Increasing the Representational Power and Scaling Reasoning in Probabilistic Databases

Amol Deshpande, University of Maryland

(joint work w/ Prof. Lise Getoor, Bhargav Kanagal, Jian Li, and Prithviraj Sen)

Motivation

- Increasing amounts of real-world uncertain data
  - Sensor networks, Scientific databases
    - Imprecise data, data with confidence or accuracy bounds
    - Widespread use of statistical and probabilistic models
  - Data integration
    - Noisy data sources, automatically derived schema mappings
    - Reputation/trust/staleness issues
  - Information extraction
    - Automatically extracted knowledge from text
  - Social networks, biological networks
    - Noisy, error-prone observations
    - Ubiquitous use of *entity resolution, link prediction, function prediction ..*
- Need to develop database systems for efficiently representing and managing uncertainty

Probabilistic Databases

- “Probability theory” a strong foundation to reason about the uncertainty
- Goal of Probabilistic Databases: Managing and querying large volumes of data annotated with probabilities
- Much work in recent years, leading up to many systems

  Mystiq (University of Washington)
  Trio (Stanford)
  MayBMS (Cornell, Oxford)
  PrDB (Maryland)
  MCDB (Univ. of Florida, IBM)
  Orion (Purdue University)
  BayesStore (Berkeley)
  Lahar (University of Washington)
  …..

- Other work on approximations, ranking, indexing, summarization etc.
- But, many challenges still remain…

Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

Probabilistic Databases

- Types of uncertainties typically supported
  - *Tuple-existence* uncertainty
    - A tuple may or may not exist in the database
    - *e.g. a sensor may detect a bird, but not 100% sure*
  - *Attribute-value* uncertainty
    - The *value* of an attribute not known precisely
    - Instead a distribution over possible values is provided
    - *e.g. a sensor detects a bird for sure, but it may be a sparrow or a dove or something else*
- Most systems assume discrete probability distributions, but some support continuous distributions as well
- Largely based on the *possible worlds semantics*

An Example Probabilistic Database

- Example from Dalvi and Suciu [2004]
- Assume independent tuples

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

Interpret as a distribution over a set of deterministic possible worlds

Possible worlds

instance	probability
{s1, s2, t1}	0.12
{s1, s2}	0.18
{s1, t1}	0.12
{s1}	0.18
{s2, t1}	0.08
{s2}	0.12
{t1}	0.08
{}	0.12

*p(s1) * p(t1) * (1-p(s2))*
*= 0.6 * 0.4 * 0.5*
= 0.12

Query Processing Semantics

- Evaluate on each possible world and combine results
- Example Query: $\pi_C(S \bowtie_B T)$

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

$\bowtie$

	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'

π_C

	C
r1	'p'

Query Processing Semantics

- Evaluate on each possible world and combine results
- Example Query: $\pi_C(S \bowtie_B T)$

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

<u>*Possible worlds*</u> <u>*Query Result*</u>

instance	prob	result
{s1, s2, t1}	0.12	{'p'}
{s1, s2}	0.18	{}
{s1, t1}	0.12	{'p'}
{s1}	0.18	{}
{s2, t1}	0.08	{'p'}
{s2}	0.12	{}
{t1}	0.0	
{}	0.1	

	C	*prob*
r1	'p'	0.32

Not clear how to do this in general

e.g. ranking ??

Consensus Answers [PODS'09]

Query Processing

- Several approaches proposed in recent years in DB literature
  - Typically make strong independence assumptions
  - Limited support for attribute-value uncertainty
  - In spite of that, query evaluation known to be #P-Hard [DS'04]
    - For very simple 3-relation queries
- **Our Goals:**
  - Increase representationl power to support:
    - Correlations among the data items
    - Uncertainties at different abstraction levels and granularities
  - Scale reasoning and querying to large-scale uncertain data while supporting the above

Correlations in Uncertain Data

- Most application domains generate correlated data
  - Data Integration
    - Conflicting information best captured using "mutual exclusivity"
    - Data from the same source may all be valid or may all be invalid

	Name	Salary
DB1:	John	$1200
DB2:	John	$1600

Name	Salary	*prob*
John	$1200	*0.3*
John	$1600	*0.7*

Mutually exclusive

  - Information extraction
    - Annotations on consecutive text segments strongly correlated

Bill can be reached at 3-4057

annotation: first name (prob: 80%)

annotation: phone number (prob: 90%)

High positive correlation

Correlations in Uncertain Data

- Most application domains generate correlated data
  - Data Integration
    - Conflicting information best captured using “mutual exclusivity”
    - Data from the same source may all be valid or may all be invalid
  - Information extraction
    - Annotations on consecutive text segments strongly correlated
  - Social networks
    - Attributes of neighboring nodes often highly correlated
    - Predicted links, class labels likely to be correlated
  - Sensor network data
    - Very strong spatio-temporal correlations
- Even if base data exhibits independence..
  - Correlations get introduced during query processing

Correlations in Uncertain Data

- Even if base data exhibits independence..
  - Correlations get introduced during query processing

S

	A	B	*prob*
s1	'm'	1	0.6
s2	'n'	1	0.5

T

	B	C	*prob*
t1	1	'p'	0.4

$S \bowtie T$

	A	B	C	*prob*
i1	'm'	1	'p'	0.24
i2	'n'	1	'p'	0.20

*p(i1) = p(s1) * p(t1)*
*= 0.6 * 0.4 = 0.24*

Correlated

π_C

	C
r1	'p'

Shared Uncertainties and Correlations

- Uncertainties and correlations often specified for groups of tuples rather than for individual tuples
- Necessary when trying to model and reason about uncertainty in large populations

AdID	Model	Color	Price
1	Honda	?	$9,000
2	?	Beige	$8,000
3	?	?	$6,000
…	…	…	…
…	…	…	…
…	…	…	…
1000000	?	?	$10,000

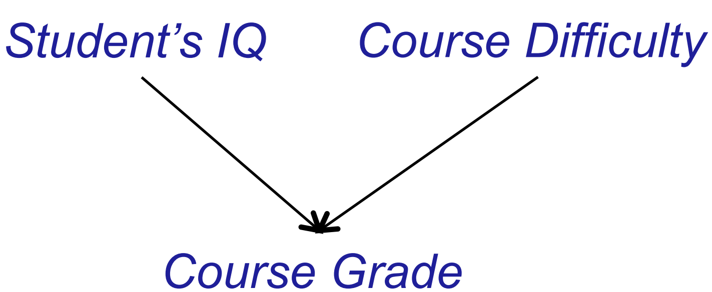
Model	Pr(M)
Honda	0.2
Mazda	0.1
…	…

Model	Color	Pr(C|M)
Honda	Beige	0.1
Honda	Red	0.2
…	…	…
Mazda	Beige	0.02

A Used Car Ads Database

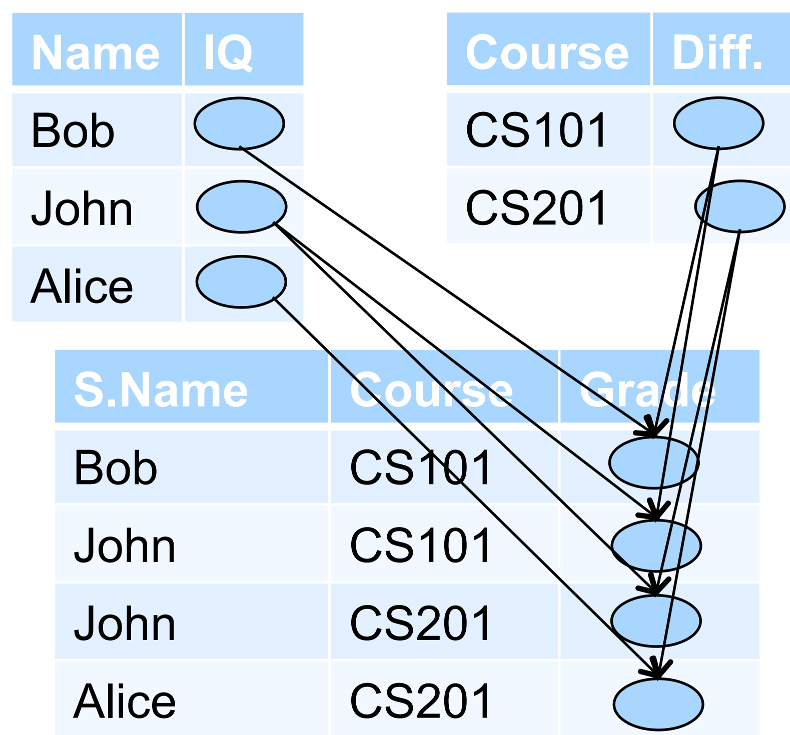
Schema-level Uncertainties

- Often we have probabilistic knowledge at the schema level (learned from a deterministic database) that we are trying to transfer
  - Using Prob. relational models (PRMs), Relational Markov networks (RMNs) etc. ("Intro. to Statistical Relational Learning"; Getoor and Taskar, 2007)



A "Schema-level" Dependence

An Instantiation



First-order Logic and Uncertainties

- Often need to reason about uncertainties at the first-order level

Example from "Markov Logic Networks"; Richardson and Domingos [2006]

English and First Order Logic	Clausal Form	Weight
"Friends of friends are friends" $\forall x \forall y \forall z\ Fr(x, y) \wedge F(y, z) \Rightarrow Fr(x, z)$	$\neg Fr(x, y) \vee \neg F(y, z) \vee Fr(x, z)$	0.7
"Smoking causes cancer". $\forall x\ Sm(x) \Rightarrow Ca(x)$	$\neg Sm(x) \vee Ca(x)$	1.5
"Friends have similar behavior w.r.t. smoking." $\forall x \forall y\ Fr(x, y) \wedge Sm(x) \Rightarrow Sm(y)$	$\neg Fr(x, y) \vee \neg Sm(x) \vee Sm(y)$	1.1

- Rules do not always hold – hence may choose to augment them with weights (approach taken in *Markov Logic Networks*)
  - *Hard vs soft constraints*

Markov Logic Networks

- A specific population defines a specific Markov network
  - Given persons: *Anna, Frank, Bob*
  - We get the (boolean) variables:
    - *Friends(Anna, Frank), Friends(Anna, Bob), Friends(Frank, Bob), …*
    - *Smokes(Anna), Smokes(Frank), Smokes(Bob), …*
    - *Ca(Anna), Ca(Frank), Ca(Bob), …*
  - An instantiation to these variables (true or false) is a possible world
  - Possible worlds that violate fewer constraints have higher probabilities
    - According to the weights
- Typical inference task: *find the most likely world*
- May want to treat the output as an uncertain database and support rich querying constructs

Reasoning over Correlated, Uncertain Data

- Huge body of work in Machine Learning community on this topic
  - Bayesian and Markov networks, statistical relational models (PRMs, MRNs)
  - On efficient algorithms for reasoning, for inference, for learning …
  - As much emphasis on *learning* as on *inference*
- Lot of work in recent years in the Probabilistic Databases literature
  - On efficient SQL query processing over very large amounts of data
  - Comparatively simpler uncertainty structures
- How to combine the representational power and richness of ML approaches with the ability to execute declarative queries over large volumes of data ?

PrDB Framework

- Flexible uncertainty model (based on probabilistic graphical models)
  - Support for representing rich correlation structures [ICDE'07]
  - Support for specifying uncertainty at multiple abstraction levels [DUNE'07]
- Declarative constructs for interacting with the database
  - Manipulating and updating uncertainty as a first class citizen
- Rich querying semantics
  - SQL queries; Inference, reasoning, and what-if queries
- New techniques for scaling reasoning and query processing
  - Inference techniques to exploit the structure in the data [VLDB'08]
  - Index structures for handling large volumes of data [SIGMOD'09,'10]
  - Efficient algorithms for ranking queries, consensus answers [VLDB'09,PODS'09]
  - Approximation techniques that enable tradeoff accuracy and speed [UAI'09]

Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

A Simple Example

- Represent the uncertainties and correlations *graphically* using small functions called *factors*
  - Concepts borrowed from the *graphical models* literature

0 = Tuple does not exist
1 = Tuple exists

Often not probability distributions
Values can be > 1

S

	A	B	prob
s1	‘m’	1	0.6
s2	‘n’	1	0.5

T

	B	C	prob
t1	1	‘p’	0.4

s1	$f_1(s1)$
0	0.4
1	0.6

s2	***t1***	$f_2(s2, t1)$
0	0	0.1
0	1	0.5
1	0	0.4
1	1	0

s2 and t1 mutually exclusive

A Simple Example

- Represent the uncertainties and correlations *graphically* using small functions called *factors*
  - Concepts borrowed from the *graphical models* literature

S

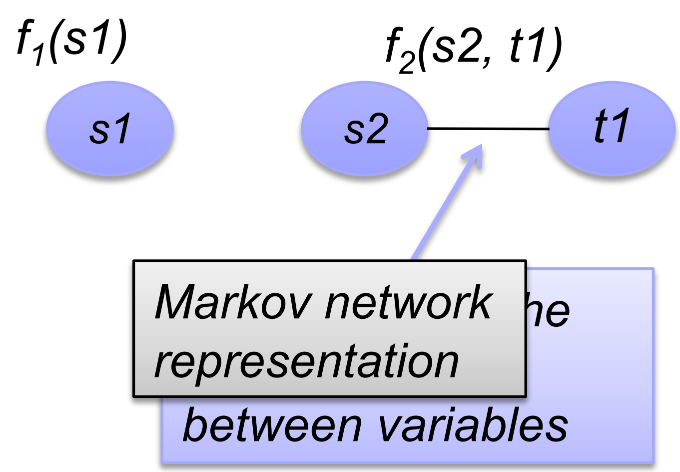
	A	B	prob
s1	'm'	1	0.6
s2	'n'	1	0.5

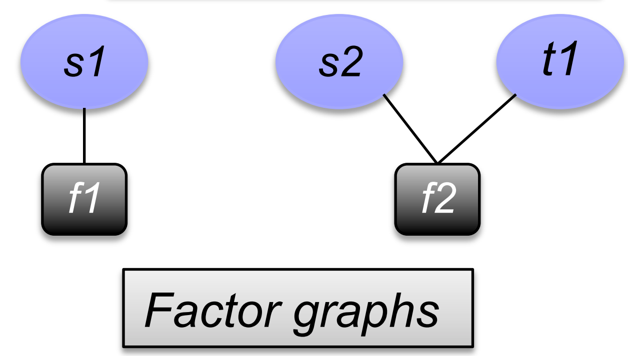
T

	B	C	prob
t1	1	'p'	0.4

s1	$f_1(s1)$
0	0.4
1	0.6

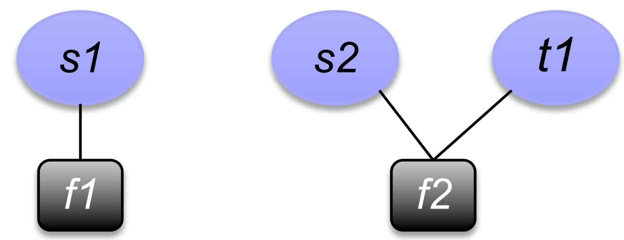
s2	*t1*	$f_2(s2, t1)$
0	0	0.1
0	1	0.5
1	0	0.4
1	1	0





Probabilistic Graphical Models

- A PGM can compactly represent a joint probability distribution over a large number of random variables with complex correlations
- Specified completely by:
  - A set of random variables
  - A set of factors over the random variables
- Joint pdf obtained by multiplying all the factors and normalizing
- An *Inference* task: Finding a marginal prob. distribution over subset of variables
  - e.g. $Pr(t_1)$



$Pr(s_1, s_2, t_1) \;\propto\; f_1(s_1)\, f_2(s_2, t_1)$

<u>For example:</u>

$$Pr(s_1 = 0, s_2 = 0, t_1 = 0) = \frac{1}{Z} f_1(s_1 = 0)\, f_2(s_2 = 0, t_1 = 0)$$

Normalizing Constant ("Partition Function")

A Simple Example

- During query processing, add new deterministic factors (hard constraints) corresponding to intermediate tuples
  - Encode the dependencies between base tuples and intermediate tuples
- Example query: $\pi_C(S \bowtie_B T)$

S

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

T

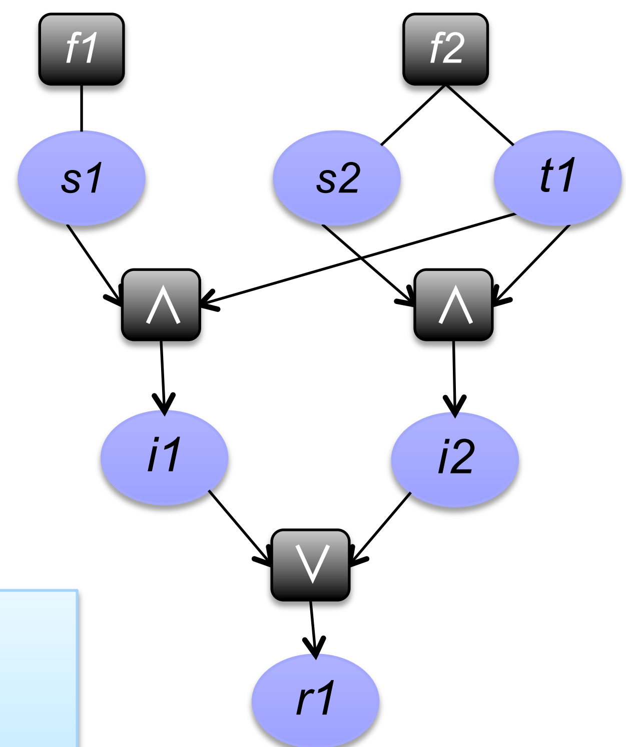
	B	C
t1	1	'p'

$S \bowtie T$

	A	B	C
i1	'm'	1	'p'
i2	'n'	1	'p'

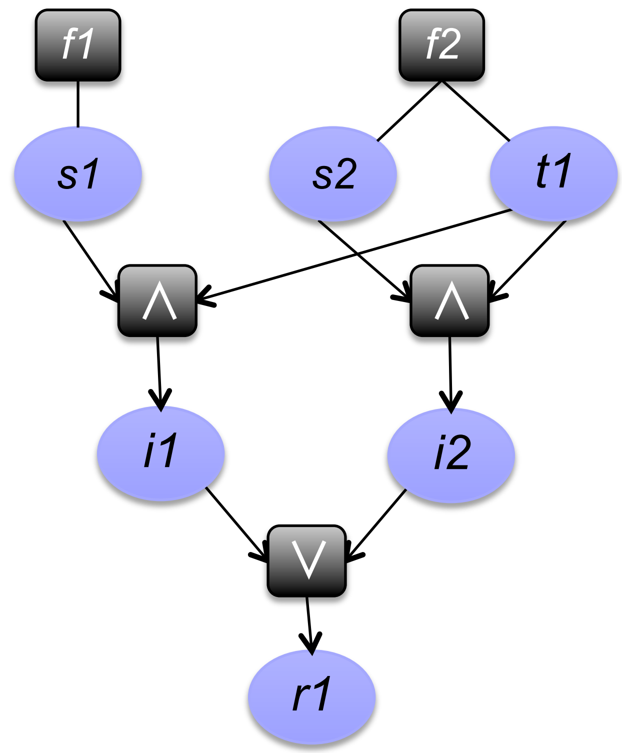
π_C

IF: *s1 and t1 are 1*
THEN: *Pr(i1 = 1) = 1, Pr(i1 = 0) = 0*
ELSE: *Pr(i1 = 1) = 0, Pr(i1 = 0) = 1*



Probabilistic Graphical Models

- A PGM can compactly represent a joint probability distribution over a large number of random variables with complex correlations
- Specified completely by:
  - A set of random variables
  - A set of factors over the random variables
- Joint pdf obtained by multiplying all the factors and normalizing



$Pr(s_1, s_2, t_1, i_1, i_2, r_1) \propto f_1(s_1)\, f_2(s_2, t_1)\, f^{\wedge}(s_1,t_1,i_1)\, f^{\wedge}(s_2,t_1,i_2)\, f^{\vee}(i_1,i_2,r_1)$

A Simple Example

- Query evaluation ≡ Find the result tuple probabilities ≡ Inference !!
  - Can use standard techniques like *variable elimination, junction trees (exact), message passing, loopy Belief propagation, Gibbs Sampling (approx)*

S

	A	B
s1	‘m’	1
s2	‘n’	1

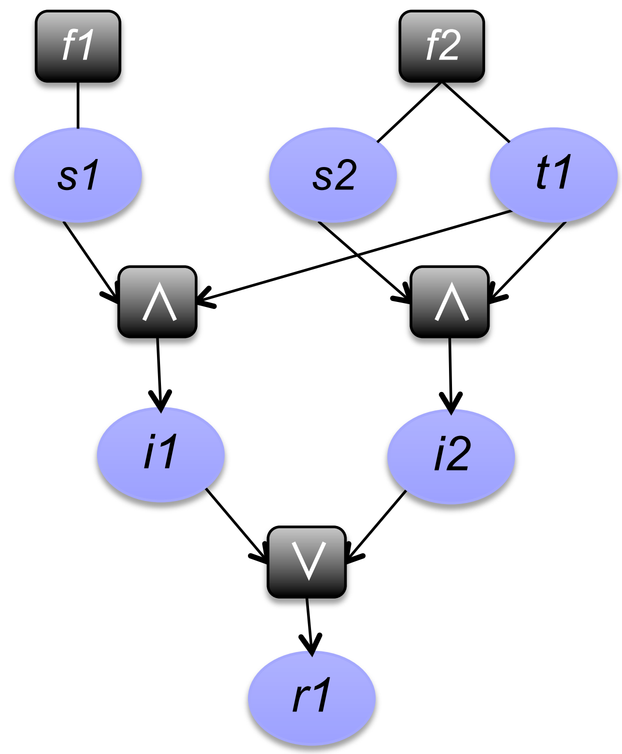
T

	B	C
t1	1	‘p’

	A	B	C
i1	‘m’	1	‘p’
i2	‘n’	1	‘p’

π_C

	C
r1	‘p’

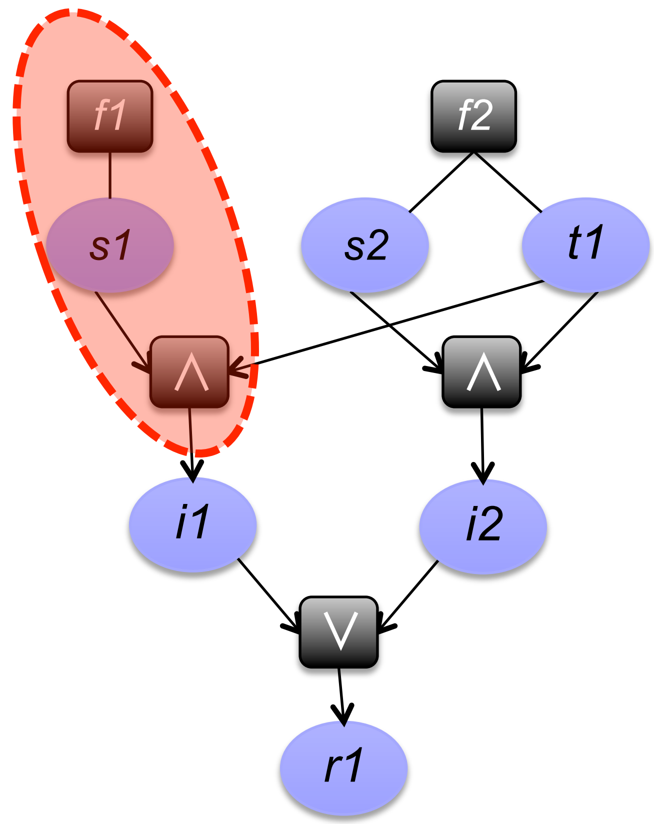


A Simple Example: Inference

- Variable Elimination
  - Sum-out non-query random variables one by one
    - Collect factors for that variable, multiply them, and sum out the variable

$$P(r_1) = \sum_{s1,\, s2,\, t1,\, i1,\, i2} Pr(s_1, s_2, t_1, i_1, i_2, r_1)$$

$$\propto \sum f_1(s_1)\, f_2(s_2, t_1)\, f^{\wedge}(s_1,t_1,i_1)\, f^{\wedge}(s_2,t_1,i_2)\, f^{\vee}(i_1,i_2,r_1)$$



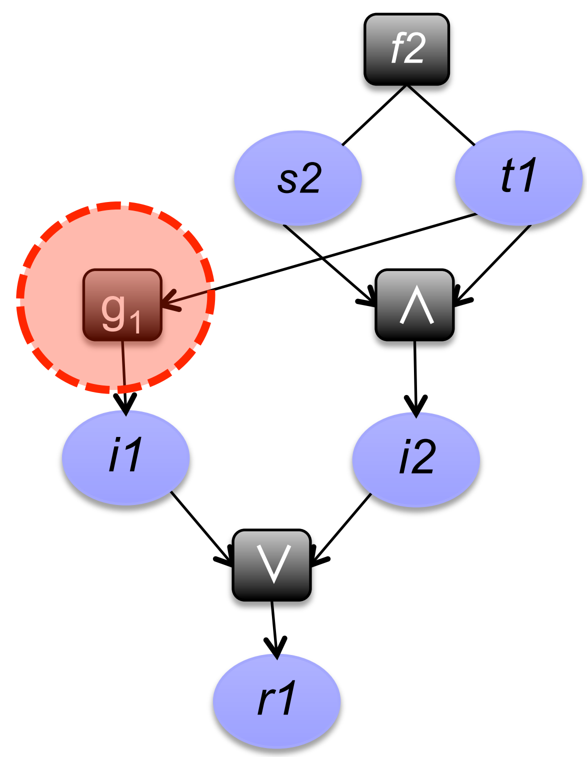
A Simple Example: Inference

- Variable Elimination
  - Sum-out non-query random variables one by one
    - Collect factors for that variable, multiply them, and sum out the variable
  - <u>Elimination Order</u>: The order in which to sum-out the random variables
    - Choosing a good elimination order critical for performance (NP-Hard)

$P(r_1) = \sum_{s1,\ s2,\ t1,\ i1,\ i2} Pr(s_1, s_2, t_1, i_1, i_2, r_1)$

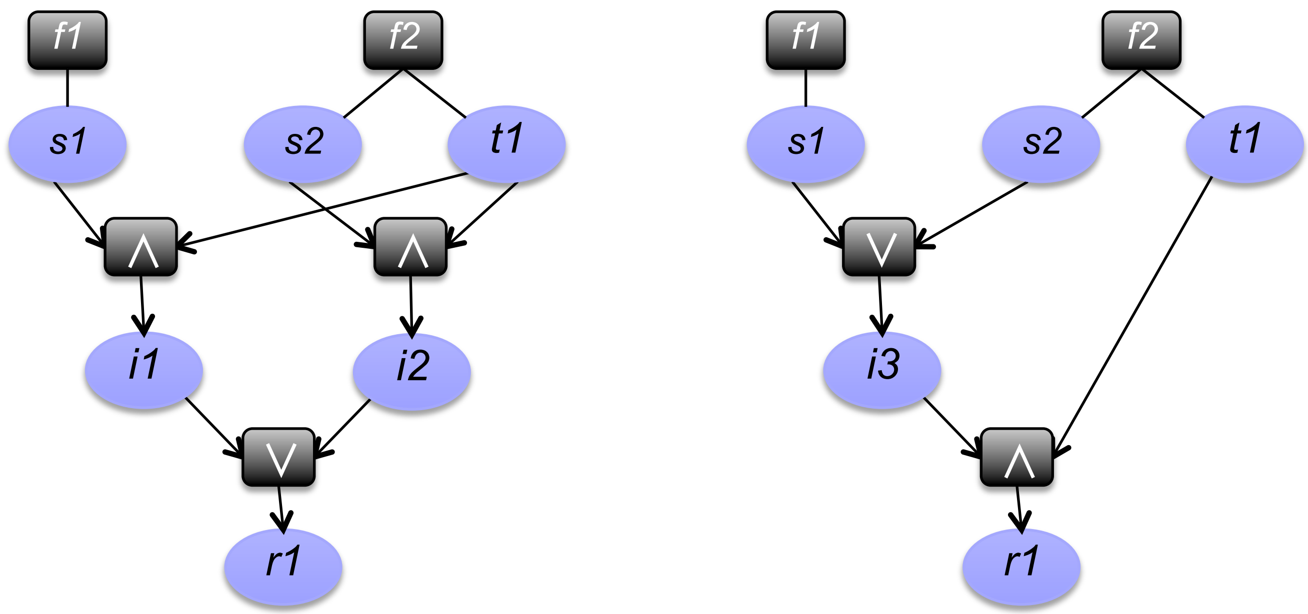
$\propto \sum f_1(s_1)\, f_2(s_2, t_1)\, f^{\wedge}(s_1,t_1,i_1)\, f^{\wedge}(s_2,t_1,i_2)\, f^{\vee}(i_1,i_2,r_1)$

$\propto \sum f_2(s_2, t_1)\, g_1(t_1,i_1)\, f^{AND}(s_2,t_1,i_2)\, f^{OR}(i_1,i_2,r_1)$



An Observation

- AND and OR factors enable reorganization of the network
  - Complexity of the generated network depends on the query plan
    - “Safe plans” generate tree networks – enabing extensional evaluation
  - But a reorganization may not necessarily correspond to a traditional query plan
  - Benefits in looking for optimal reorganization for a given query and dataset
- Efficient inference in presence of special types of factors largely open



Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

PrDB: Representation and Storage

- Underlying representation essentially a factor graph
  - Tuples and factors stored separately in different tables
- Factors can be inserted on any set of random variables
  - Corresponding to tuple existences or attribute values
- **Semantics**: the joint pdf over the random variables is obtained by multiplying all the factors and normalizing
  - No special care taken right now to ensure this is correct
- Allows specifying ***shared factors*** *that apply to groups of tuples, or to all tuples of a relation (schema-level)*

PrDB: Representation and Storage

insert into *S* ***values*** *('s1', 'm', 1)* ***uncertain****('f 0.2; t 0.8');*

S

tid	A	B	e
s1	'm'	1	*π*
s2	'n'	1	*π*

T

tid	B	C	e
t1	*π*	'p'	***t***

fid	rv	pos
f1	s1.e	1
f2	s2.e	1
f3	s1.e	1
f3	t1.B	2

fid	funcid
f1	φ1
f2	φ1
f3	φ2

funcid	func
φ1	{[f] : 0.2, [t] : 0.8}
φ2	{[f, 2] : 0.2, [f, 3] : 0.8, [t, 2] : 0.9, [t, 3] : 0.1}

Data Tables

Uncertainty Parameters (factors)

PrDB: Representation and Storage

insert into *T* ***values*** *('t1', uncertain, 'p');*
insert factor *'f 2 0.2; f 3 0.8; t 2 0.9; t 3 0.1'* ***in*** *S, T* ***on*** *'s1.e', 't1.B';*

S

tid	A	B	e
s1	'm'	1	*Π*
s2	'n'	1	*Π*

T

tid	B	C	e
t1	*Π*	'p'	***t***

Data Tables

fid	rv	pos
f1	s1.e	1
f2	s2.e	1
f3	s1.e	1
f3	t1.B	2

fid	funcid
f1	φ1
f2	φ1
f3	φ2

funcid	func
φ1	{[f] : 0.2, [t] : 0.8}
φ2	{[f, 2] : 0.2, [f, 3] : 0.8, [t, 2] : 0.9, [t, 3] : 0.1}

Uncertainty Parameters (factors)

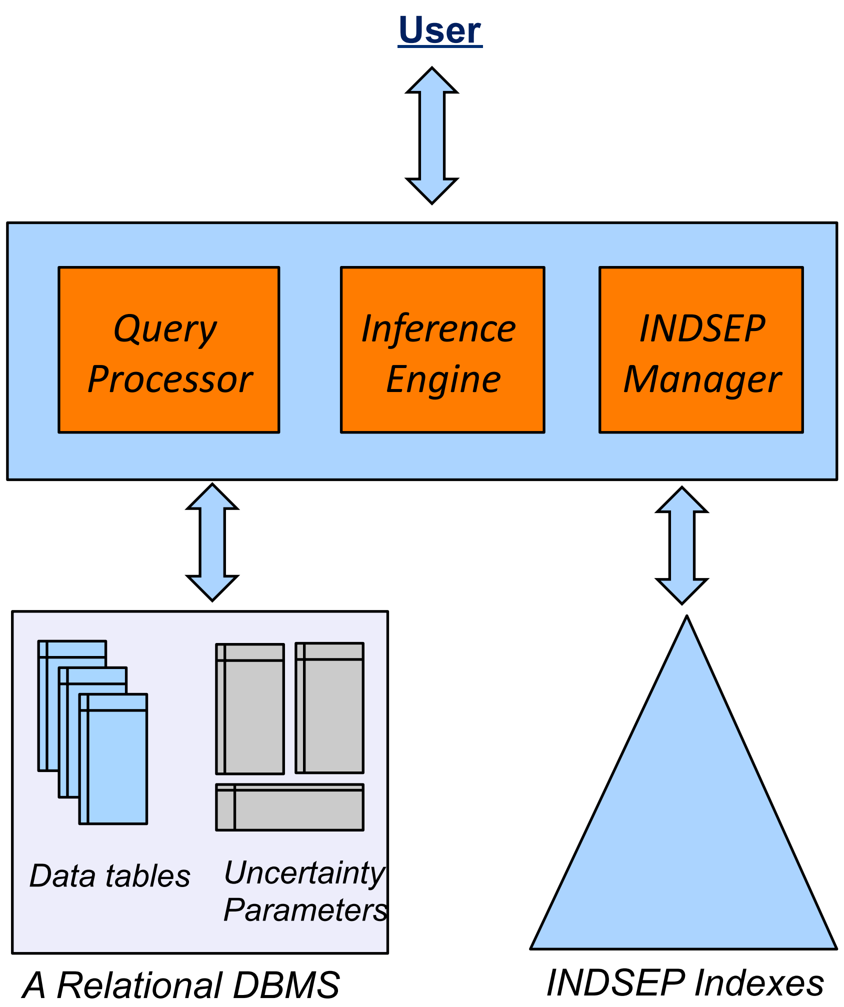
PrDB: Query Processing Overview

No Index on the Data

- Load the base PGM into memory
- Construct an augmented PGM [ICDE'07]
- Use exact or approximate inference [VLDB'08, UAI'09]

INDSEP Indexes Present

- Aggregation or inference queries: Use index directly [SIGMOD'09]
- SQL SPJ Queries [SIGMOD'10]
  - Gather a minimal set of correlations & uncertainties using the index
  - Use exact or approximate inference
  - In some cases, solve using the index



Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

Inference with Shared Factors

AdID	Model	Color	Price
1	Honda	?	$9,000
2	?	Beige	$8,000
3	?	?	$6,000
…	…	…	…
…	…	…	…
…	…	…	…
1000000	?	?	$10,000

Model	Pr(M)
Honda	0.2
Mazda	0.1
…	…

Model	Color	Pr(C|M)
Honda	Beige	0.1
Honda	Red	0.2
…	…	…
Mazda	Beige	0.02

Query: How many "red" cars are for sale ?

- Option 1: "Ground out" (propositionalize) the random variables, and use standard techniques
- Option 2: Directly operate on the shared factors

Inference with Shared Factors

- Option 1: "Ground out" (propositionalize) the random variables, and use standard techniques
  - Would need to create a PGM with a few million nodes
- Option 2: Directly operate on the shared factors
  - Compute a distribution over *makes* for cars with unknown *color* ("Honda" ? "Mazda" ? "Unknown" ?)
  - Use it to estimate the number of red cars
    - E.g. If 1000 Hondas with unknown color, 200 are expected to be red
  - "Lifted inference": Much work in recent years in the ML community
- We developed a general purpose lifted inference technique based on *bisimulation* [VLDB'08, UAI'09]

First-order Lifted Inference

- Huge potential speedups
- … but hard to design general purpose techniques
  - #P-hardness of prob. query evaluation holds with all probabilities = 0.5

R

ID	A	B
1	α	?
2	β	?
3	α	?
4	α	?
5	β	?
…	…	…

A schema-level factor on A and B

A	*B*	*f*
α	0	0.2
α	1	0.8
β	0	0.3
β	1	0.7

A Conjunctive Query: *Compute the prob. that there is a tuple in R with A = α and B = 0*

q :- R(ID, α, 0)

1. *Propositionalizing (grouding out) would take at least O(|R|) time*
2. *However, if we know |R.a = α|, then:*

   answer $= 1 - (1 - 0.2)^{|R.a\, =\, \alpha|}$

   Essentially O(1) time

Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
  - Bisimulation-based Lifted Inference
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

Bisimulation-based Lifted Inference

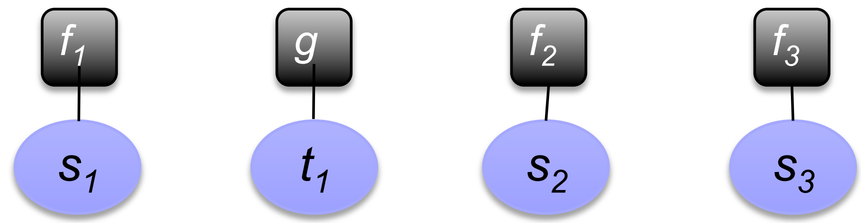
Query: $S \bowtie T$

S

	A	B	prob
s1	'm'	1	0.8
s2	'n'	1	0.8
s3	'o'	1	0.6

T

	B	C	prob
t1	1	'p'	0.5



s1	f_1(s1)
0	0.2
1	0.8

s2	f_2(s2)
0	0.2
1	0.8

s3	f_3(s3)
0	0.4
1	0.6

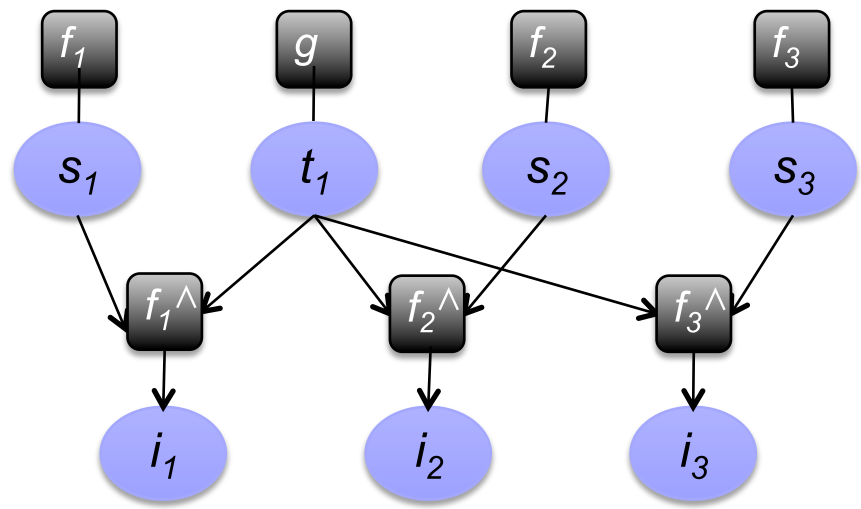
t1	g(t1)
0	0.5
1	0.5

Bisimulation-based Lifted Inference

Query: S ⋈ T

S	A	B	*prob*
s1	'm'	1	0.8
s2	'n'	1	0.8
s3	'o'	1	0.6

T	B	C	*prob*
t1	1	'p'	0.5



Inferences required:

$$\mu_1(i_1) = \sum_{s1,t1} f_1(s_1)\, g(t_1)\, f_1^\wedge(s_1,t_1,i_1)$$

$$\mu_2(i_2) = \sum_{s2,t1} f_2(s_2)\, g(t_1)\, f_2^\wedge(s_2,t_1,i_2)$$

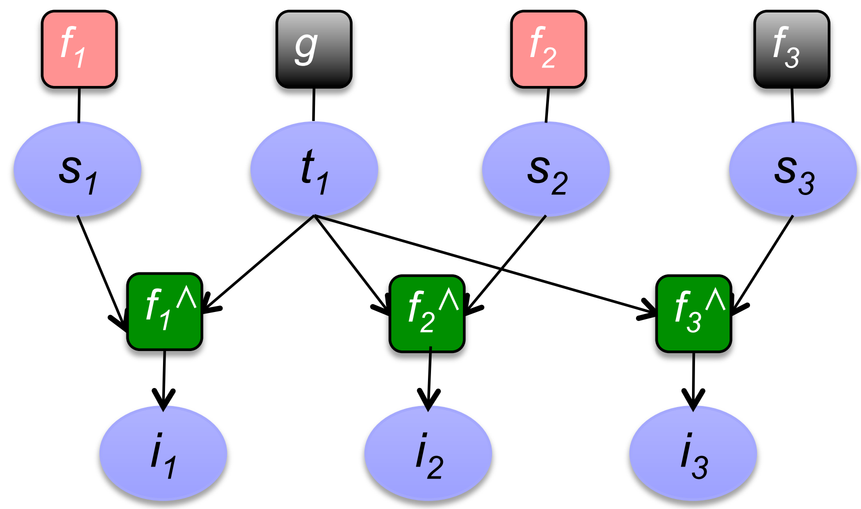
$$\mu_3(i_3) = \sum_{s3,t1} f_3(s_3)\, g(t_1)\, f_3^\wedge(s_3,t_1,i_3)$$

Bisimulation-based Lifted Inference

Query: S ⋈ T

S	A	B	prob
s1	'm'	1	0.8
s2	'n'	1	0.8
s3	'o'	1	0.6

T	B	C	prob
t1	1	'p'	0.5



Inferences required:

$\mu_1(i_1) = \sum_{s1,t1} f_1(s_1)\, g(t_1)\, f_1{}^{\wedge}(s_1,t_1,i_1)$

$\mu_2(i_2) = \sum_{s2,t1} f_2(s_2)\, g(t_1)\, f_2{}^{\wedge}(s_2,t_1,i_2)$

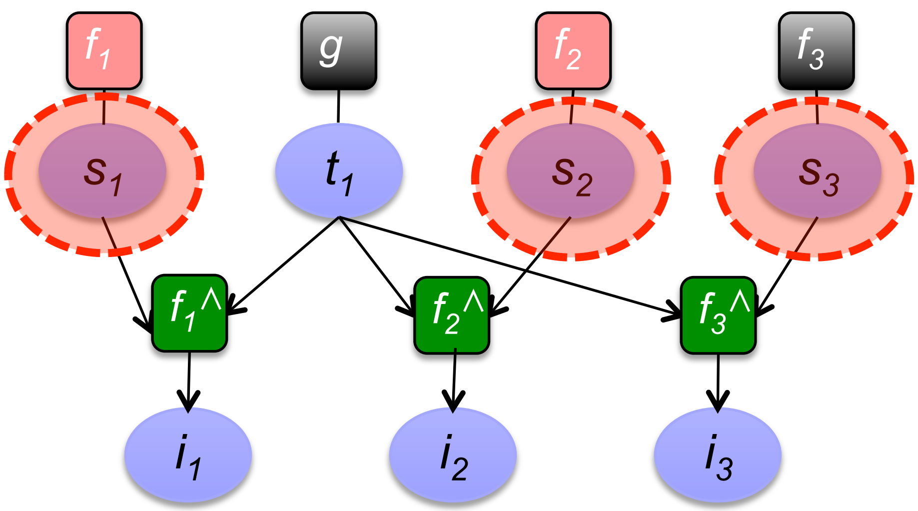
Identical computation Repeated during evaluation

$\mu_3(i_3) = \sum_{s3,t1} f_3(s_3)\, g(t_1)\, f_3{}^{\wedge}(s_3,t_1,i_3)$

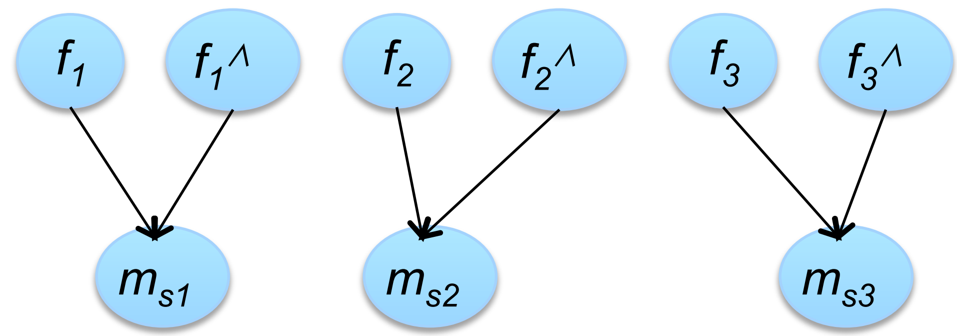
Bisimulation-based Lifted Inference

Step 1: Capture a (simulated) run of variable elimination as a graph

Graphical Model



RV-Elim Graph



$m_{s1}(t_1, i_1) = \sum_{s1} f_1(s_1)\, f_1{}^\wedge(s_1,t_1,i_1)$

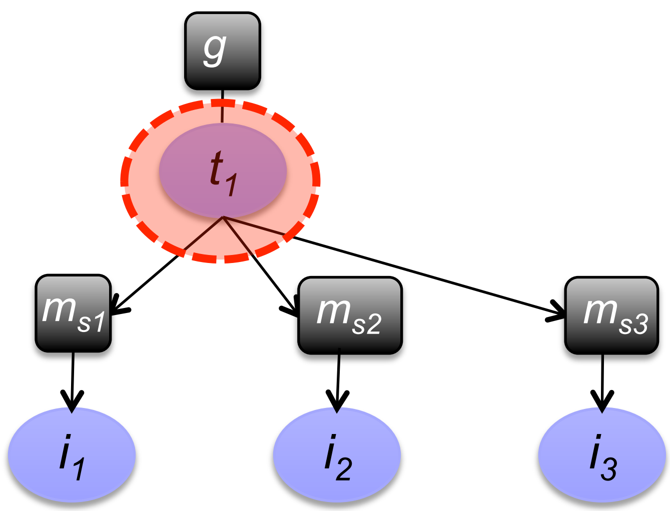
$m_{s2}(t_2, i_2) = \sum_{s2} f_2(s_2)\, f_2{}^\wedge(s_2,t_1,i_2)$

$m_{s1}(t_3, i_3) = \sum_{s3} f_3(s_3)\, f_3{}^\wedge(s_3,t_1,i_3)$

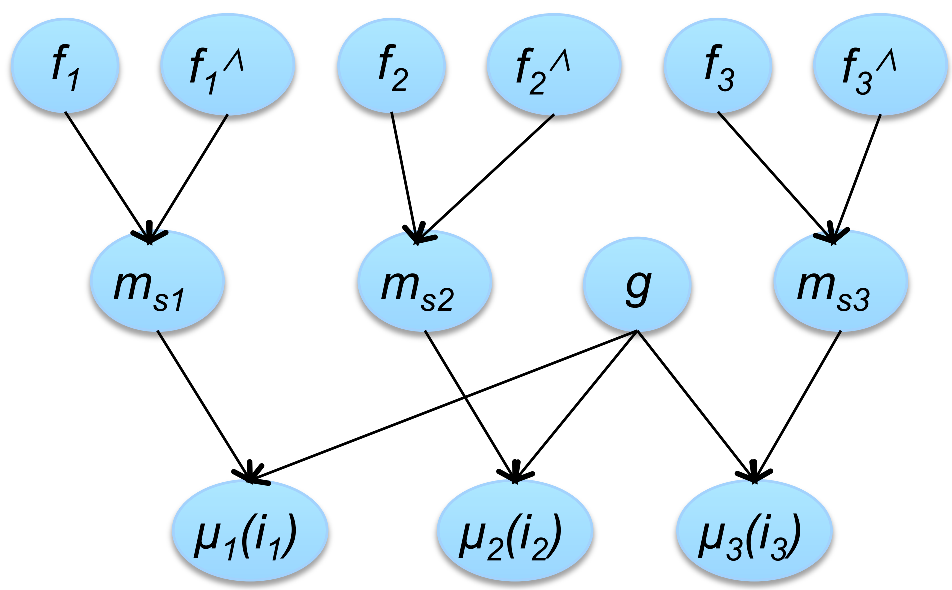
Bisimulation-based Lifted Inference

Step 1: Capture a (simulated) run of variable elimination as a graph

Graphical Model



RV-Elim Graph



$\mu_1(i_1) = \sum_{t1} m_{s1}(t_1,i_1)\ g(t_1)$

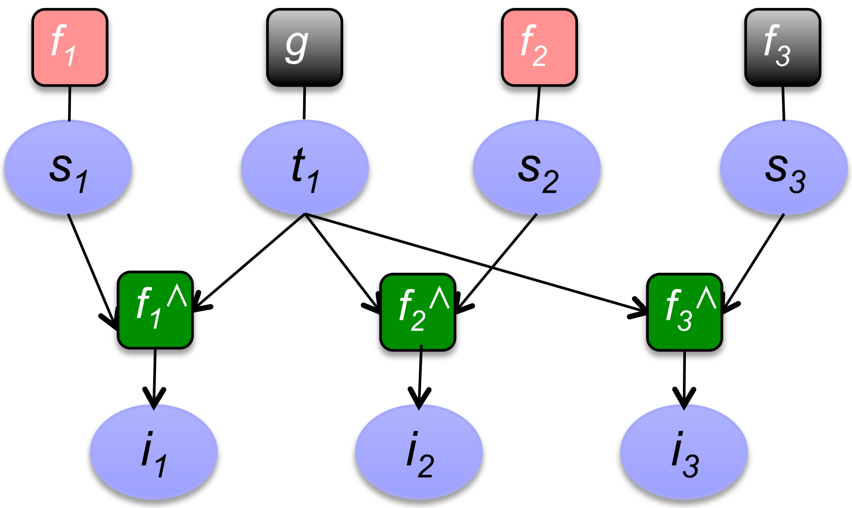
$\mu_2(i_2) = \sum_{t1} m_{s2}(t_1,i_2)\ g(t_1)$

$\mu_3(i_3) = \sum_{t1} m_{s3}(t_1,i_3)\ g(t_1)$

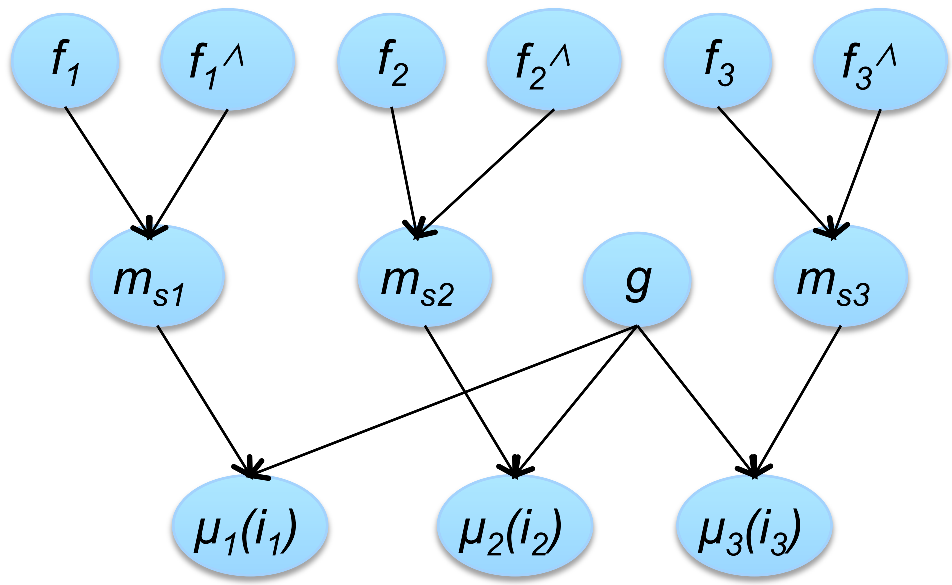
Bisimulation-based Lifted Inference

Step 2: Run *bisimulation* on the RV-Elim graph to identify symmetries

Graphical Model



RV-Elim Graph

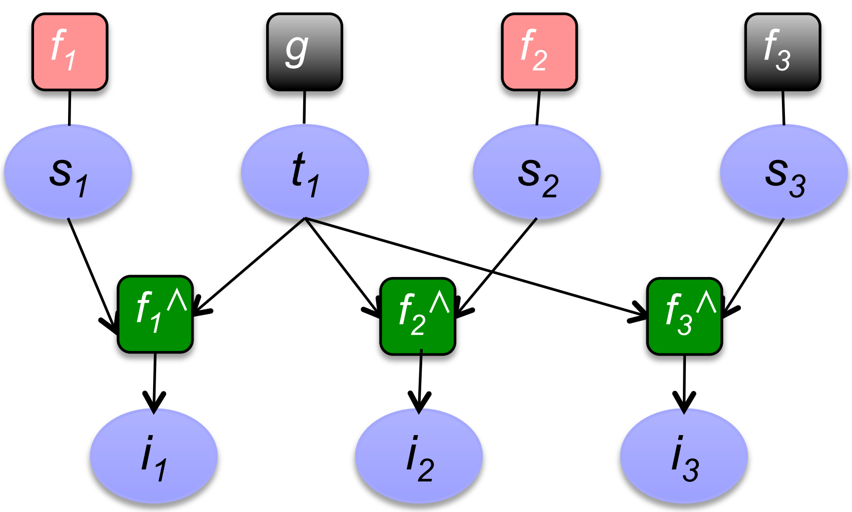


Intuitively, two nodes are bisimilar if
(1) they represent identical factors, and
(2) their parents are identically colored

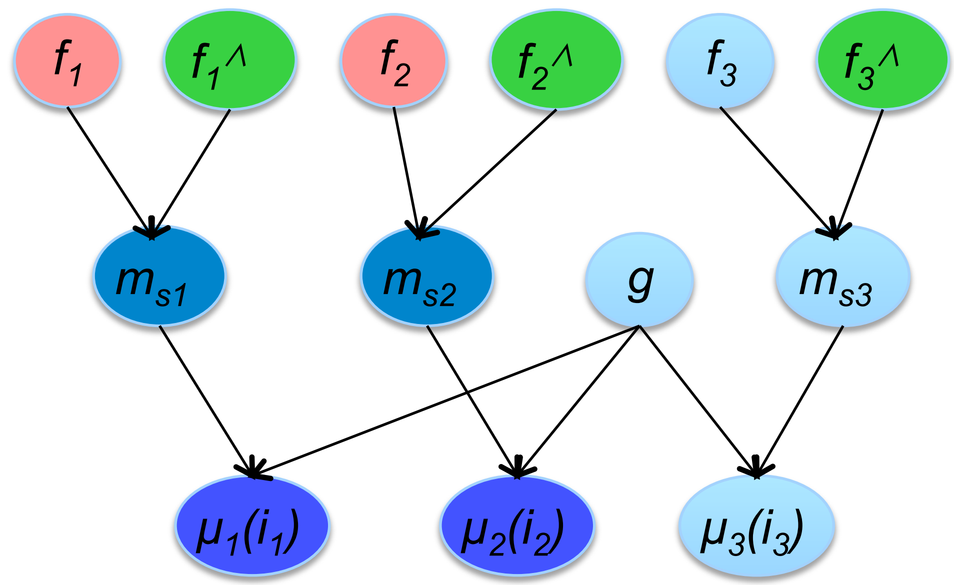
Bisimulation-based Lifted Inference

Step 2: Run *bisimulation* on the RV-Elim graph to identify symmetries

Graphical Model



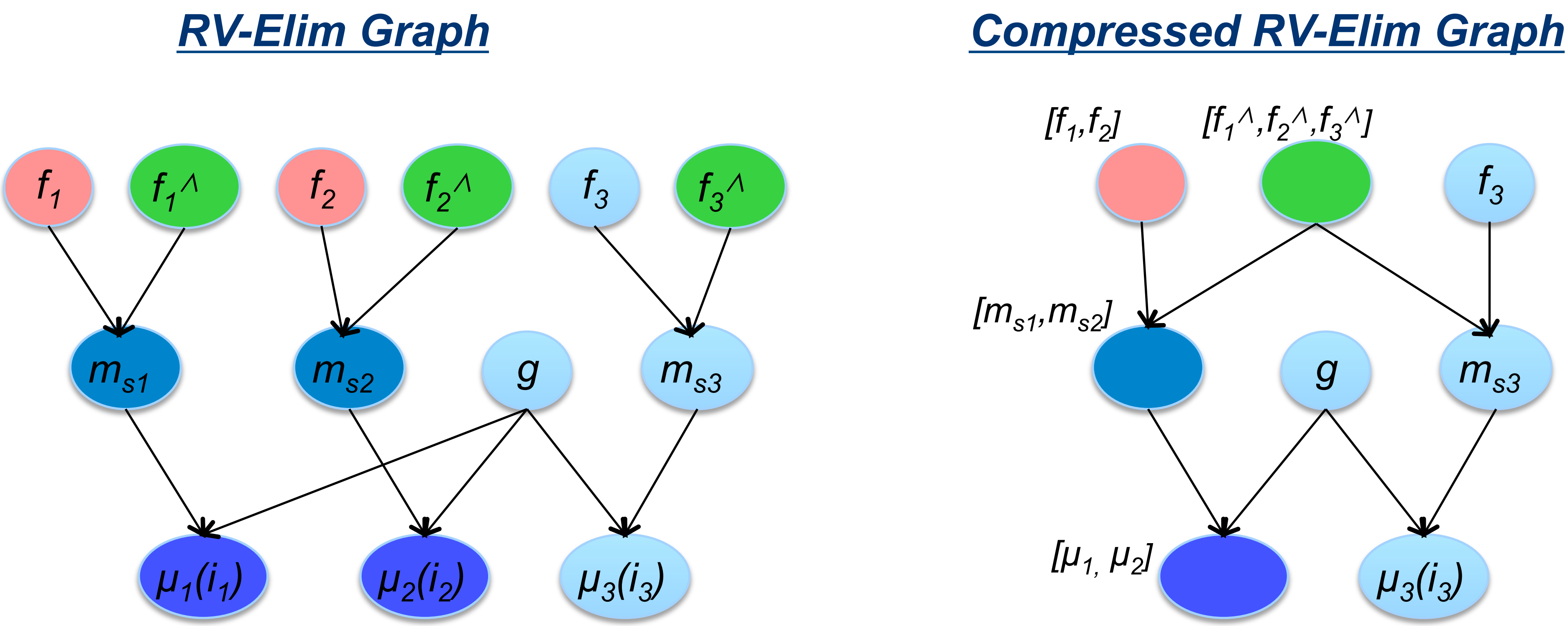
RV-Elim Graph



Intuitively, two nodes are bisimilar if
(1) they represent identical factors, and
(2) their parents are identically colored

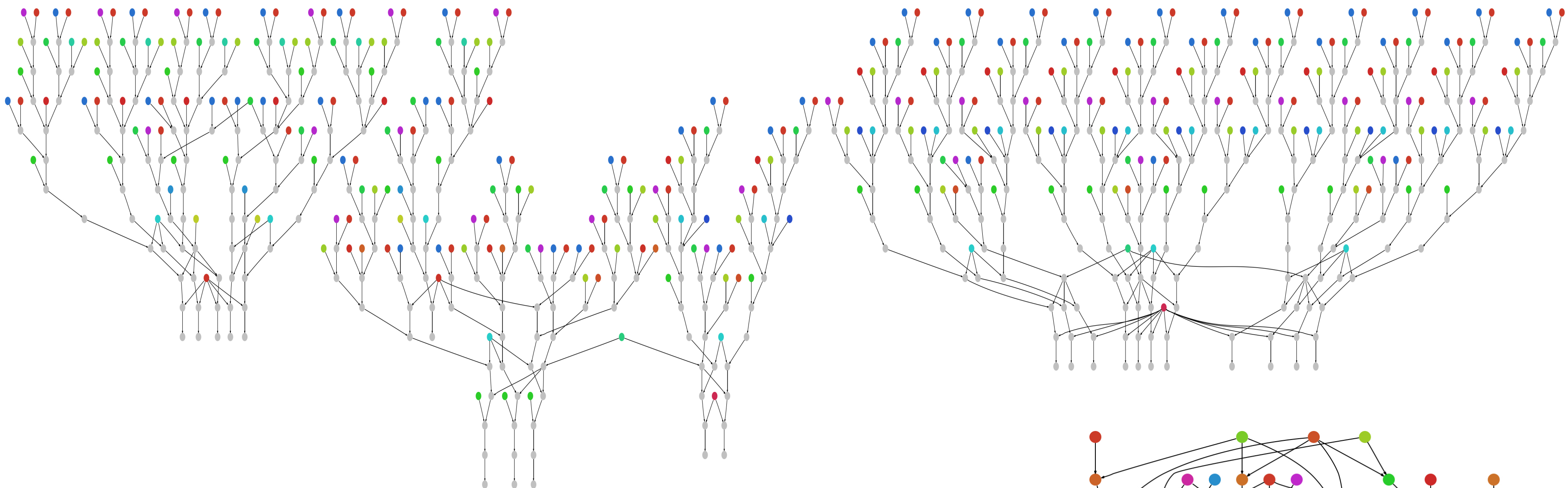
Bisimulation-based Lifted Inference

Step 3: Compress the RV-Elim graph; run inference on compressed graph



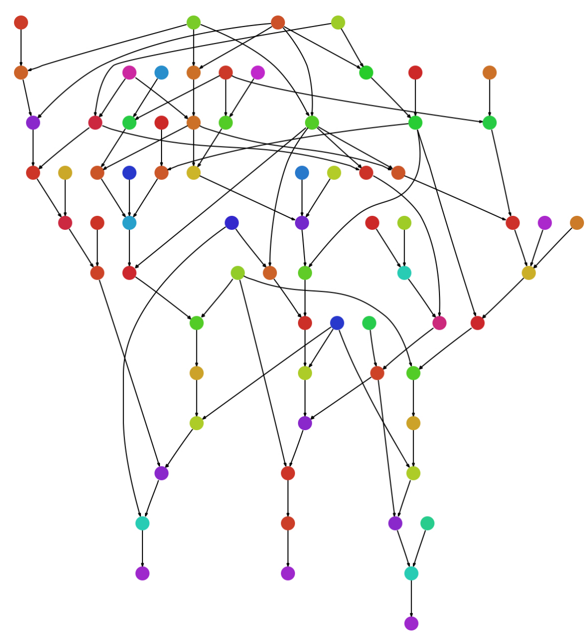
Example RV-Elim Graph

[[3 relation join with 3 tuples each, attribute and tuple uncertainty]]



Original RV-Elim graph, 1170 vertices

Compressed RV-Elim graph, 78 vertices



Bisimulation-based Lifted Inference

- Orders of magnitude performance improvements with symmetry
- Bisimulation can be done in linear time on DAGs
  - Somewhat more involved here
    - Need to keep track of the order in which factors were multiplied
    - Must construct labels on-the-fly as opposed to standard bisimulation
  - Our algorithm runs in $O(|E| \log(D) + |V|)$ time
- Choice of elimination order crucial
  - Dictates the amount of compression possible
  - We choose it by running bisimulation on the graphical model itself
- Our technique works on the ground (propositionalized) model
  - Enables approximations: e.g. allow approximate matches on factors [UAI'09]
- Many open challenges in effectively exploiting symmetry and first order representations

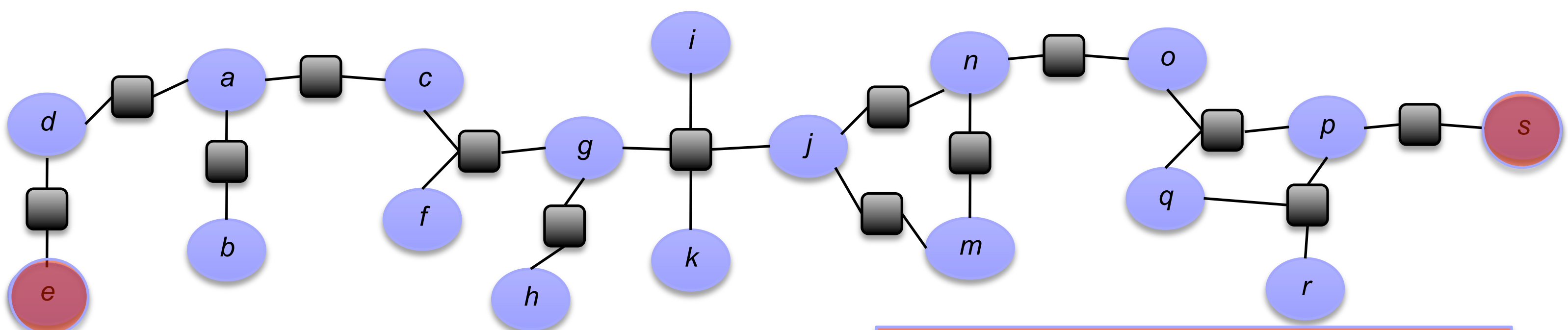
Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

Querying Very Large CPDBs

- Base representation of PGMs can't handle large datasets
  - Queries may only reference a small set of variables
    - Still may need to touch the entire dataset
  - Infeasible to load into memory and operate upon the full PGM

An example PGM



Q1: Need to do an inference operation involving nearly all variables

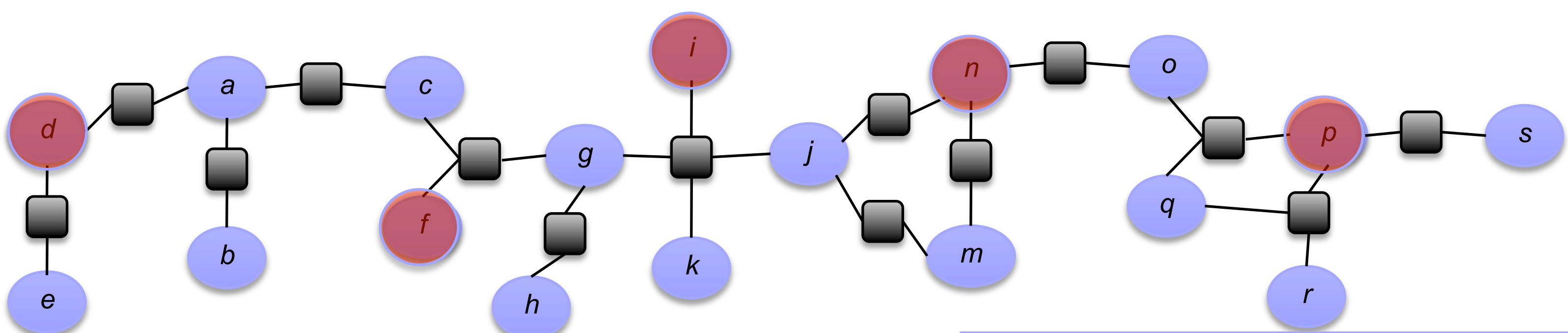
Queries of interest

Q1: How does the value of "s" affect the value "e" ?

Querying Very Large CPDBs

- Base representation of PGMs can't handle large datasets
  - Queries may only reference a small set of variables
    - Still may need to touch the entire dataset
  - Infeasible to load into memory and operate upon the full PGM

An example PGM



Q2: Must compute a potentially large probability distribution: Pr(d, i, f, n, p)

Queries of interest

Q1: How does the value of "n" affect the value "e" ?

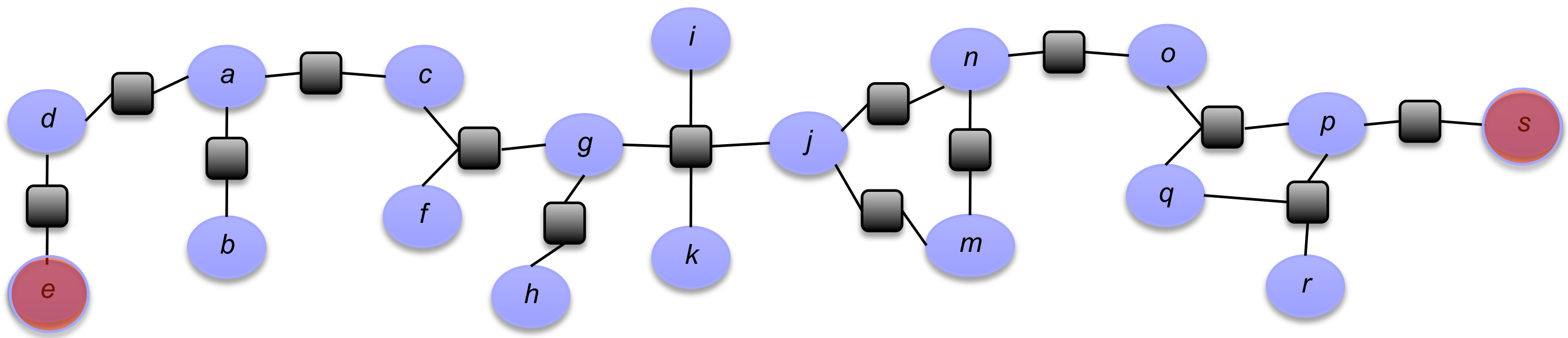
Q2: Compute probability distribution of "d + i + f + n + p"

Querying Very Large CPDBs

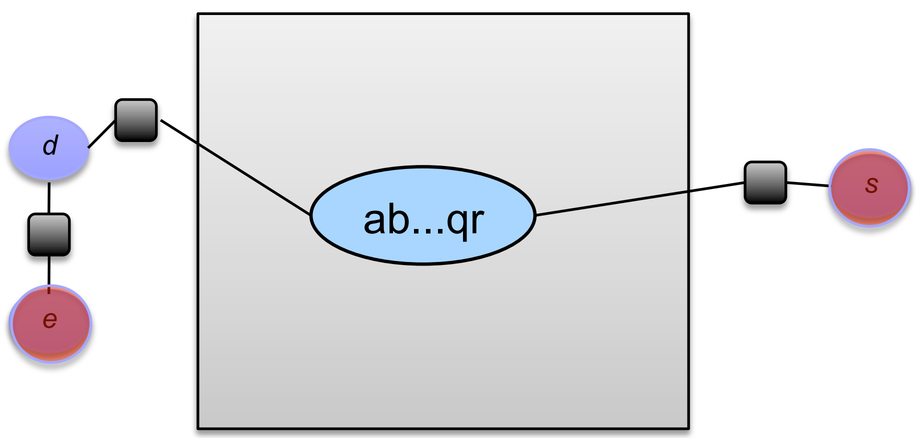
- Base representation of PGMs can't handle large datasets
- Need data structures that:
  - Reuse computation during different inference operations
  - Support updating data as well as uncertainty parameters
  - Minimize the number of variables that need to be accessed
  - Support computation of aggregates and lineage expressions required for SQL query processing
- Some prior techniques *(e.g. junction trees)* help with some of these, but not all

Key Insight

Original PGM



What if we could "shortcut" the in-between nodes ?



Many fewer computations

Can do inference much faster

INDSEP: Overview

- Unclear how to do this on the graphical model directly
- Instead we work with a *junction tree* of the model
  - Essentially a *tree decomposition* of the factor graph, treated as a hypergraph
  - Caveat: Inherit the limitations of the junction tree approach – only works for models with *bounded treewidth*
- INDSEP is a hierarchical data structure over junction tree
  - Built using tree partitioning algorithms
  - Several techniques used to reduce the size of the index

INDSEP: Overview

- Very large speedups for *inference queries,* and for *decomposable aggregate functions (like SUM, MAX)*
- Lineages (boolean formulas) trickier (not decomposable), but similar speedups with more complex algorithms
- Supports a lazy approach for updates
  - Future queries inherit the burden of updating the index
  - Needed because a single update can affect the entire junction tree

Outline

- Probabilistic Databases: Overview, Limitations
- PrDB: Example and Background
- PrDB: Overview
- Inference with Shared Factors
- Indexing Structures for Correlated Databases
- Ongoing and Future Work

Ongoing Work and Open Problems

- Better connections with the work in the ML community
  - Many ML problems and application domains ideal use cases for probabilistic databases
    - Need to scale to large (relational) databases
    - Need support for rich querying over uncertain data
  - Significant overlap in the tools and techniques being developed
  - But many important differences
    - Learning and knowledge transfer equally important there
      - <u>Typical use case for PRMs or MLNs</u>: learn weights/probabilities from a deterministic database, and transfer to other (incomplete) database
    - Not much work in the probabilistic database community

Ongoing Work and Open Problems

- Language constructs and semantics
  - Flexibility in specifying uncertainties at different abstraction levels results in significant interpretation issues
  - *How to resolve conflicting uncertainties ?*
  - *How to keep the semantics simple enough that users can make sense of it ?*

- Efficient algorithms for lifted inference
  - Much work in recent years, but many interesting open problems remain

Ongoing Work and Open Problems

- Querying very large correlated probabilistic databases
  - Our indexing structures inherit the limitations of junction trees
    - Can only handle datasets or queries with low treewidths
  - *How to incorporate approximations into the framework ?*
  - *Lineage formula probability computation especially hard*
    - Computing probabilities of *read-once* lineages easy with tuple independence, but #P-Hard for simplest of correlations
- Uncertain graph data
  - Shared correlations prevalent in settings like social networks, biological networks
  - Compact models of correlations required

Thank You !!

- More details at:

  http://www.cs.umd.edu/~amol/PrDB