Providing Support for Full Relational Algebra in Probabilistic Databases

ICDE 2011, Hannover

Robert Fink and Dan Olteanu and Swaroop Rath

Computing Laboratory, University of Oxford

April 2011

Goal of this work

To propose evaluation techniques for relational algebra queries on complete representation systems for probabilistic data

- We consider a formalism that can represent any discrete probability space over any finite set of possible worlds
- Both exact and approximate techniques are based on incremental compilation of the query lineage
- Exact evaluation requires exhaustive compilation
- For approximate evaluation, the evaluation can stop depending on the time budget or when the desired approximation is reached

Further contributions

- Implementation of the evaluation techniques within the probabilistic database system SPROUT (based on PostgreSQL)
- Complexity analysis for „quantified queries“ such as relational division

Motivating example: Queries with universal quantification

Business scenario: List suppliers that supply all products

- Relations $S(\mathit{SupplierId}, \mathit{ProductId})$, $P(\mathit{ProductId})$
- Query: Find suppliers that supply all products: $S \div P$
- In probabilistic setting: What is the *likelihood* that a given supplier supplies all products?

Healthcare scenario: Match of antigen types of recipients and donors

- Database with recipients, donors, and their antigen types
- Query: Find pairs $(\mathit{rec}, \mathit{don})$ with matching antigen types
- In probabilistic setting: Find $(\mathit{rec}, \mathit{don})$ pairs ranked by the probability that they have matching antigen types.

Outline

Query evaluation in probabilistic databases

Exact evaluation of relational algebra queries

Approximate evaluation of relational algebra queries

Outline

Query evaluation in probabilistic databases

Exact evaluation of relational algebra queries

Approximate evaluation of relational algebra queries

Query evaluation in probabilistic databases

- Given:
  1. Finite set of regular databases, $\mathcal{D} = \{D_1, \cdots\}$,
  2. Probability distribution $P(\cdot)$ over $\mathcal{D}$
  3. Query Q
- Query evaluation problem for probabilistic databases: Compute answer tuples on databases in $\mathcal{D}$ together with *probability that they are true in a randomly chosen instance from* $\mathcal{D}$

Annotated databases as representation for probabilistic databases

- Tuples are annotated with propositional formulas („event“) over a set of Boolean random variables
- Every valuation of the Boolean variables defines a possible world
- Example:

Customer

cKey	cName	E
1	Joe	$x_1 x_3$
2	Dan	$\bar{x}_1 x_4$
3	Li	$x_2 \bar{x}_4$
4	Mo	$\bar{x}_2 x_5$

- Possible world for valuation $[x_1, \bar{x}_2, x_3, x_4, x_5]$: $\{(1, Joe), (4, Mo)\}$
- Events can represent arbitrary correlations, for example:
  - $(1, Joe)$ and $(3, Li)$ are independent since they use different variables. They can occur in the same world.
  - $(1, Joe)$ and $(2, Dan)$ are mutually exclusive. They never occur together in the same world.

Computation of event expression in case of relational division

- Queries map annotated databases to annotated databases. In particular, for every query, one can construct the event Φ that reflects the computation of the query answer.
- We consider full relational algebra queries. We illustrate the construction with the example of relational division.
- $Q(\text{sid}) = S(\text{sid}, \text{pid}) \div P(\text{pid})$
- „Suppliers that supply all products“
- Relational algebra: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$
- Note that the following example considers a tuple-independent database without correlation to keep the formulas handy.

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

P

pid	E
1	y_1
2	y_2
3	y_3

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

$\pi_{\text{sid}}(S) \times P$

sid	pid	E
1	1	$(x_1 + x_2 + x_3 + x_4)y_1$
1	2	$(x_1 + x_2 + x_3 + x_4)y_2$
1	3	$(x_1 + x_2 + x_3 + x_4)y_3$
2	1	$(x_5 + x_6 + x_7)y_1$
2	2	$(x_5 + x_6 + x_7)y_2$
2	3	$(x_5 + x_6 + x_7)y_3$

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

$\pi_{\text{sid}}(S) \times P - S$

sid	pid	E
1	1	$(x_1 + x_2 + x_3 + x_4)y_1\neg x_1$
1	2	$(x_1 + x_2 + x_3 + x_4)y_2\neg x_2$
1	3	$(x_1 + x_2 + x_3 + x_4)y_3\neg x_3$
2	1	$(x_5 + x_6 + x_7)y_1\neg x_5$
2	2	$(x_5 + x_6 + x_7)y_2\neg x_6$
2	3	$(x_5 + x_6 + x_7)y_3$

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

$\pi_{\text{sid}}(S) \times P - S$

sid	pid	E
1	1	$(x_1 + x_2 + x_3 + x_4)y_1\neg x_1$
1	2	$(x_1 + x_2 + x_3 + x_4)y_2\neg x_2$
1	3	$(x_1 + x_2 + x_3 + x_4)y_3\neg x_3$
2	1	$(x_5 + x_6 + x_7)y_1\neg x_5$
2	2	$(x_5 + x_6 + x_7)y_2\neg x_6$
2	3	$(x_5 + x_6 + x_7)y_3$

$\pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

sid	E
1	$(x_1 + x_2 + x_3 + x_4)(y_1\neg x_1 + y_2\neg x_2 + y_3\neg x_3)$
2	$(x_5 + x_6 + x_7)(y_1\neg x_5 + y_2\neg x_6 + y_3)$

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

$\pi_{\text{sid}}(S) \times P - S$

sid	pid	E
1	1	$(x_1 + x_2 + x_3 + x_4)y_1\neg x_1$
1	2	$(x_1 + x_2 + x_3 + x_4)y_2\neg x_2$
1	3	$(x_1 + x_2 + x_3 + x_4)y_3\neg x_3$
2	1	$(x_5 + x_6 + x_7)y_1\neg x_5$
2	2	$(x_5 + x_6 + x_7)y_2\neg x_6$
2	3	$(x_5 + x_6 + x_7)y_3$

$\pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

sid	E
1	$(x_1 + x_2 + x_3 + x_4)(y_1\neg x_1 + y_2\neg x_2 + y_3\neg x_3)$
2	$(x_5 + x_6 + x_7)(y_1\neg x_5 + y_2\neg x_6 + y_3)$

$S \div P : \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

sid	E
1	$(x_1 + x_2 + x_3 + x_4)\neg[(x_1 + x_2 + x_3 + x_4)(y_1\neg x_1 + y_2\neg x_2 + y_3\neg x_3)]$
2	$(x_5 + x_6 + x_7)\neg[(x_5 + x_6 + x_7)(y_1\neg x_5 + y_2\neg x_6 + y_3)]$

Relational division: $S \div P = \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

S

sid	pid	E
1	1	x_1
1	2	x_2
1	3	x_3
1	7	x_4
2	1	x_5
2	2	x_6
2	5	x_7

P

pid	E
1	y_1
2	y_2
3	y_3

$\pi_{\text{sid}}(S)$

sid	E
1	$x_1 + x_2 + x_3 + x_4$
2	$x_5 + x_6 + x_7$

$\pi_{\text{sid}}(S) \times P - S$

sid	pid	E
1	1	$(x_1 + x_2 + x_3 + x_4)y_1\neg x_1$
1	2	$(x_1 + x_2 + x_3 + x_4)y_2\neg x_2$
1	3	$(x_1 + x_2 + x_3 + x_4)y_3\neg x_3$
2	1	$(x_5 + x_6 + x_7)y_1\neg x_5$
2	2	$(x_5 + x_6 + x_7)y_2\neg x_6$
2	3	$(x_5 + x_6 + x_7)y_3$

$\pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

sid	E
1	$(x_1 + x_2 + x_3 + x_4)(y_1\neg x_1 + y_2\neg x_2 + y_3\neg x_3)$
2	$(x_5 + x_6 + x_7)(y_1\neg x_5 + y_2\neg x_6 + y_3)$

$S \div P : \pi_{\text{sid}}(S) - \pi_{\text{sid}}\left[\pi_{\text{sid}}(S) \times P - S\right]$

sid	E
1	$(x_1 + x_2 + x_3 + x_4)(y_1 \to x_1)(y_2 \to x_2)(y_3 \to x_3)$
2	$(x_5 + x_6 + x_7)(y_1 \to x_5)(y_2 \to x_6)(\neg y_3)$

Connection between query probability and events

- Event encodes in which worlds an answer tuple is true
- Furthermore: Probability of an answer tuple is equivalent to probability of the event of this answer tuple.

- Probability computation is #P-hard for general propositional formulas and already simple conjunctive queries
- Our goal: Support expressive queries involving negation, universal quantification, relational division, etc.

- Hence, we seek seek a method that can provide efficient exact evaluation where possible, and default to approximations where necessary

Approximation techniques for query evaluation

Existing approaches

1. Modify query and database (e.g. **?**)
2. Expand event to DNF and use sampling-based approach to obtain FPRAS (e.g. **??**)
3. Knowledge compilation techniques (**?**)

Queries with negation and approaches 1,2

- Approach 1 has only been investigated for conjunctive queries and tuple-independent databases
- Arbitrary events may not directly permit an FPRAS
- Expansion to DNF can yield exponential blowup *in the size of* $|D|$

$\Rightarrow$ **We extend approach 3 to arbitrary relational algebra queries**

Outline

Query evaluation in probabilistic databases

Exact evaluation of relational algebra queries

Approximate evaluation of relational algebra queries

Decomposition trees for propositional formulas

Basic idea: First compute event Φ for a query. Then incrementally decompose Φ using using knowledge compilation techniques.

Decomposition trees for propositional formulas

Basic idea: First compute event Φ for a query. Then incrementally decompose Φ using using knowledge compilation techniques.

Decomposition rules

1. If $\Phi = \Phi_1 \wedge \Phi_2$ and Φ_1, Φ_2 are independent:

$$P(\Phi) = P(\Phi_1) \cdot P(\Phi_2)$$

2. If $\Phi = \Phi_1 \vee \Phi_2$ and Φ_1, Φ_2 are independent:

$$P(\Phi) = 1 - (1 - P(\Phi_1)) \cdot (1 - P(\Phi_2))$$

3. Shannon-expansion based on mutually exclusive sub-formulas:

$$P(\Phi) = P(x) \cdot P(\Phi|_x) + P(\neg x) \cdot P(\Phi|_{\neg x})$$

Example: Decomposition of relational division event

$$(x_5 + x_6 + x_7)(y_1 \to x_5)(y_2 \to x_6)(\neg y_3)$$

Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

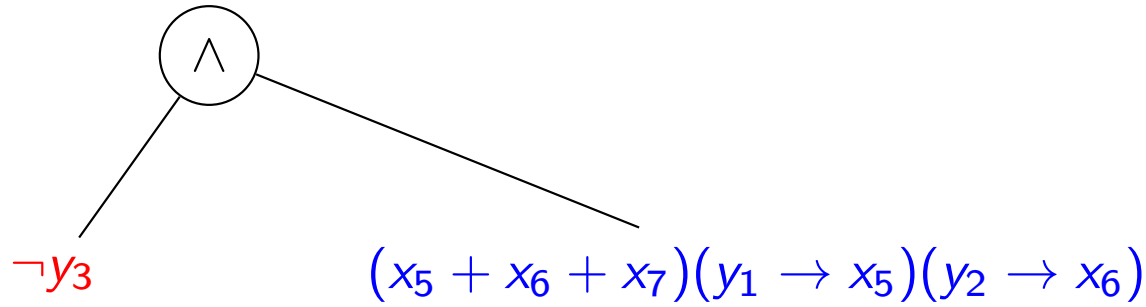
Example: Decomposition of relational division event

$$(x_5 + x_6 + x_7)(y_1 \to x_5)(y_2 \to x_6)(\neg y_3)$$

Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

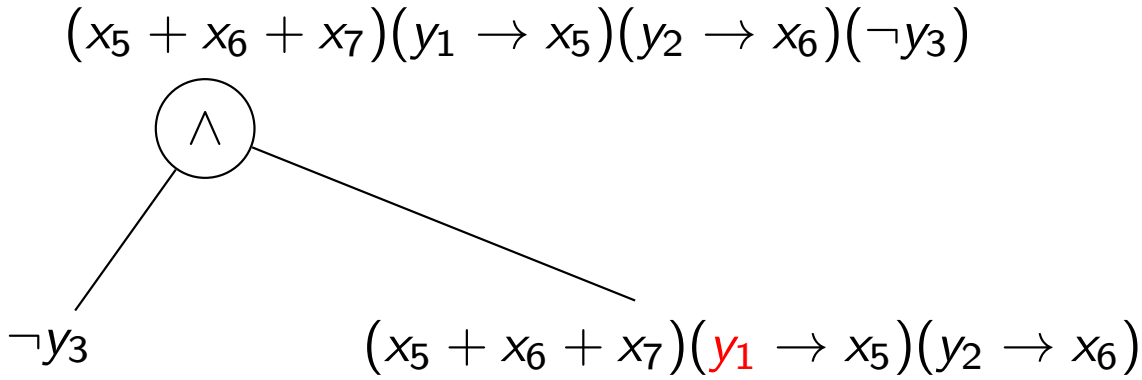
Example: Decomposition of relational division event

$$(x_5 + x_6 + x_7)(y_1 \to x_5)(y_2 \to x_6)(\neg y_3)$$



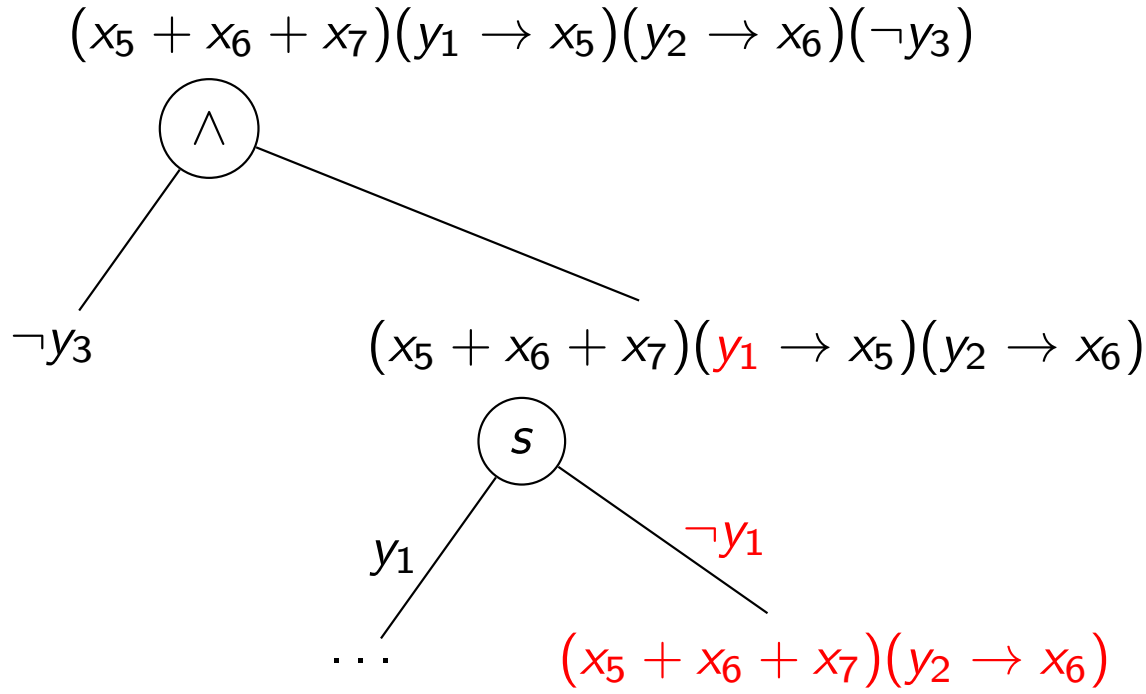
Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

Example: Decomposition of relational division event



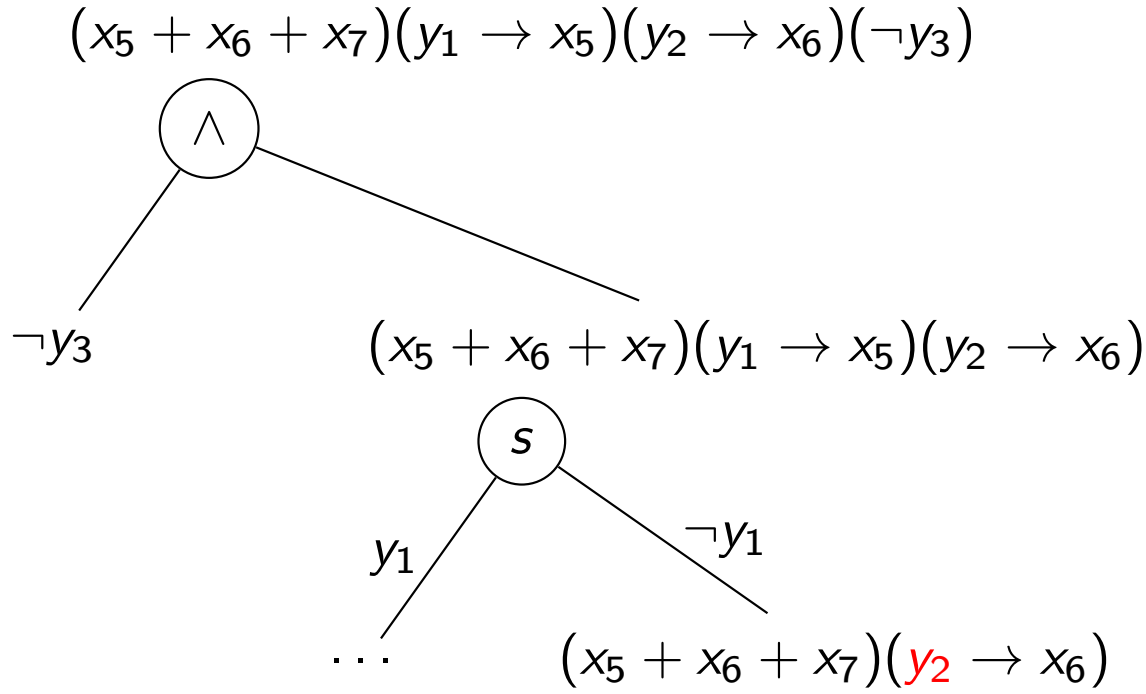
Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

Example: Decomposition of relational division event



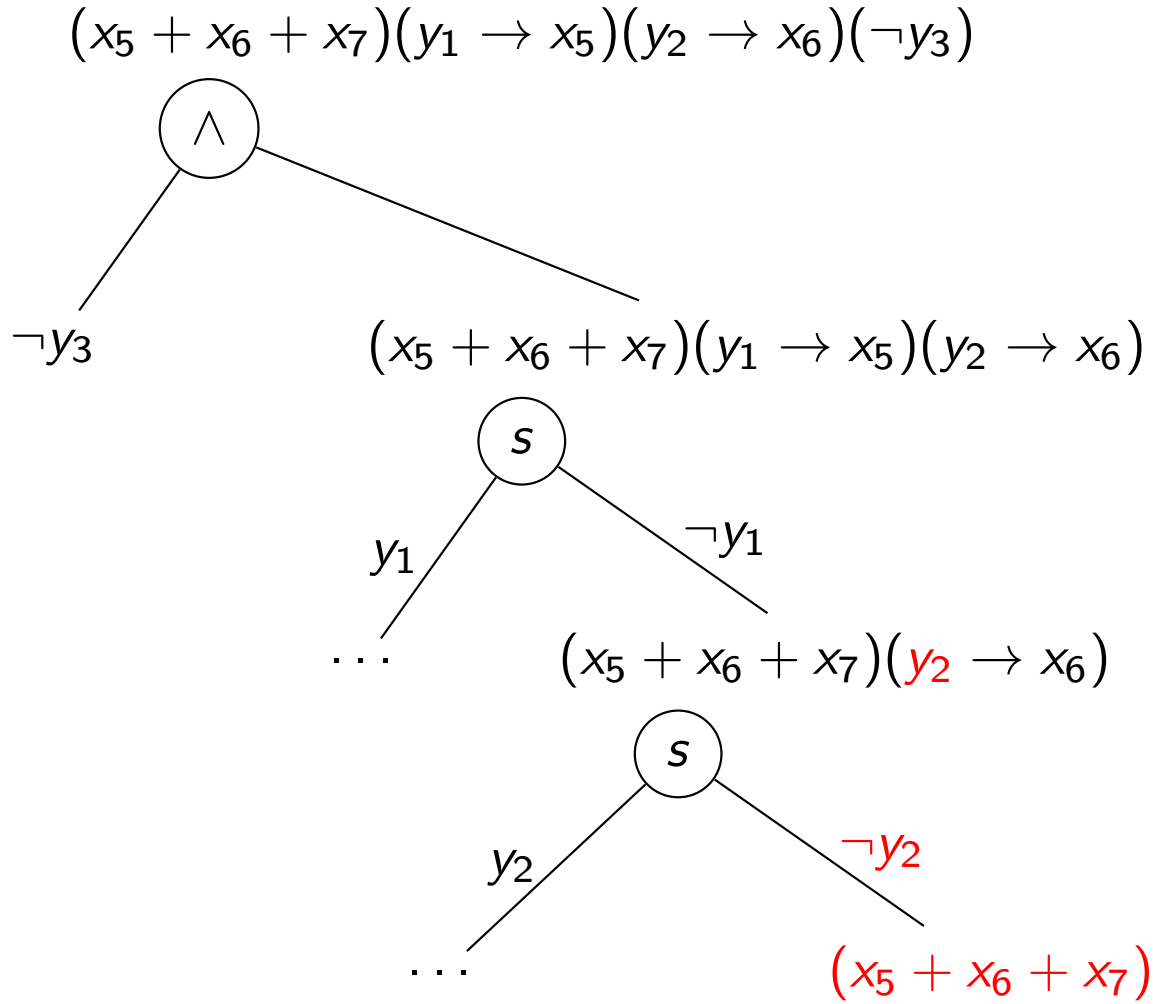
Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

Example: Decomposition of relational division event



Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

Example: Decomposition of relational division event



Algorithmic idea: Recursively decompose the formula until every leaf is atomic. Then compute the formula's probability while traversing back up.

Outline

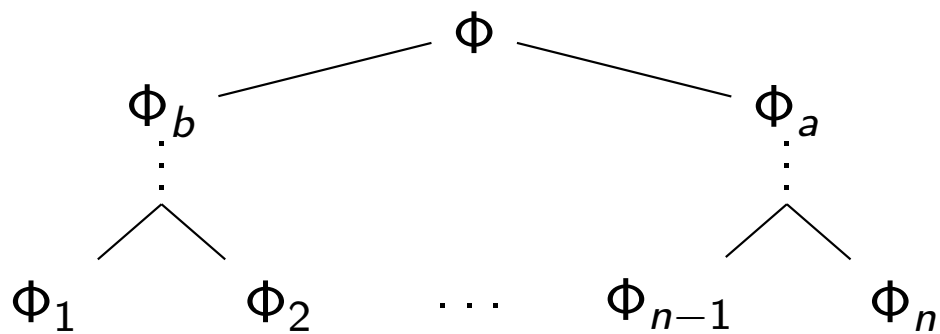
Query evaluation in probabilistic databases

Exact evaluation of relational algebra queries

Approximate evaluation of relational algebra queries

Approximate probability computation using d-trees

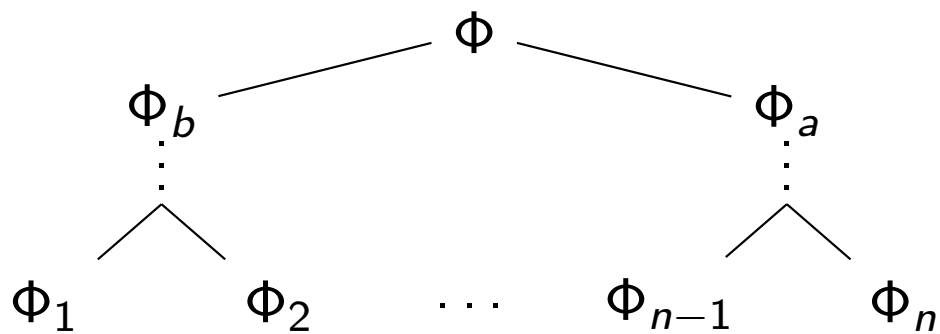
- Exact approximation requires fully expanded decision tree
- Idea for approximate evaluation: Don't decompose a formula fully, but stop when sufficient approximation is reached



- Leafs Φ_i are still non-atomic formulas

Approximate probability computation using d-trees

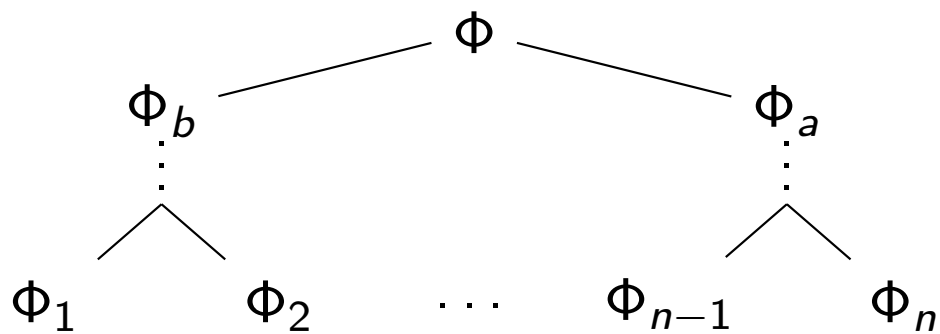
- Exact approximation requires fully expanded decision tree
- Idea for approximate evaluation: Don't decompose a formula fully, but stop when sufficient approximation is reached



- Leafs Φ_i are still non-atomic formulas
- For each leaf formula Φ_i, compute bounds $P_i^L \leq P(\Phi_i) \leq P_i^U$

Approximate probability computation using d-trees

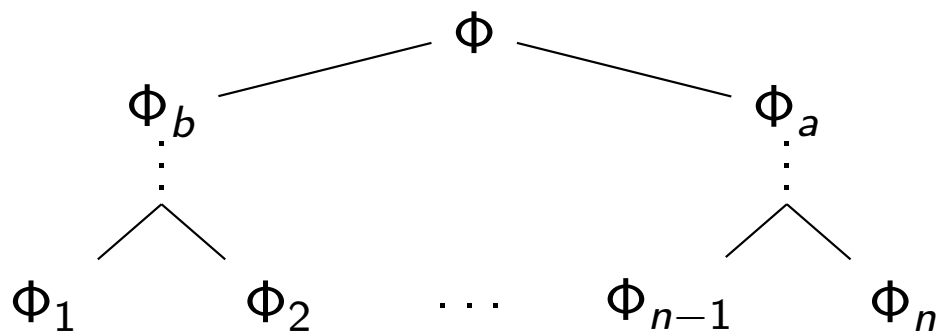
- Exact approximation requires fully expanded decision tree
- Idea for approximate evaluation: Don't decompose a formula fully, but stop when sufficient approximation is reached



- Leafs Φ_i are still non-atomic formulas
- For each leaf formula Φ_i, compute bounds $P_i^L \leq P(\Phi_i) \leq P_i^U$
- Propagate those bounds up to obtain $P_a^L \leq P(\Phi_a) \leq P_a^U$, $P_b^L \leq P(\Phi_b) \leq P_b^U$, **and eventually** $P^L \leq P(\Phi) \leq P^U$

Approximate probability computation using d-trees

- Exact approximation requires fully expanded decision tree
- Idea for approximate evaluation: Don't decompose a formula fully, but stop when sufficient approximation is reached



- Leafs Φ_i are still non-atomic formulas
- For each leaf formula Φ_i, compute bounds $P_i^L \leq P(\Phi_i) \leq P_i^U$
- Propagate those bounds up to obtain $P_a^L \leq P(\Phi_a) \leq P_a^U$, $P_b^L \leq P(\Phi_b) \leq P_b^U$, **and eventually** $P^L \leq P(\Phi) \leq P^U$
- Example: $\Phi = \Phi_1 \wedge \Phi_2$.

$$P_1^L \cdot P_2^L \leq P(\Phi) \leq P_1^U \cdot P_2^U$$

Computation of bounds for leaf formulas

- Construct model-based bounds in *read-once form* by:
  - Dropping clauses (lower bounds)
  - Dropping variables (upper bounds)
- Example: Leaf formula $\Phi = x_1 y_1 \vee x_1 y_2 \vee x_2 y_2$
  Lower bounds: $x_1 y_2$ or $x_1 y_1 \vee x_2 y_2$
  Upper bounds: $x_1 y_1 \vee y_2$ or $x_1 \vee x_1 y_2 \vee x_2$
- Details on optimal model-based bounds: **?**

Relative and absolute approximations with d-trees

- Absolute approximation: Given ϵ, find $\hat{P}$ such that

$$P - \epsilon \leq \hat{P} \leq P + \epsilon$$

- Relative approximation: Given ϵ, find $\hat{P}$ such that

$$(1 - \epsilon)P \leq \hat{P} \leq (1 + \epsilon)P$$

- Given Φ and lower and upper bounds P^L, P^U obtained from a partial decomposition of Φ, we have:
  - If $P^U - \epsilon \leq P^L + \epsilon$, then any value $\hat{P}$ with $P^U - \epsilon \leq \hat{P} \leq P^L + \epsilon$ is an absolute ϵ-approximation of $P(\Phi)$
  - If $(1 - \epsilon) \cdot P^U \leq (1 + \epsilon) \cdot P^L$, then any value $\hat{P}$ with $(1 - \epsilon) \cdot P^U \leq \hat{P} \leq (1 + \epsilon) \cdot P^L$ is a relative ϵ-approximation of $P(\Phi)$

In what order should one expand the decomposition tree to obtain a memory efficient algorithm?

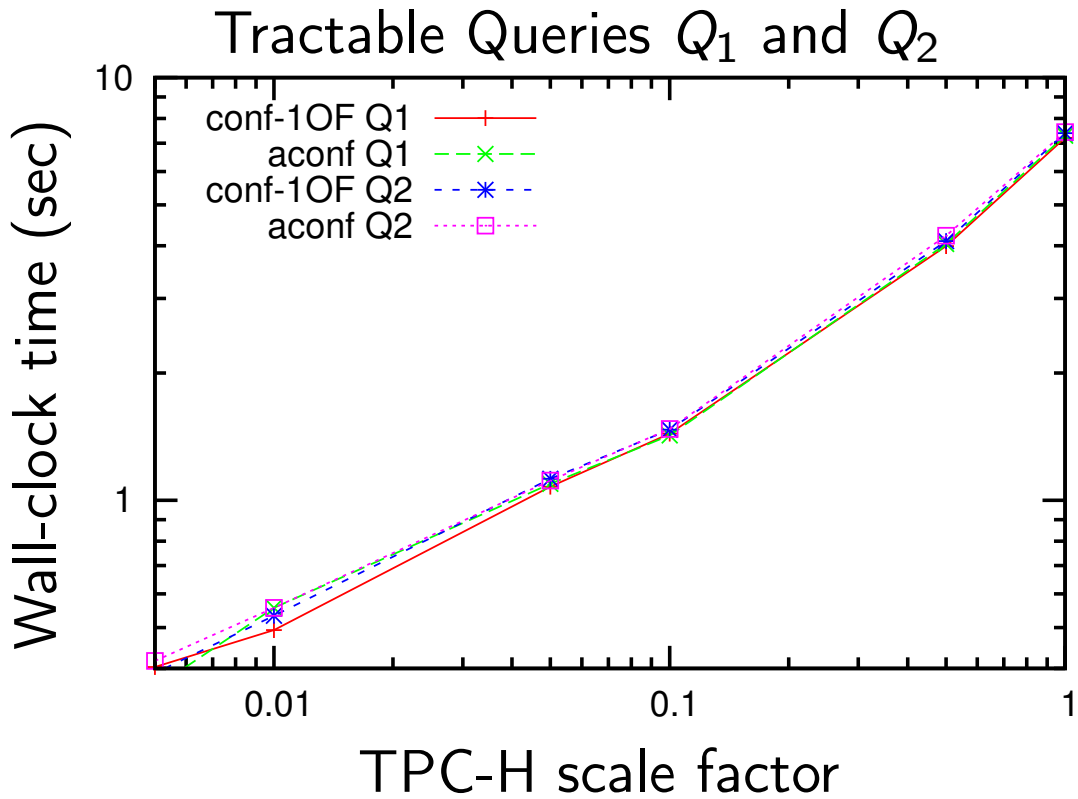
- Depth-first expansion may run into branches that only improve probability bounds marginally
- Breadth-first expansion is not memory-efficient
- There exists a stopping criterion: At every node, we can efficiently decide whether we need to expand it further in order to reach absolute or relative approximation guarantees, or whether we can safely move to and expand the next node and still reach the approximation guarantee. (**?**)

End. ?

References

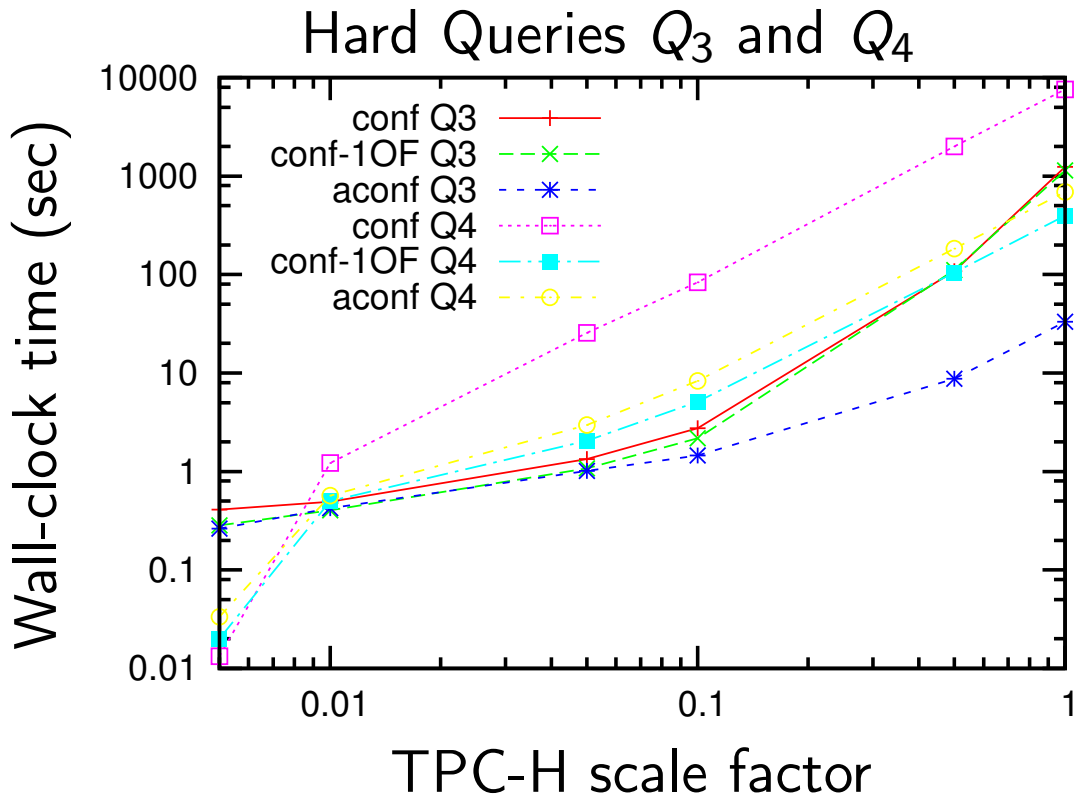
Experimental evaluation using TPC-H scenarios

- We consider two tractable and two #P-hard TPC-H queries
- The TPC-H datasets are extended with event annotations and probabilities



Experimental evaluation using TPC-H scenarios

- We consider two tractable and two #P-hard TPC-H queries
- The TPC-H datasets are extended with event annotations and probabilities



Relational division and set inclusion queries

Consider the following queries

- Relational division as introduced before
- Set inclusion query: Consider a relation $Set(setId, itemId)$ that encodes which items belong to which set.
- Query: List sets that are contained in other sets
- Example: $S(\text{sid}, \text{pid})$ containing suppliers and their products. „Which suppliers supply only items that are also provided by another supplier?“ I.e., list redundant suppliers.
- Furthermore: strict set inclusion, set equality, set incomparability

Data complexity of relational division and set inclusion queries

- The afore-mentioned relational division and set inclusion queries are tractable (PTIME data complexity) if all attributes are head attributes, i.e. queries of the form: „List all pairs of sets identifiers (s_1, s_2) such that set s_1 is included in set s_2.“
- The queries are intractable (#P-hard) if any attribute is projected out. For example:
  - „List all sets that are included in another set.“
  - „Is there a set that is included in another set.“