UNIVERSITY OF MARYLAND

LOCAL STRUCTURE AND DETERMINISM IN PROBABILISTIC DATABASES

Theodoros Rekatsinas, Amol Deshpande, Lise Getoor

Motivation

- Probabilistic databases store, manage and query uncertain data

- Numerous applications generate **correlated** data (e.g., sensor networks, data integration etc.)

Problem

Scaling query evaluation in probabilistic databases with correlated data

Contributions in a nutshell

Challen

1. How

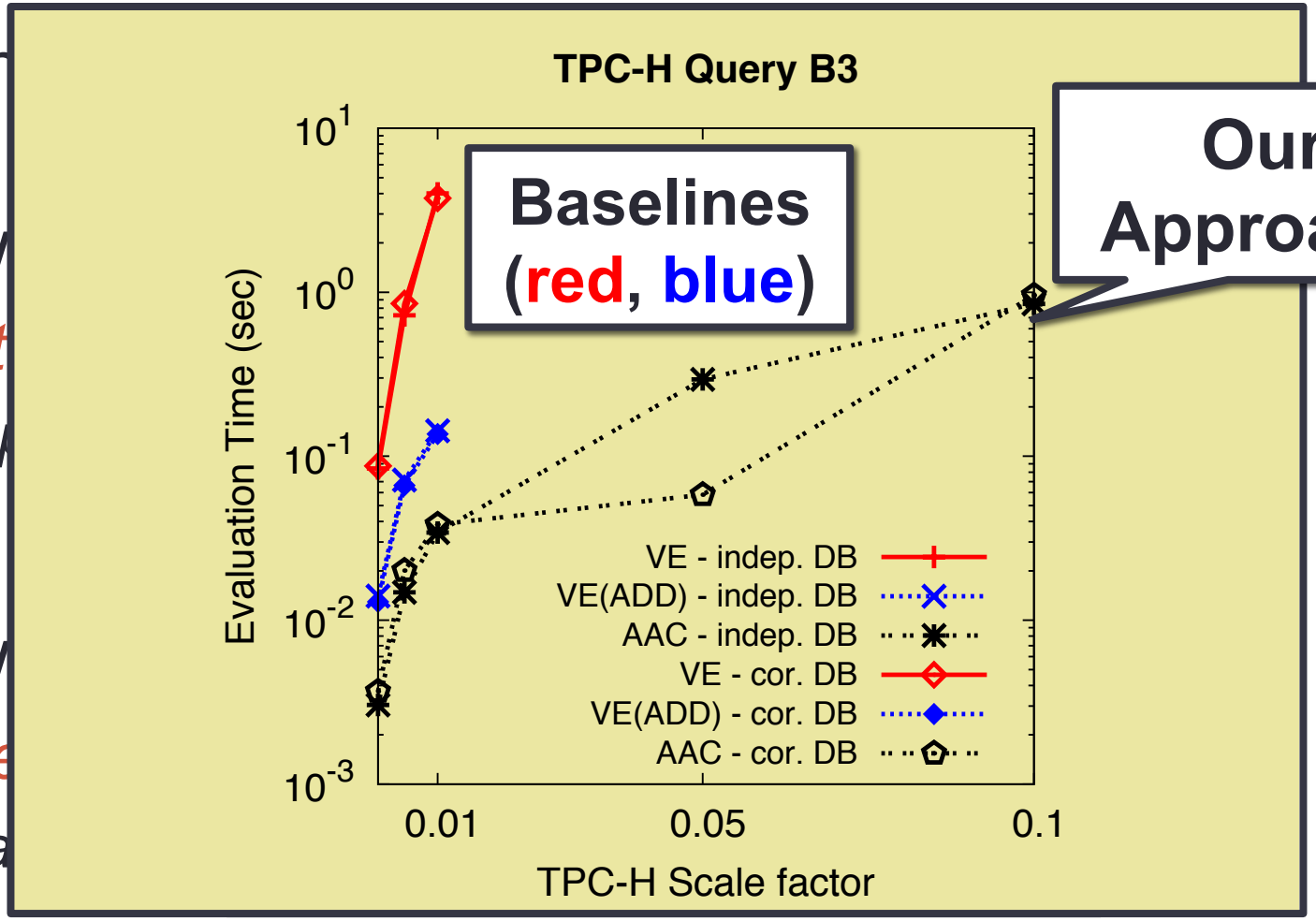
Annot

- *Exp*

2. How

Online

- *Tra*



Outline

- Background
- Arithmetic Circuits in probabilistic DBs
- Experiments
- Conclusion and future work

Outline

- Background
- Arithmetic Circuits in probabilistic DBs
- Experiments
- Conclusion and future work

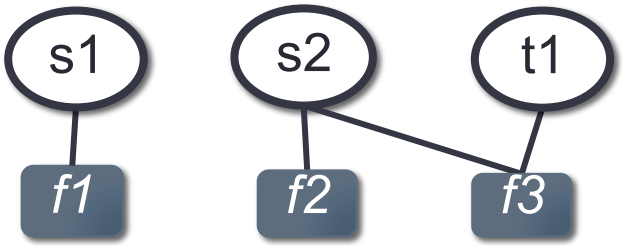
Example correlated probabilistic DB

Database

S	A	B
s1	'm'	1
s2	'n'	1

T	B	C
t1	1	'p'

Factor graph



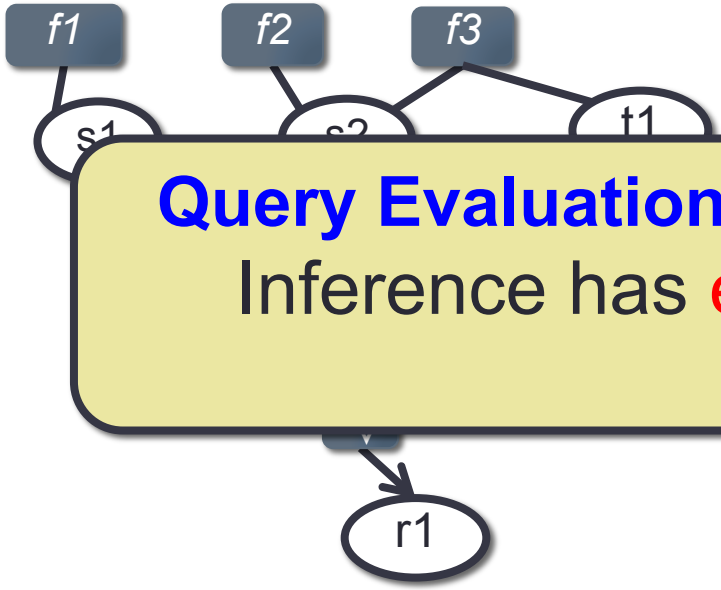
s1	$f_1(s1)$
0	0.6
1	0.4

s2	$f_2(s2)$
0	0.2
1	0.8

Query

Query: q(c):- s(a,b),t(b,c)

Lineage: $\ell = (s1 \wedge t1) \vee (s2 \wedge t1)$



Query Evaluation ≅ Probabilistic inference

Inference has exponential complexity

1	1	1

Speeding inference

Compilation: separate inference into two phases

Offline: compile distribution into a data structure

Online: use data structure to answer queries

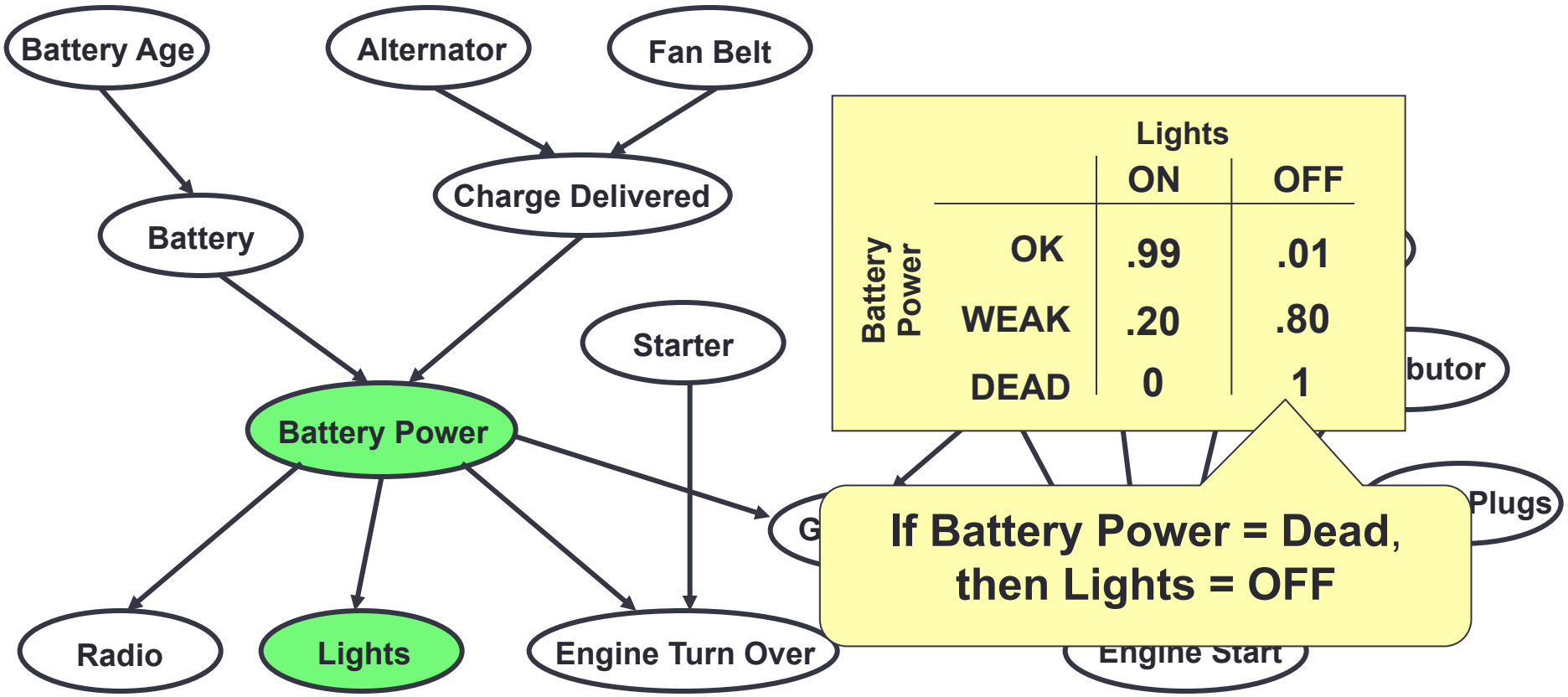
Goal: Push work to offline phase to optimize online inference time

Exploit local and global structure

- **Global structure:** Model topology (treewidth)
- **Local structure:** Exhibited in model parameters
  - Determinism
  - Context specific independence (CSI)

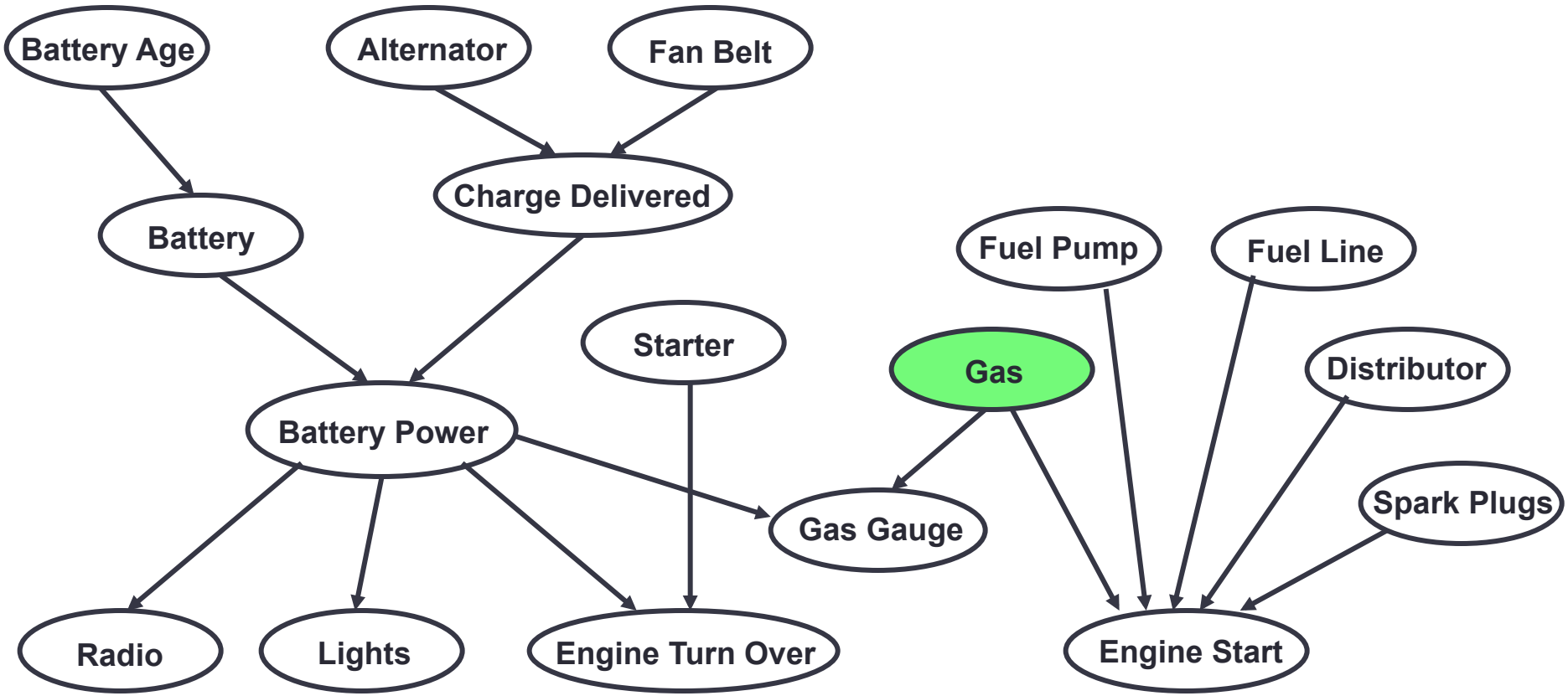
Determinism

Example Bayesian network [Darwiche, UAI `06]



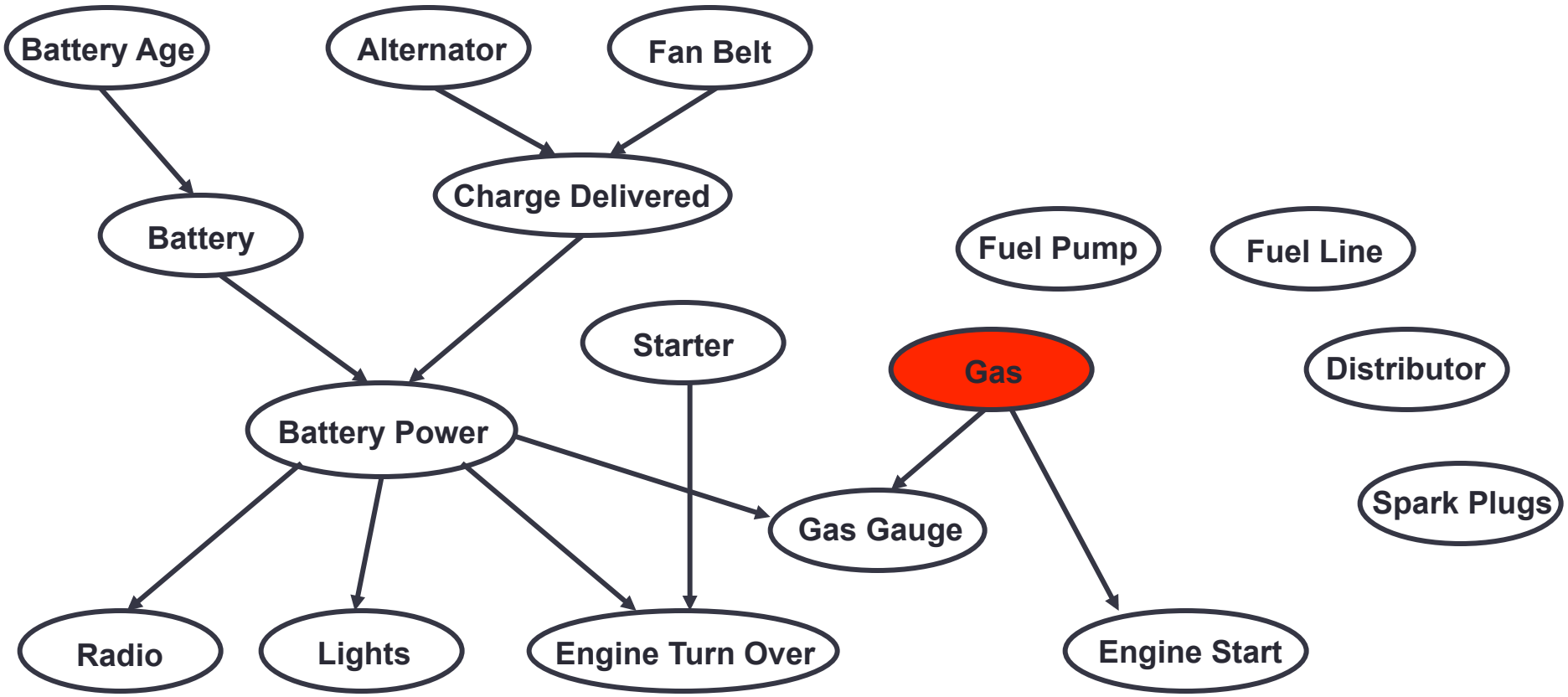
Context specific independence

Example Bayesian network [Darwiche, UAI `06]



Context specific independence

Example Bayesian network [Darwiche, UAI `06]

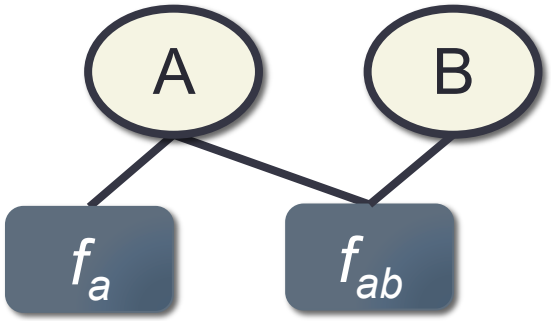


Overview of correlation representations

Data correlations	Query correlation
factor graphs [Sen et al., `07]	augmented factor graphs (variable elimination)
pc-tables [Green et al., `06]	lineage, OBDDs [Olteanu et al., `08], D-trees [Olteanu et al., `10] - OBDDs, D-trees exploit **local strucure**
junction trees [Kanagal et al., `10] - **compilation** - exploit **global** structure	lineage
arithmetic circuits [Darwiche, `03], AACs [this paper] - **compilation** - exploit **global** and **local** structure	[this paper]

Example arithmetic circuit

Factor graph

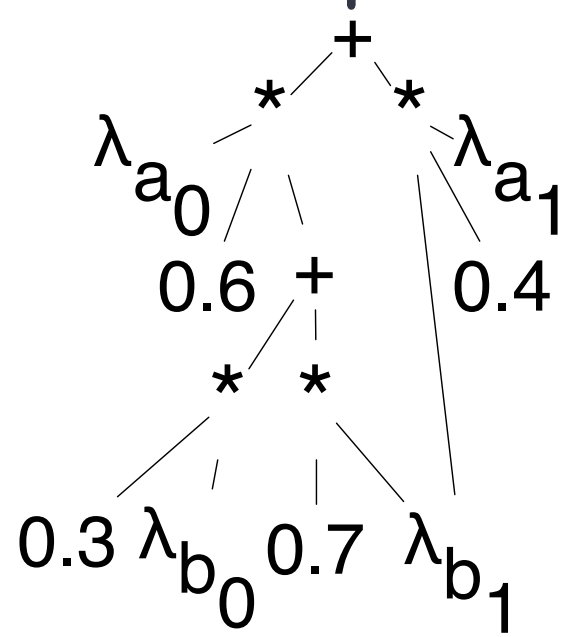


A	$f_a(.)$
0	0.6
1	0.4

A	B	$f_{ab}(.)$
0	0	0.3
0	1	0.7
1	0	0
1	1	1

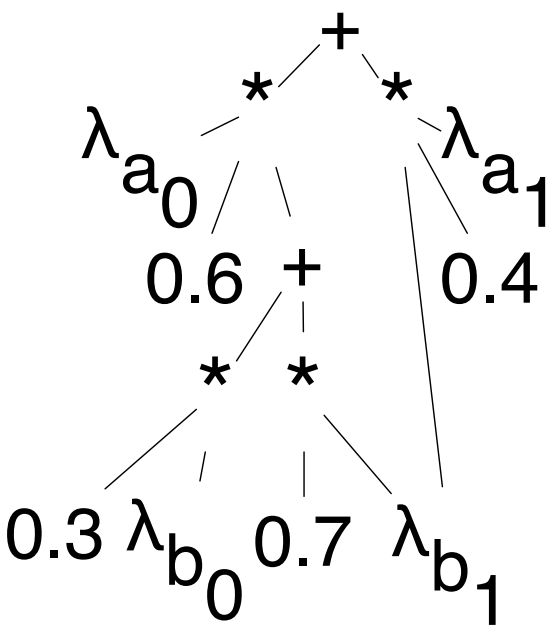
MLF (a_0 = false, a_1 = true):

$$\underbrace{\lambda_{a_0}\lambda_{b_0}\theta_{a_0}\theta_{b_0a_0} + \lambda_{a_0}\lambda_{b_1}\theta_{a_0}\theta_{b_1a_0} + \lambda_{a_1}\lambda_{b_0}\theta_{a_1}\theta_{b_0a_1} + \lambda_{a_1}\lambda_{b_1}\theta_{a_1}\theta_{b_1a_1}}_{\text{factorization}}$$



Querying an arithmetic circuit

Arithmetic circuit



Probability of evidence query

Pr(A = false, B = false)

- Set $\lambda_{a_0} = 1, \lambda_{a_1} = 0, \lambda_{b_0} = 1, \lambda_{b_1} = 0$
- Evaluate the circuit to get Pr(A = false, B = false) = 0.18

Outline

- Background
- Arithmetic Circuits in probabilistic DBs
- Experiments
- Conclusion and future work

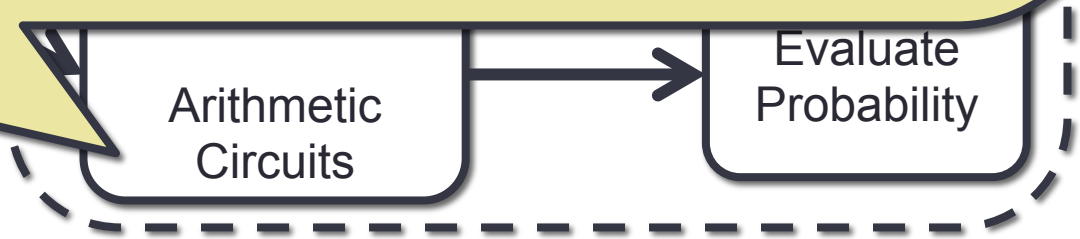
Overview of our Framework

Problem
No online updates

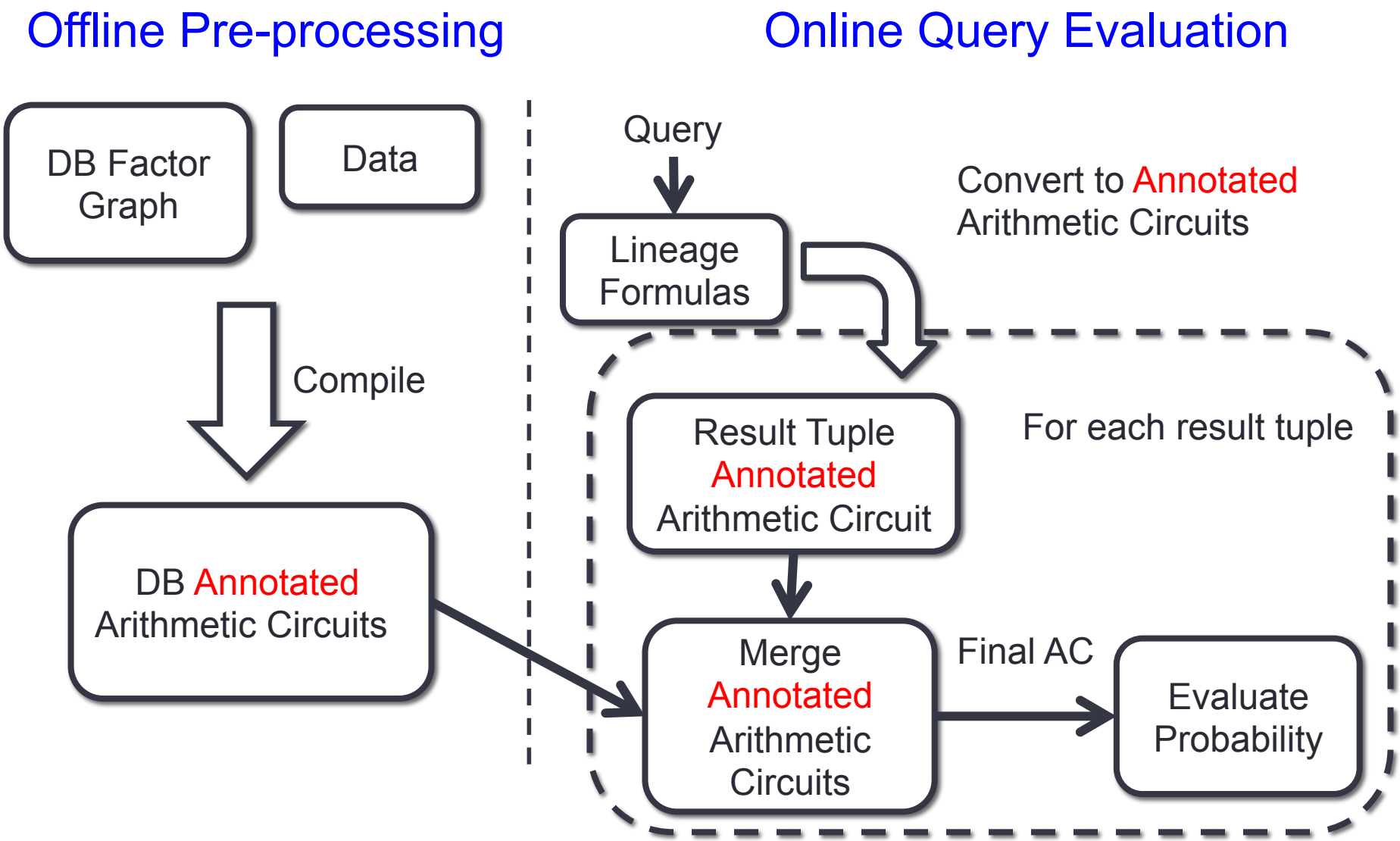
Solution (bad)
Repeat entire compilation process (expensive)

Solution (good)

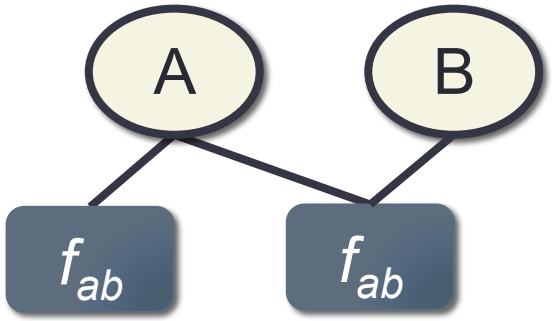
- Annotated Arithmetic Circuits (AACs): Add variable annotations to summation nodes, value annotations to edges
- Merge AACs during query evaluation



Overview of our Framework

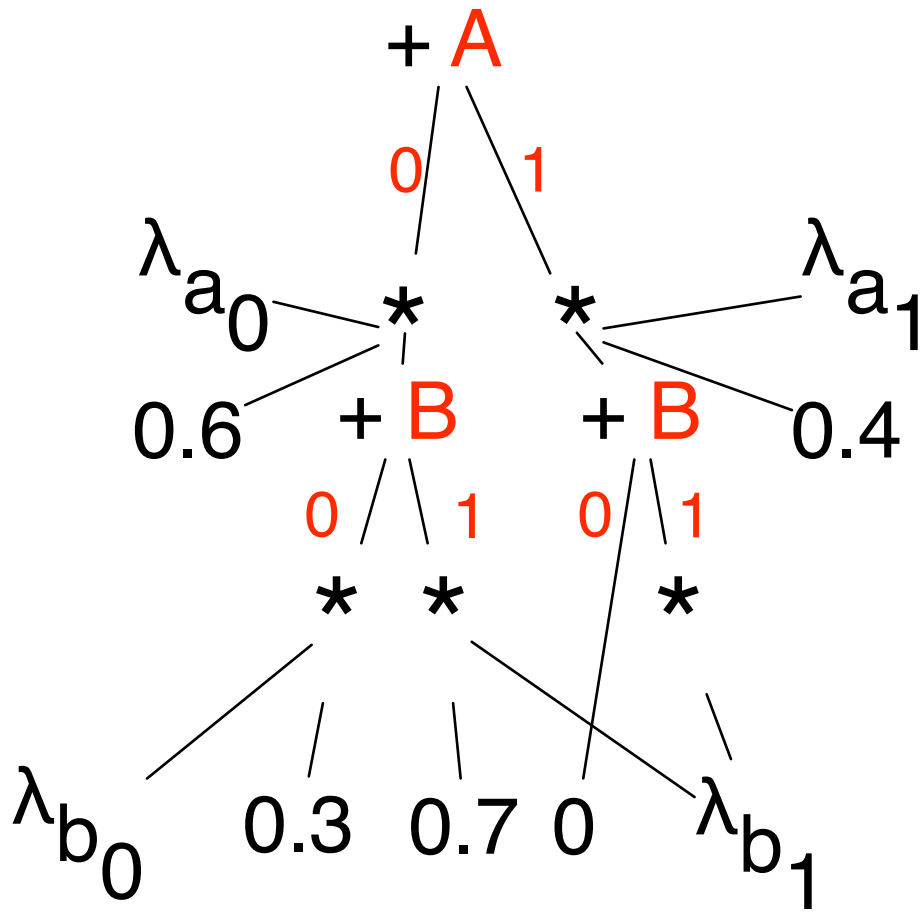


Example – Annotated Arithmetic Circuit

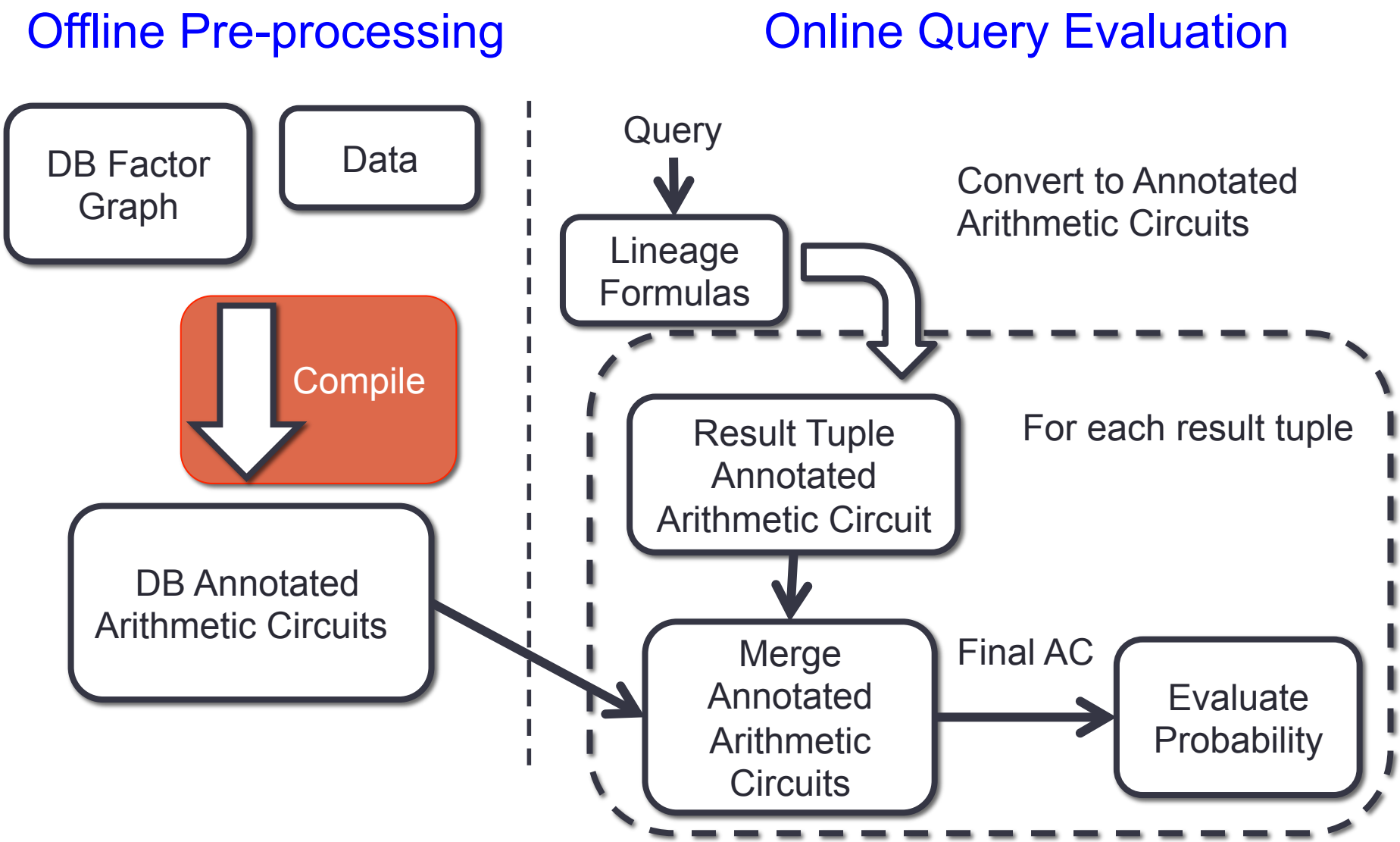


A	$f_a(.)$
0	0.6
1	0.4

A	B	$f_{ab}(.)$
0	0	0.3
0	1	0.7
1	0	0
1	1	1



Overview of our Framework



Compiling Factor Graphs

- Extend compilation technique based on VE [Chavira et al., `07]
  - Uses Algebraic Decision Diagrams (ADDs)
  - Add variable annotations to summation nodes
- VE requires a variable ordering
  - Output AACs follow the reverse VE ordering
- Collection of disjoint ordered sets of variables
  - One for each disconnected part of the factor graph

Example

Database

S	A	B
s1	‘m’	1
s2	‘n’	1

T	B	C
t1	1	‘p’

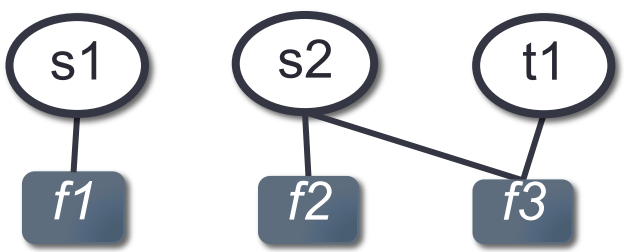
VE Ordering:

$O = s_1, s_2, t_1$

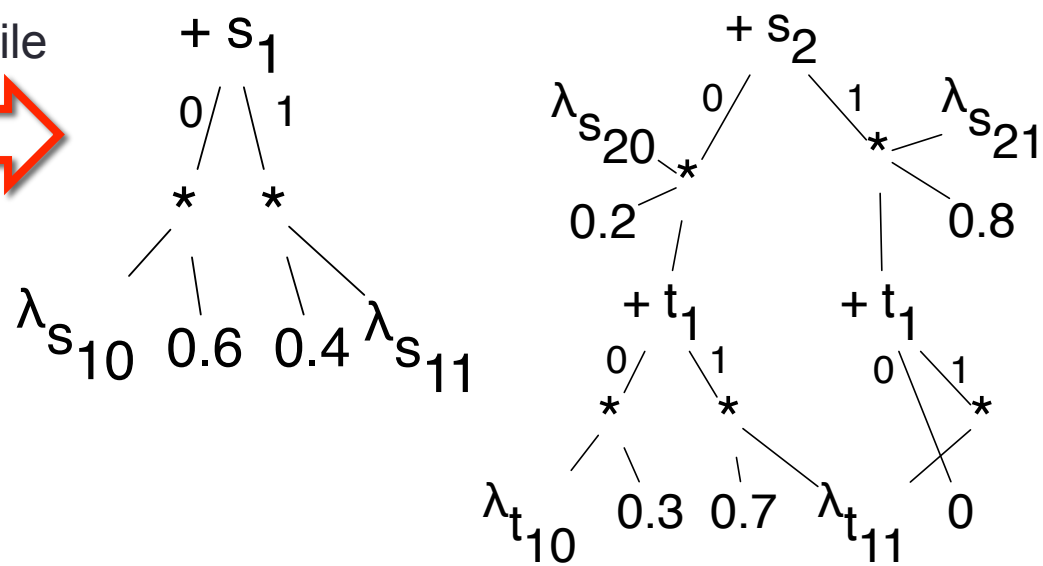
Partial Orderings:

$O_1 = s_1, O_2 = s_2, t_1$

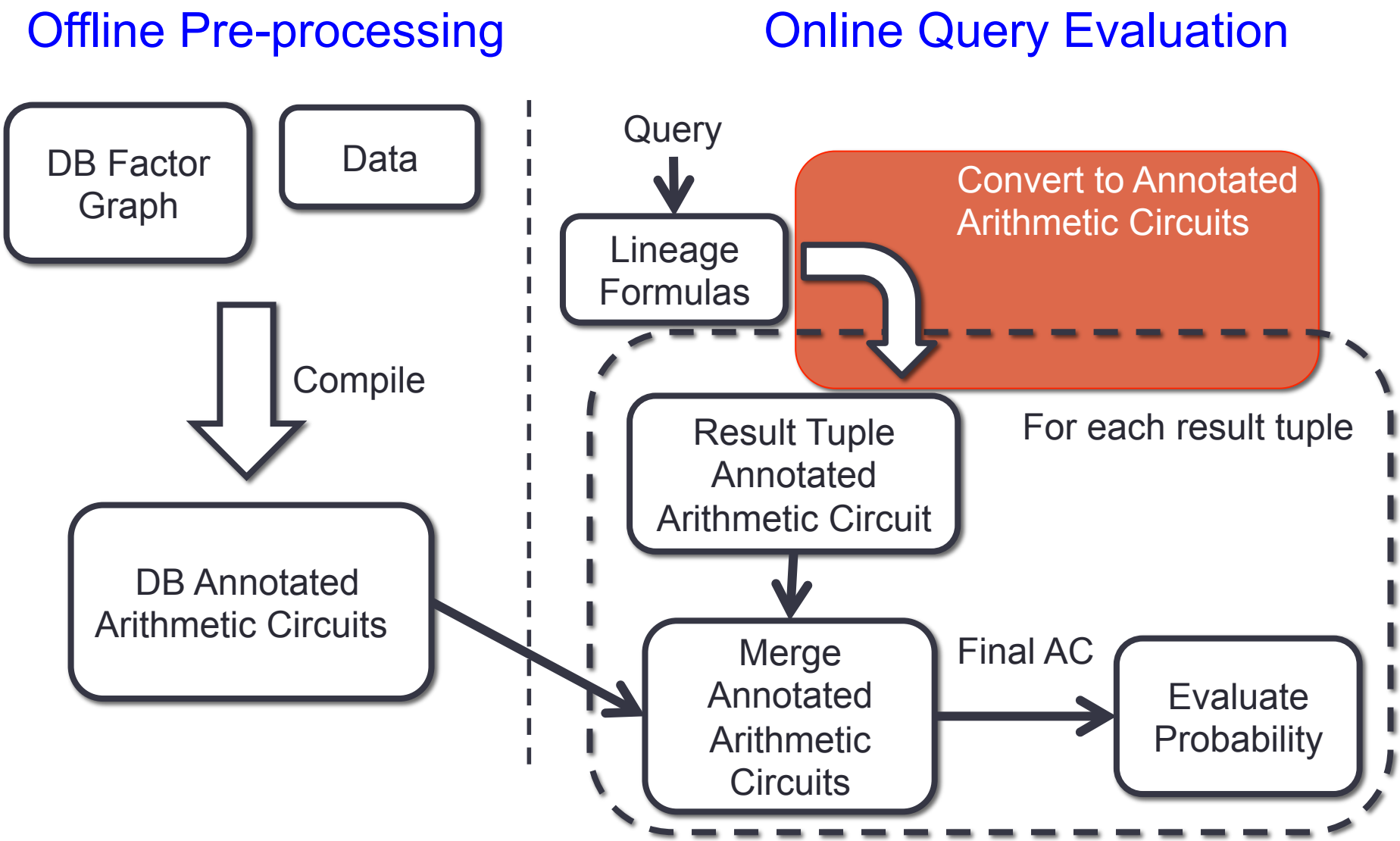
Factor graph



Compile



Overview of our Framework



Compiling Lineage

- Use Ordered Binary Decision Diagrams [Olteanu et al., 2008]
  - Lineage consists of only deterministic (AND/OR) factors
  - More efficient compilation

Compilation algorithm (sketch)

1. Choose a total variable ordering
2. Compile lineage formula to an OBDD
3. Construct an AAC from an OBDD

Getting a total ordering

- OBDD construction requires a total variable ordering
  - Finding an optimal variable ordering is NP-hard

Designing a heuristic (ideas):

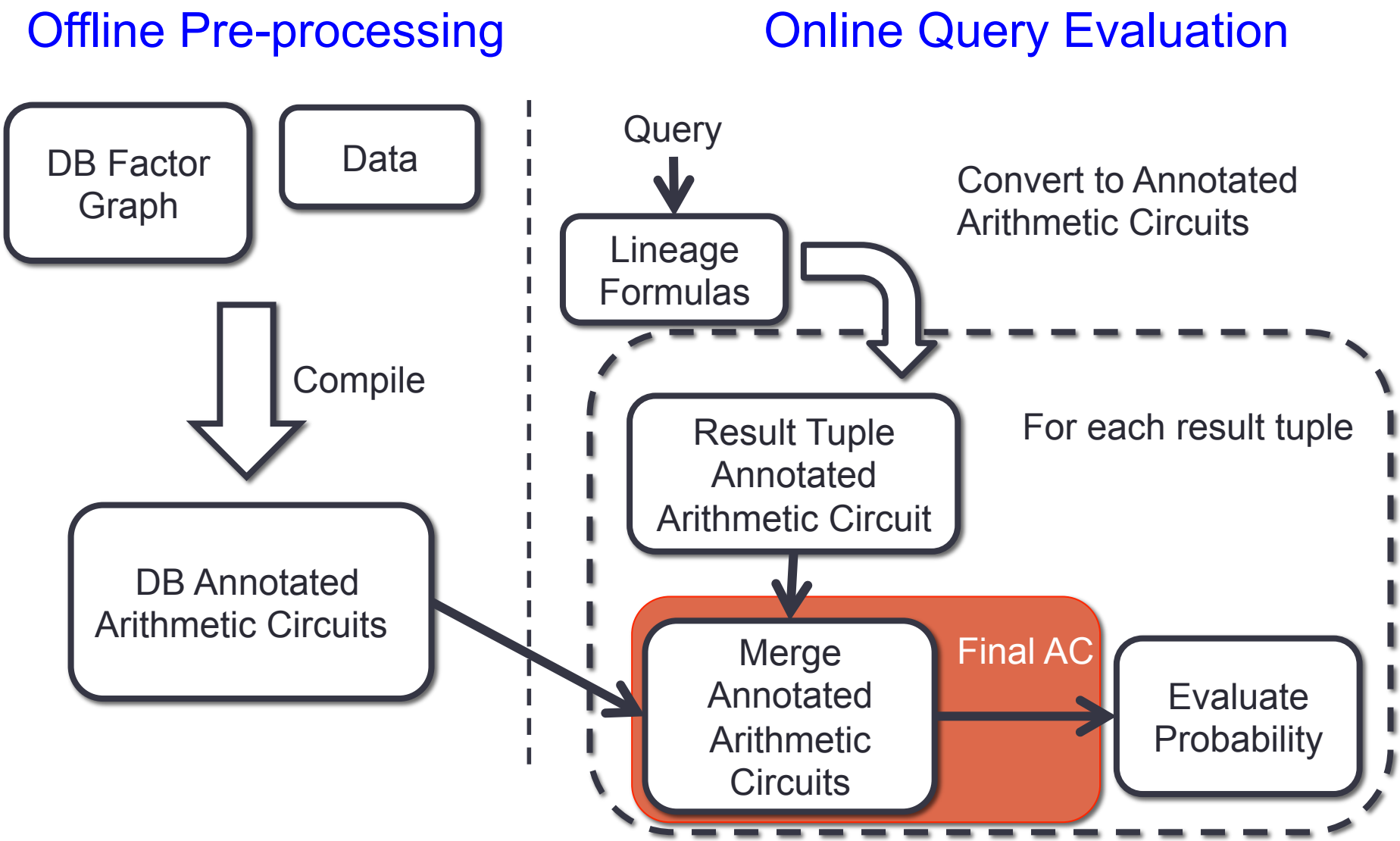
1. Connected variables in lineage must appear together [Ebendt et al., 2005]
2. More influential variables must appear sooner in the ordering [Ebendt et al., 2005]
3. Total ordering must respect partial ordering constraints
   - Necessary for merging
4. Variables in a constraint must be kept together
   - Facilitates caching during merging

Variable Ordering under Partial Order Constraints

Algorithm (Goal: Find total order of variables)

1. Score variables by their position in the partial orderings
2. Group random variables that are connected in the lineage
3. Use scores to get a total variable ordering

Overview of our Framework



Merging Annotated Arithmetic Circuits

Algorithm:

- Traverse the query and DB AACs simultaneously
- At each step merge exactly two AACs (rooted at summation nodes)
- If the result can be computed immediately, done
- Else: choose one root and recurse on its sub-circuits
  - Choose the root with the variable that appears first in the total ordering

Complexity: $O\left(m \sum_{i \in A} s_i\right)$

m: size of lineage-AAC, A: set of relevant database AACs,
s_i : size of i-th AAC in A

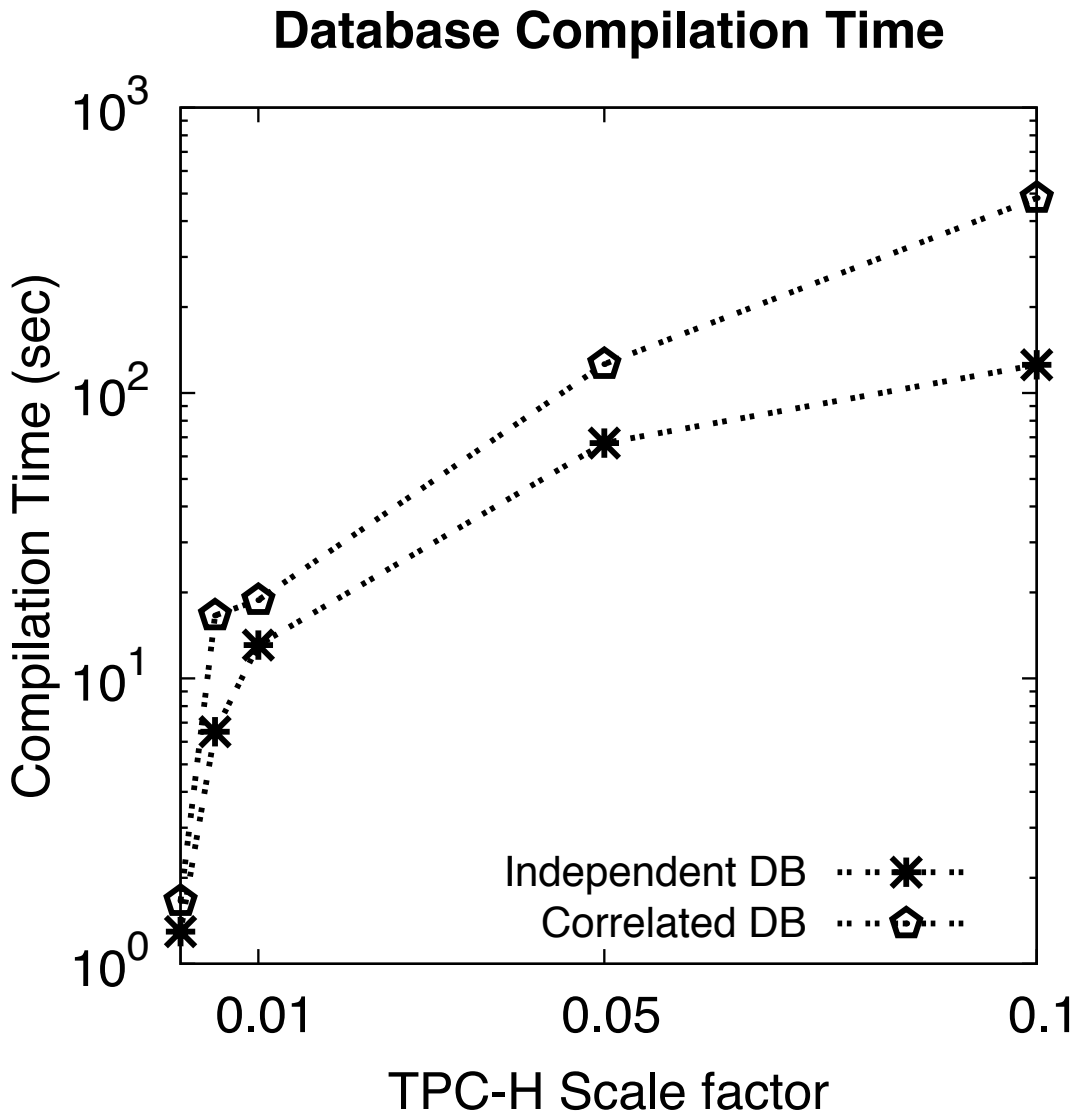
Outline

- Background
- Arithmetic Circuits in probabilistic DBs
- Experiments
- Conclusion and future work

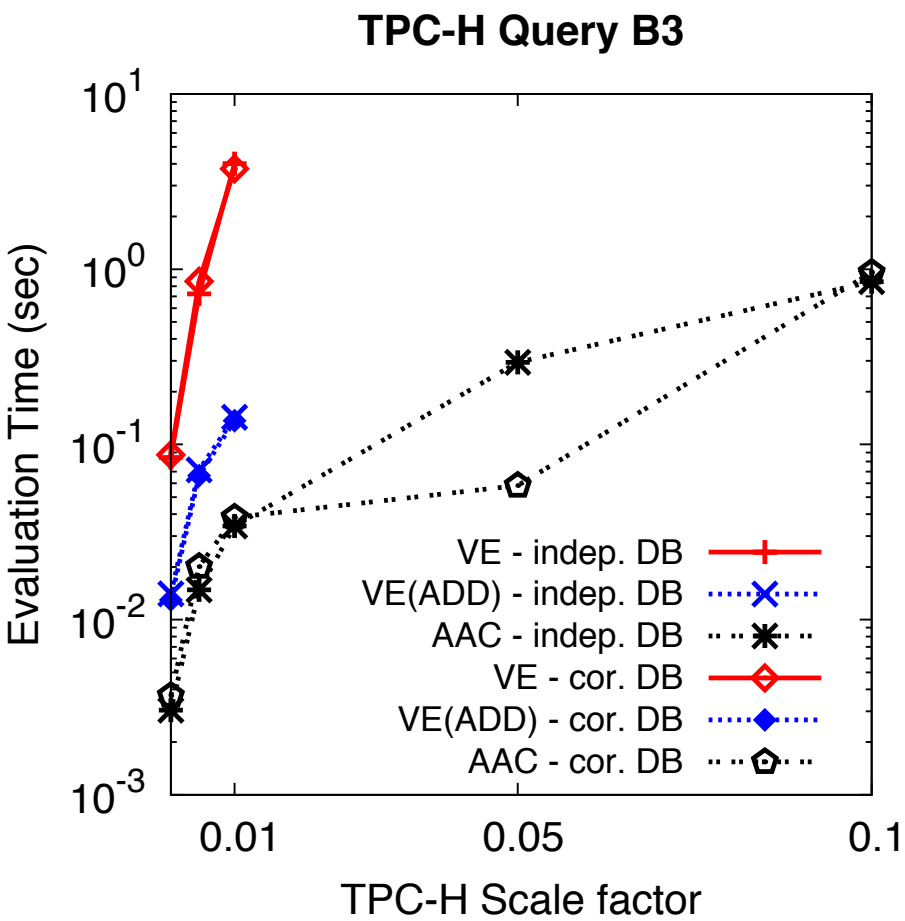
Experimental Evaluation

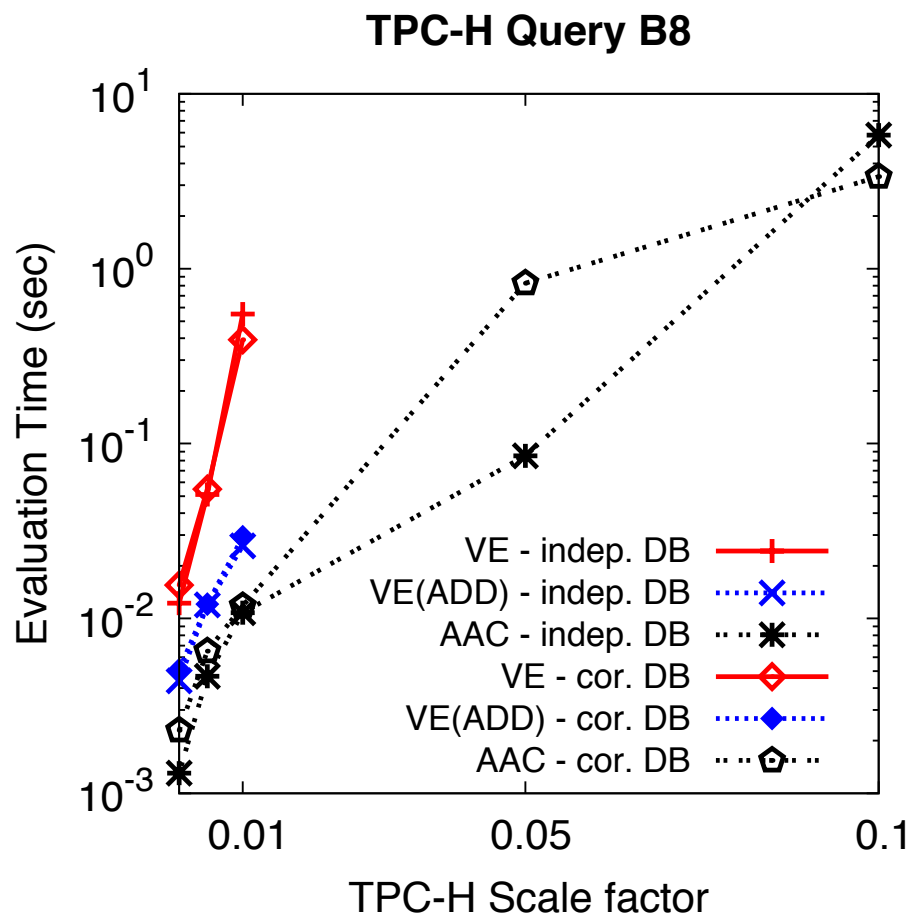
- Experimental evaluation using the TPC-H benchmark
  - Tuple independence
  - Schema level correlations
  - Multiple scale factors (0.001, 0.005, 0.01, 0.05, 0.1)
- Evaluation on queries with both safe and not-safe evaluation plans
  - Boolean versions of TPC-H queries Q2, Q3, Q5, Q6, Q8 and Q16 without top-level aggregates
- Comparison to:
  - Variable elimination with Conditional Probability Tables [Sen et al., `07]
  - Variable elimination with Algebraic Decision Diagrams
    - Captures determinism and context specific independence

Compilation Time



Evaluation Time

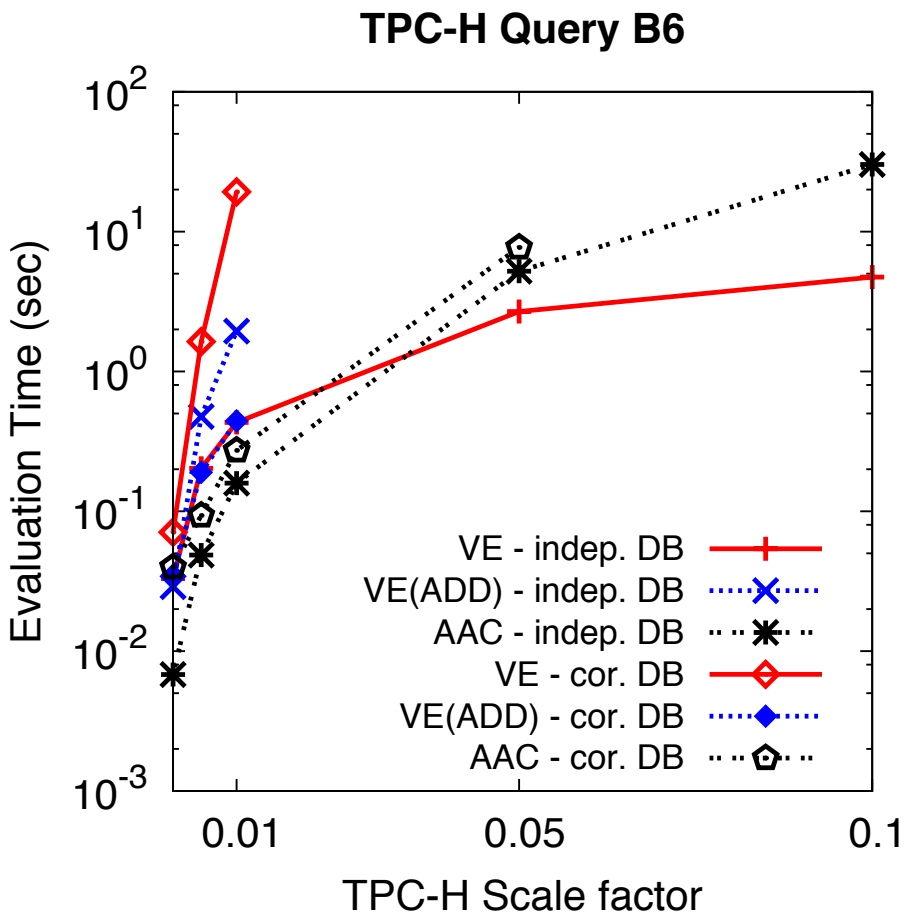


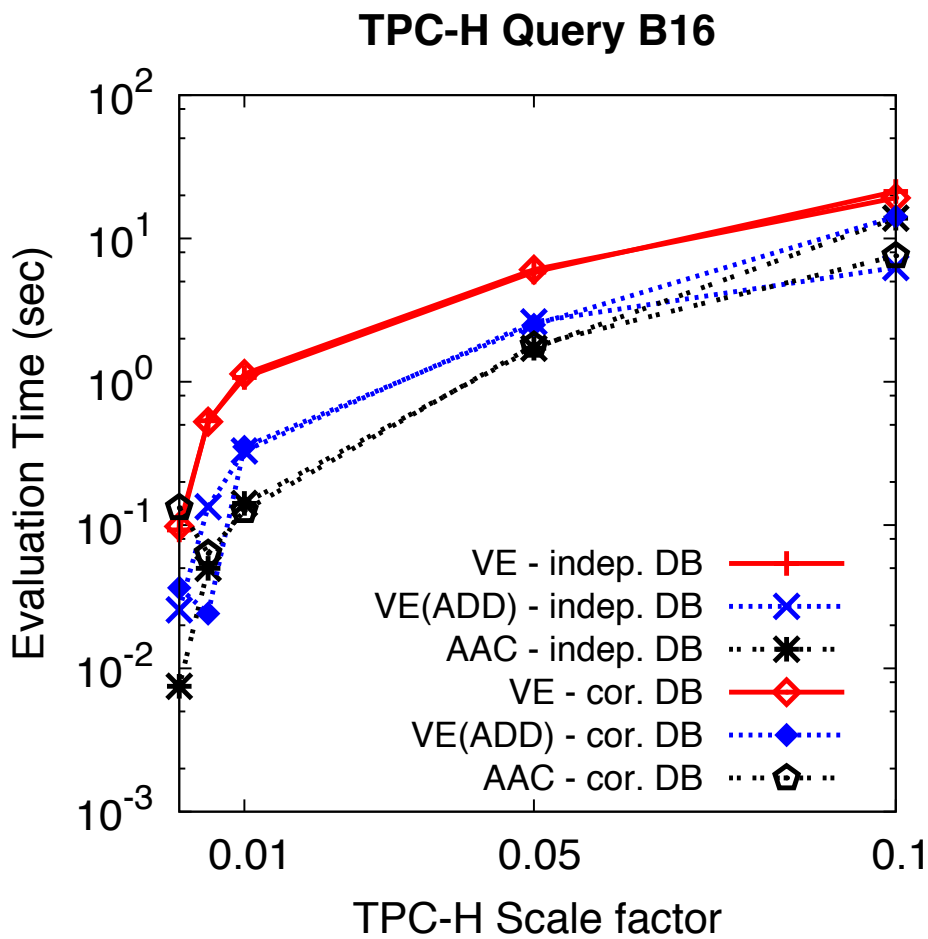


Missing points: Evaluation time exceeded 100 sec

Evaluation Time

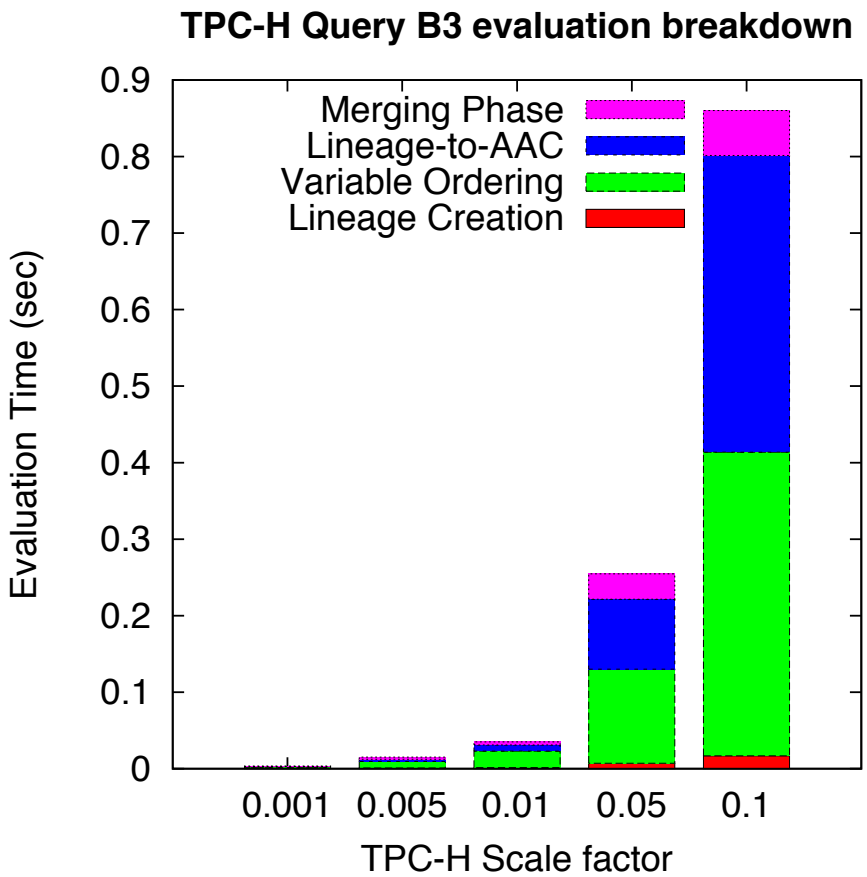
Longer lineage formulas

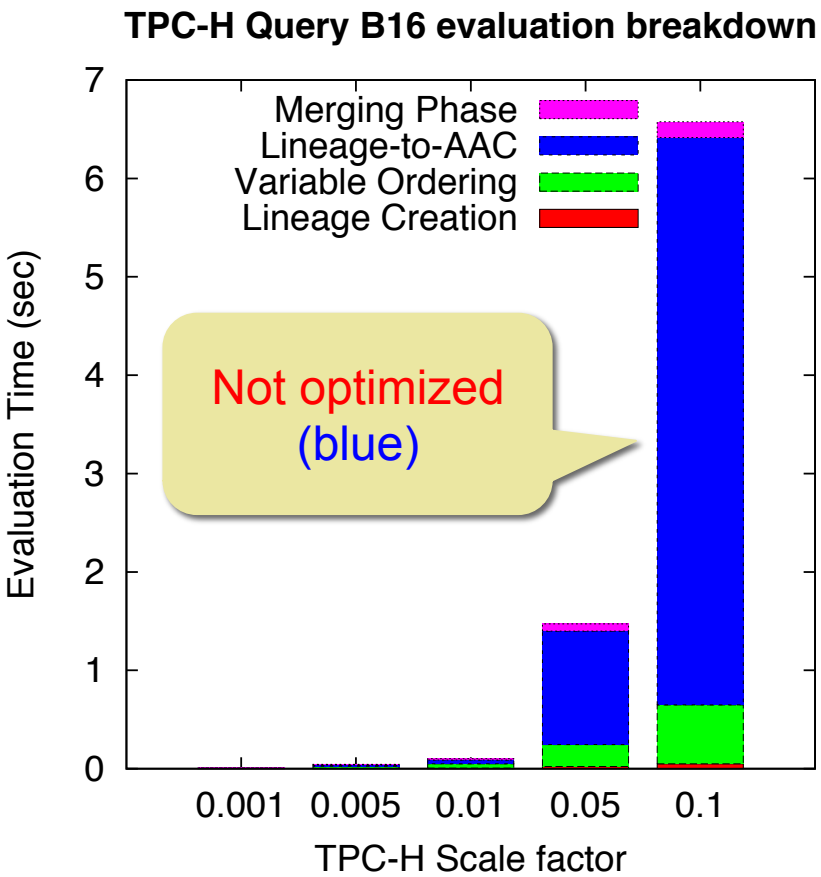




Missing points: Evaluation time exceeded 100 sec

Micro-benchmarking





Longer lineage formula

Outline

- Background
- Arithmetic Circuits in probabilistic DBs
- Experiments
- Conclusion and future work

Conclusion and future work

Conclusion

- Exploit compilation, global and local structure for efficient exact query evaluation
  - At least one order of magnitude improvement
- Introduced Annotated Arithmetic Circuits
  - New merging algorithm to support updates
  - Of independent interest to probabilistic reasoning

Thank you!

Open problems

- Lifted Inference in Arithmetic Circuits
  - isomorphic parts of an arithmetic circuit with respect to operations and θ variables ?
- Approximate inference in Arithmetic Circuits
  - How to incorporate approximations into the framework ?