optimal iDNF upper bounds

- Starting point: Generic characterisation of upper bounds:
  Φ_U is an upper bound of Φ if and only if Φ_U is obtainable by adding clauses to Φ or removing literals from its clauses.
- Idea for syntactic and algorithmic treatment: Start with the most general upper bound $x_1 \vee \cdots \vee x_n$ and refine it until it gets optimal.

Syntactic characterisation of optimal iDNF upper bounds

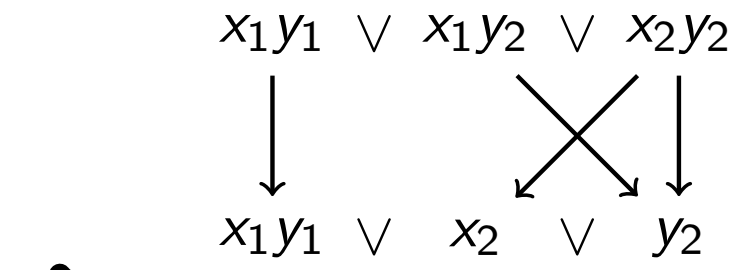
Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?

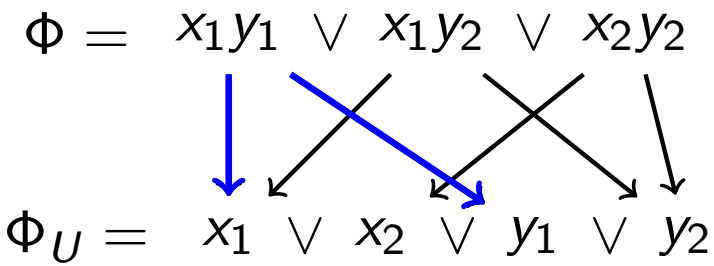


Syntactic characterisation of optimal iDNF upper bounds

Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?

x_1y_1 implies both x_1 and y_1 which can be merged.

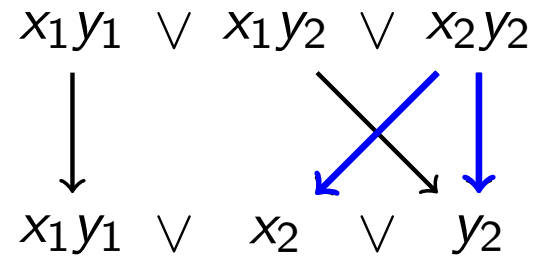


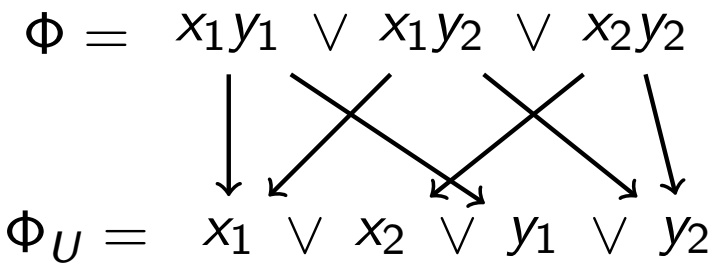


Syntactic characterisation of optimal iDNF upper bounds

Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?

x_2 is not necessary and can be removed.

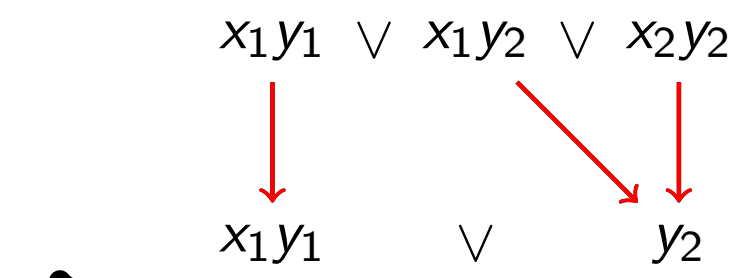


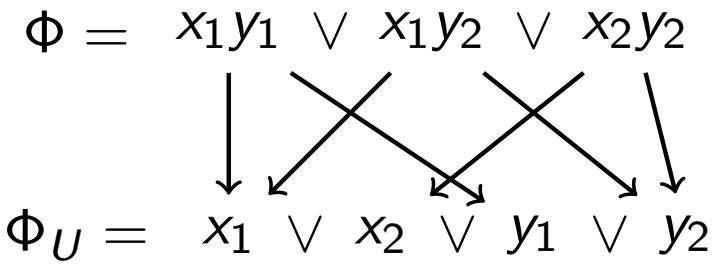


Syntactic characterisation of optimal iDNF upper bounds

Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?

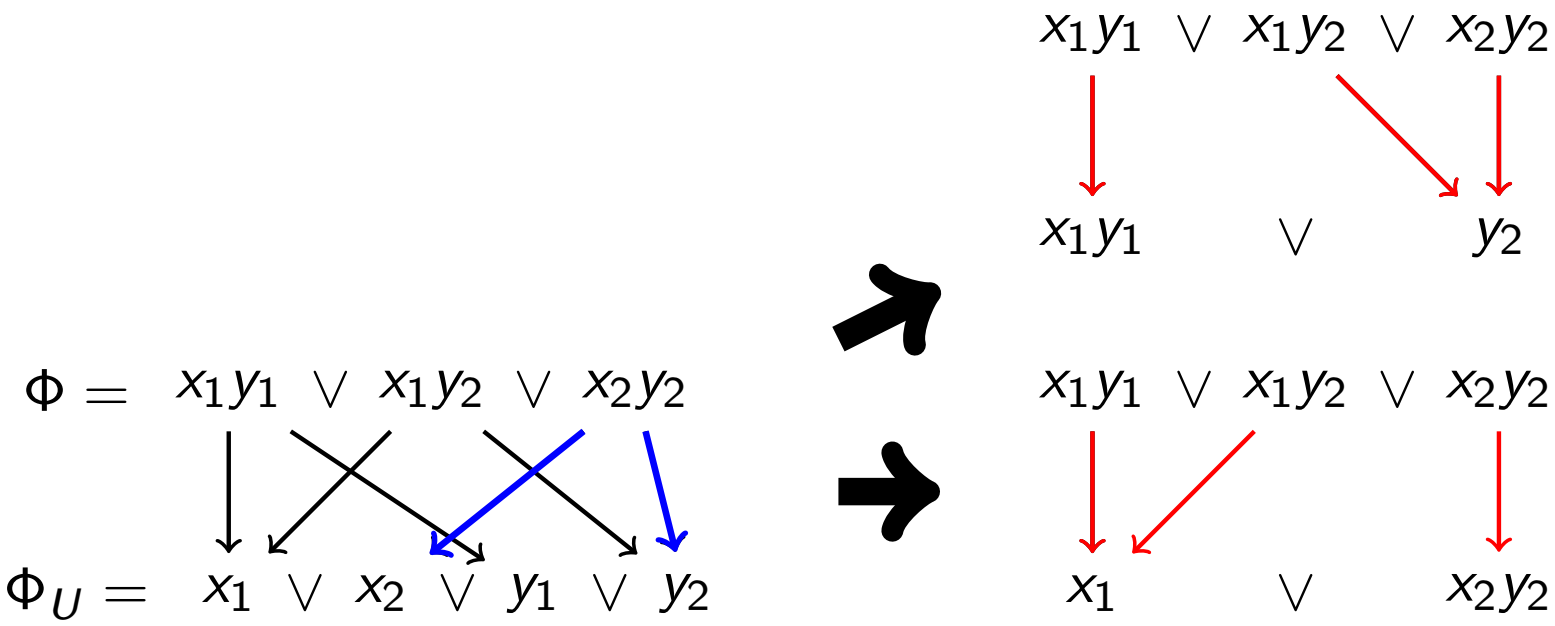
No non-necessary clauses. No clause can be extended by x_2.





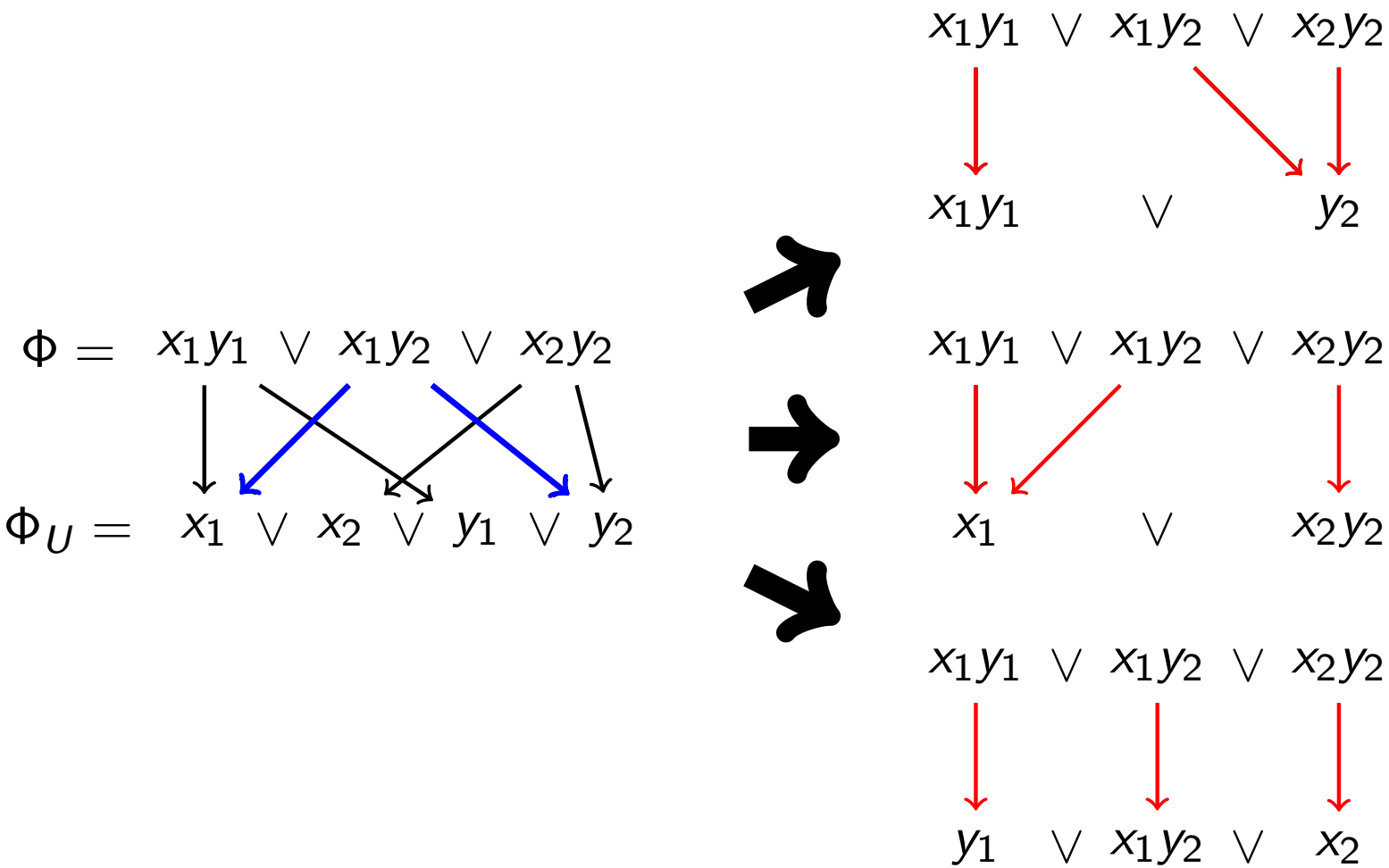
Syntactic characterisation of optimal iDNF upper bounds

Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?



Syntactic characterisation of optimal iDNF upper bounds

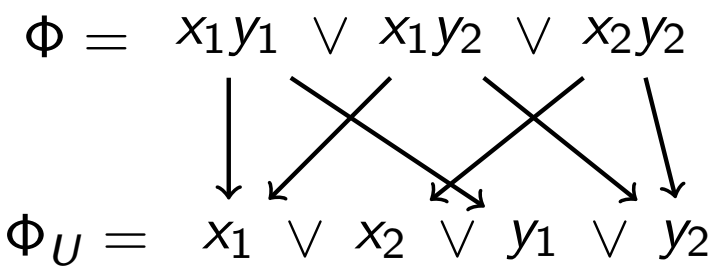
Example: How to find upper bounds for $x_1y_1 \vee x_1y_2 \vee x_2y_2$?

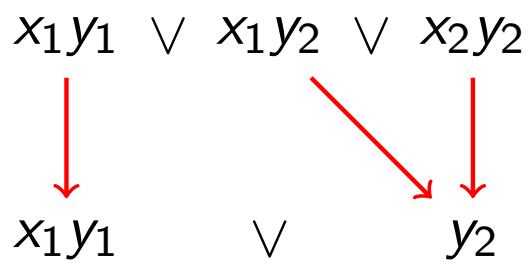


Syntactic characterisation of optimal iDNF upper bounds

Ingredients to syntactic definition of optimal upper bounds:

- Every clause in Φ implies a clause in Φ_U
- Every clause in Φ_U must be implied by one clause in Φ *exclusively*
- No unnecessary clauses in Φ_U
- No clause in Φ_U can be extended by a variable from Φ while preserving the above conditions





Syntactic characterisation of optimal iDNF upper bounds

- Theorem: The semantic and syntactic characterisations of optimal iDNF upper bounds are equivalent.
- How many optimal upper bounds exist for a given formula? Exponentially many!

$$\Phi = (x_1 y_1 \vee x_1 y_2) \vee \cdots \vee (x_n y_{2n-1} \vee x_n y_{2n})$$

  has $3n$ variables, $2n$ clauses and 3^n iDNF greatest upper bounds.
- Polynomial enumeration of all optimal upper bounds is thus not possible. Next best thing: **Polynomial delay**
- We present two algorithms in the paper:
  1. Enumeration of all optimal iDNF upper bounds.
  2. Enumeration with polynomial delay of all optimal iDNF upper bounds that preserve the variables of the input formula.

Outline

Optimal bounds with respect to arbitrary read-once formulas

- So far: iDNF bounds
- Next best: Read-once bounds (that is, without the restriction to DNF formulas)
- We succeeded at finding optimal read-once k-partite bounds for k-partite formulas
- Those bounds are also optimal w.r.t. general read-once formulas.

Optimal bounds with respect to arbitrary read-once formulas

- Query Q:-$R(A), S(A,B), T(B)$ with event formula

  $\Phi = x_1y_1z_1 \vee x_1y_2z_2 \vee x_2y_3z_1 \vee x_2y_4z_2$ is no read-once formula

- Find k-partite upper bounds by adding clauses to Φ such that it factorises. There may be several choices for this expansion:

$$\begin{aligned} \Phi_{U,1} = &\quad (x_1 \vee x_2)[z_1(y_1 \vee y_3) \vee z_2(y_2 \vee y_4)] \\ \Phi_{U,2} = &\quad [x_1(y_1 \vee y_2) \vee x_2(y_3 \vee y_4)](z_1 \vee z_2) \end{aligned}$$

- Find k-partite lower bounds by removing clauses from Φ such that it factorises.

$$\begin{aligned} \Phi_{L,1} = &\quad (x_1)[y_1z_1 \vee y_2z_2)] \\ \Phi_{L,2} = &\quad (x_2)[y_3z_1 \vee y_4z_2)] \\ &\quad \ldots \end{aligned}$$

Optimal bounds with respect to arbitrary read-once formulas

- We give an algorithm to enumerate *some* optimal read-once upper bounds with polynomial delay. The problem of enumerating all optimal read-once upper bounds with polynomial delay is still open.
- We give an algorithm to compute all optimal read-once lower bounds. The problem of enumeration with polynomial delay is still open.

Approximation by queries

- Idea: Rewrite a given (hard) query Q into bound queries Q_L and Q_U such that their event formulas are read-once bounds for the event of Q
- Catch 1: Expressing the query for upper bounds requires a query language that is able to express transitive closure
- Catch 2: Removing edges to get lower bounds requires non-deterministic choice, or a linear order on tuples
- There are different upper and lower bounds for a given formula. These choices correspond to different rewritings of Q.

Conclusion

- Framework for model-based characterisation of optimal bounds for propositional formulas
- Applications: Probabilistic databases, provenance databases
- Syntactic characterisations that are equivalent to model-based definitions yet much easier to turn into algorithms

Open questions

- The read-once results are so far only for k-partite formulas which is great for conjunctive queries without self-joins. What happens beyond k-partite approximations?
- Bounds for non-DNF input formulas?

End. ?