Conditioning Probabilistic Databases

Dan Olteanu (Oxford University Computing Laboratory)

Joint work with Christoph Koch (Cornell University Database Systems Group)

Main goals of the MayBMS project

Create a scalable DBMS for uncertain/probabilistic data

1. Representation and storage mechanisms
2. Uncertainty-aware query and data manipulation language
3. Efficient processing techniques for queries and constraints

This talk covers aspects of (3).

MayBMS available at sourceforge.net !

Conditioning c-table-like Probabilistic Databases

Transform a probabilistic database of priors into a posterior probabilistic database.

Example: Probabilistic database representing four *weighted* instances of relation R defining social security numbers and names:

R^1	SSN	NAME
	1	John
	4	Bill

P = .06

R^2	SSN	NAME
	7	John
	4	Bill

P = .24

R^3	SSN	NAME
	1	John
	7	Bill

P = .14

R^4	SSN	NAME
	7	John
	7	Bill

P = .56

Events: A_x = Bill has SSN x; B = SSN is unique in R.

Q_1 = select SSN, **conf()** from R where NAME = 'Bill' group by SSN;
assert SSN→NAME on R;Q_1

$$P(A_4) = .3 \qquad P(A_4 \mid B) = \frac{P(A_4 \land B)}{P(B)} = \frac{.3}{.06+.24+.14} \approx .68$$

Challenges

Conditioning/confidence computation is NP-hard on *succinct* representations.

- No prior work on conditioning probabilistic databases (i.e., on using **assert**)
- Some prior work on confidence computation (MystiQ, Trio, MayBMS, ...)

Exact versus approximate computation.

- Approximation problematic for compositional query languages for probabilistic databases.
  - Introduced errors aggregate and grow.
  - **conf()** used in comparison predicates.

Materialize the (succinct) probabilistic database result of conditioning.

- **assert** is natural for data cleaning under possible worlds semantics.

Our representation system: U-Relational Databases

- Discrete independent (random) variables.
- Representation: U-relations + table W representing distributions.
- The schema of each U-relation consists of
  - a set of column pairs WSD = ($Var \rightarrow Dom$) representing variable assignments,
  - a set of value columns,
  - (a tuple id column).

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

Properties of U-relational databases

- **Complete** representation system for finite sets of possible worlds.
- **Purely relational** representation of uncertainty at **attribute-level**.
- **Efficient relational evaluation** of SPJ queries (without **conf()**).

Our representation system: U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

Our representation system: U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

R^1	SSN	NAME
	1	John
	4	Bill

$P = .2 \cdot .3 = .06$

R^2	SSN	NAME
	7	John
	4	Bill

$P = .8 \cdot .3 = .24$

R^3	SSN	NAME
	1	John
	7	Bill

$P = .2 \cdot .7 = .14$

R^4	SSN	NAME
	7	John
	7	Bill

$P = .8 \cdot .7 = .56$

Our representation system: U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

R^1	SSN	NAME
	1	John
	4	Bill

$P = .2 \cdot .3 = .06$

R^2	SSN	NAME
	7	John
	4	Bill

$P = .8 \cdot .3 = .24$

R^3	SSN	NAME
	1	John
	7	Bill

$P = .2 \cdot .7 = .14$

R^4	SSN	NAME
	7	John
	7	Bill

$P = .8 \cdot .7 = .56$

Our representation system: U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

R^1	SSN	NAME
	1	John
	4	Bill

$\text{P} = .2 \cdot .3 = .06$

R^2	SSN	NAME
	7	John
	4	Bill

$\text{P} = .8 \cdot .3 = .24$

R^3	SSN	NAME
	1	John
	7	Bill

$\text{P} = .2 \cdot .7 = .14$

R^4	SSN	NAME
	7	John
	7	Bill

$\text{P} = .8 \cdot .7 = .56$

Our representation system: U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

R^1	SSN	NAME
	1	John
	4	Bill

$P = .2 \cdot .3 = .06$

R^2	SSN	NAME
	7	John
	4	Bill

$P = .8 \cdot .3 = .24$

R^3	SSN	NAME
	1	John
	7	Bill

$P = .2 \cdot .7 = .14$

R^4	SSN	NAME
	7	John
	7	Bill

$P = .8 \cdot .7 = .56$

Queries on U-Relational Databases

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

$Q_1 =$ select SSN, **conf()** as P from R where NAME $=$ 'Bill' group by SSN;

Q_1	SSN	P
	4	$P(\{b \mapsto 4\})$
	7	$P(\{b \mapsto 7\})$

What makes confidence computation hard?

1. **Succinct** representation of uncertainty.
   - Each tuple in a probabilistic database is associated with a *world-set descriptor* that succinctly encodes the set of worlds containing that tuple.
     World-set descriptor = Conjunction of variable assignments.
     Examples: $\{j \to 1\}, \{j \to 1, b \to 4\}$.
   - Arbitrary combinations of input world-set descriptors produced by query joins.

2. Queries with **projections** can create duplicate answer tuples.
   - Distinct tuples can be associated with **sets** of world-set descriptors.
     Set of world-set descriptors = DNF expression over variable assigments.
     Examples: $\{\{j \to 1\}\}$ and $\{\{j \to 1\}, \{j \to 1, b \to 4\}, \{b \to 7\}\}$.

3. #SAT (Model counting) is #P-hard for **arbitrary DNF expressions**.
   - Model counting is a special case of confidence computation.
   - Arbitrary sets of world-set descriptors can be created by queries.
   - The sets of models of different conjunctions in a DNF expression can overlap and have exponential size.

Knowledge Compilation Techniques to the Rescue

- Useful for compiling formulas into propositional theories with tractable properties, e.g., (#)SAT.
  ROBDDs (Bryant), d-NNFs (Darwiche), and variations thereof.
- Successfully applied to system modelling and verification.

In this paper: ws-sets compiled into **ws-trees**.

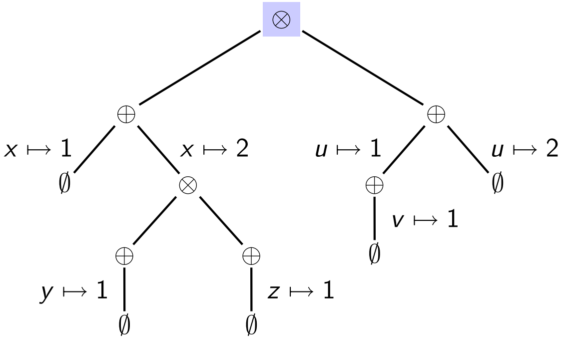
- more succinct than OBDDs and similar to d-NNFs
- structurally limited (trees) and with multistate variables
- ws-sets can be compiled into ws-trees of exponential size
  but like OBDDs tend to behave well in practice

Idea behind ws-tree construction: Given a tuple t with a ws-set S, partition S

- into independent subsets (*exploit contextual independence*)
- by variable elimination (*Davis-Putnam procedure*)

Building ws-trees

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$
Assume $\mathrm{dom}_x = \{1, 2, 3\}$ and $\mathrm{dom}_y = \mathrm{dom}_z = \mathrm{dom}_u = \mathrm{dom}_v = \{1, 2\}$.

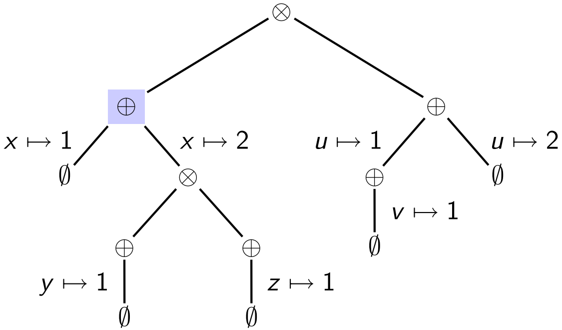


Apply independence partitioning to S:

- left: $\{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}\}$
- right: $\{\{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$.

Building ws-trees

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$
Assume $\mathrm{dom}_x = \{1, 2, 3\}$ and $\mathrm{dom}_y = \mathrm{dom}_z = \mathrm{dom}_u = \mathrm{dom}_v = \{1, 2\}$.

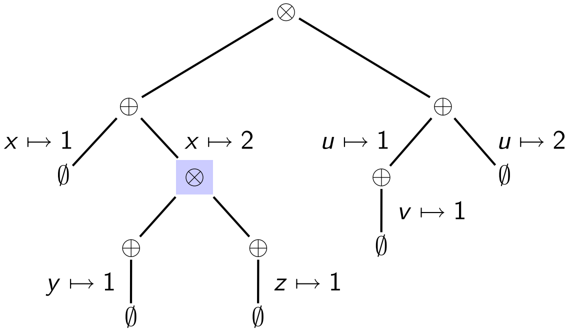


Apply variable elimination to $\{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}\}$.

- left: $x \mapsto 1 : \emptyset$
- right: $x \mapsto 2 : \{\{y \mapsto 1\}, \{z \mapsto 1\}\}$.

Building ws-trees

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$
Assume $\mathrm{dom}_x = \{1, 2, 3\}$ and $\mathrm{dom}_y = \mathrm{dom}_z = \mathrm{dom}_u = \mathrm{dom}_v = \{1, 2\}$.

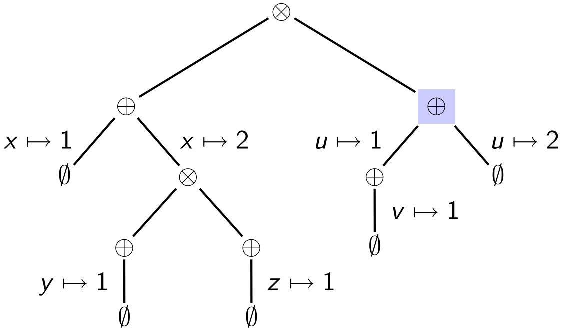


Apply independence partitioning to $\{\{y \mapsto 1\}, \{z \mapsto 1\}\}$.

- left: $\{\{y \mapsto 1\}\}$
- right: $\{\{z \mapsto 1\}\}$

Building ws-trees

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$
Assume $\mathrm{dom}_x = \{1, 2, 3\}$ and $\mathrm{dom}_y = \mathrm{dom}_z = \mathrm{dom}_u = \mathrm{dom}_v = \{1, 2\}$.



Apply variable elimination to $\{\{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$.

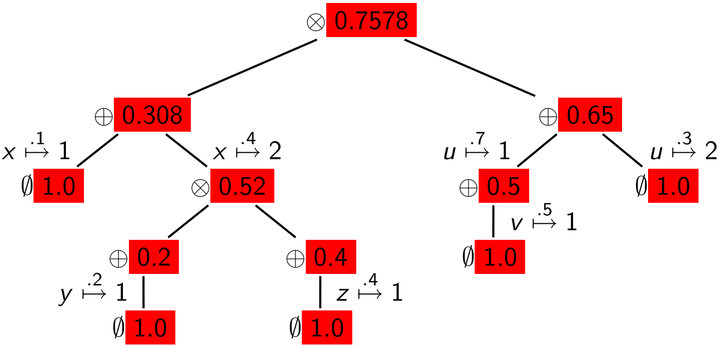
- left: $u \mapsto 1 : \{\{v \mapsto 1\}\}$
- right: $u \mapsto 2 : \emptyset$

Confidence computation using ws-trees

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$

Assume: $\mathrm{dom}_x = \{1, 2, 3\}$ and $\mathrm{dom}_y = \mathrm{dom}_z = \mathrm{dom}_u = \mathrm{dom}_v = \{1, 2\}$.

$$x \overset{.1}{\mapsto} 1, x \overset{.4}{\mapsto} 2, y \overset{.2}{\mapsto} 1, z \overset{.4}{\mapsto} 1, u \overset{.7}{\mapsto} 1, u \overset{.3}{\mapsto} 2, v \overset{.5}{\mapsto} 1.$$



$$P(S) = 0.7578.$$

Conditioning using ws-trees

Assert constraint ϕ on U-relational database U.

1. Compute the ws-set S that describes the worlds in which ϕ holds.
   Evaluation of Boolean query for ϕ followed by complement with W.
2. Compile S into a ws-tree T.
3. Renormalize T such that the probabilities of all remaining worlds sum up to 1.
   Introduce new variables to reflect renormalization.
4. Update the ws-descriptors WSD in U according to renormalized T.
   While traversing T, remove from WSD the encountered variables and add the newly created ones.

The last three steps can be done together and T need not be materialized.

Data cleaning example: Evaluate

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

Keep only those worlds that satisfy the key constraint on R:

assert SSN$\rightarrow$ NAME on R;

Expressed as a Boolean query as a complement of $\pi_{\emptyset}(R \bowtie_{\phi} R)$ where $\phi := (1.SSN = 2.SSN \wedge 1.NAME \neq 2.NAME)$. On U-relation U_R,

$$\pi_{WSD}(U_R \bowtie_{\phi \wedge 1.WSD \text{ consistent with } 2.WSD} U_R).$$

Result consists of WSD $\{j \mapsto 7, b \mapsto 7\}$. Its complement with the (entire) world-set given by W is:

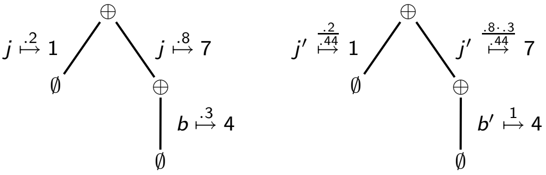
- $\{\{j \mapsto 1\}, \{j \mapsto 7, b \mapsto 4\}\}$, or (equivalently)
- $\{\{b \mapsto 4\}, \{b \mapsto 7, j \mapsto 1\}\}$.

Data cleaning example: Compile and Renormalize

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill

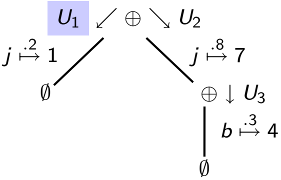
SSN$\rightarrow$NAME holds in the worlds defined by $S = \{\{j \mapsto 1\}, \{j \mapsto 7, b \mapsto 4\}\}$.
Compile S into a ws-tree and renormalize the latter.

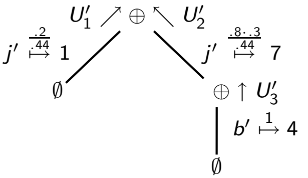


Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill





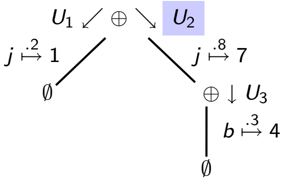
The U-relation tuples to be conditioned are passed down the ws-tree:

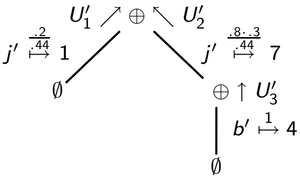
$U_1 = j \mapsto 1 : U_R$	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 1, b \mapsto 4\}$	4	Bill
	$\{j \mapsto 1, b \mapsto 7\}$	7	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill





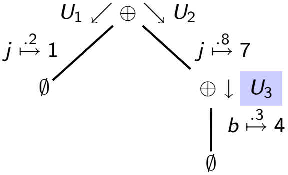
The U-relation tuples to be conditioned are passed down the ws-tree:

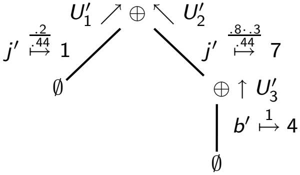
$U_2 = j \mapsto 7 : U_R$	WSD	SSN	NAME
	$\{j \mapsto 7\}$	1	John
	$\{j \mapsto 7, b \mapsto 4\}$	4	Bill
	$\{j \mapsto 7, b \mapsto 7\}$	7	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill





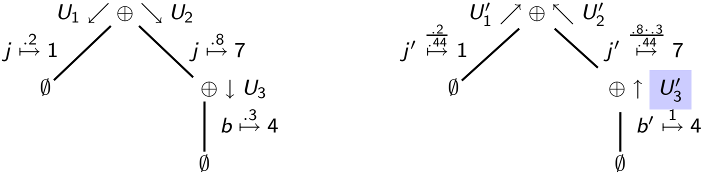
The U-relation tuples to be conditioned are passed down the ws-tree:

$U_3 = b \mapsto 4 : U_2$	WSD	SSN	NAME
	$\{j \mapsto 7, b \rightarrow 4\}$	1	John
	$\{j \mapsto 7, b \mapsto 4\}$	4	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill



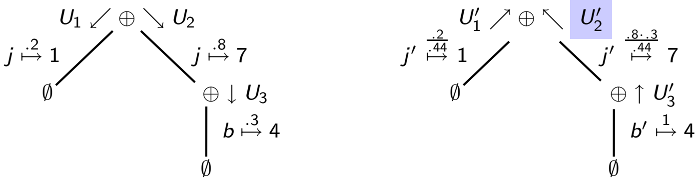
Replace old variables by new variables in the U-relations to be pushed up the normalized ws-tree:

$U_3' =$ Replace b by b' in U_3	WSD	SSN	NAME
	$\{j \mapsto 7, b' \mapsto 4\}$	1	John
	$\{j \mapsto 7, b' \mapsto 4\}$	4	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill



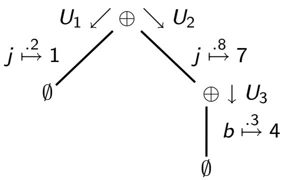
Replace old by new variables in the U-relation tuples to be pushed up the normalized ws-tree:

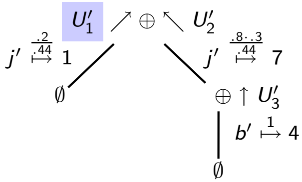
U'_2 = Replace j by j' in U'_3	WSD	SSN	NAME
	$\{j' \mapsto 7, b' \mapsto 4\}$	1	John
	$\{j' \mapsto 7, b' \mapsto 4\}$	4	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill





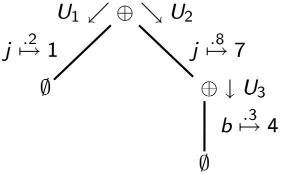
Replace old by new variables in the U-relation tuples to be pushed up the normalized ws-tree:

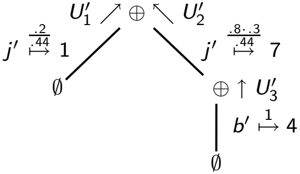
$U_1' =$ Replace j by j' in U_1	WSD	SSN	NAME
	$\{j' \mapsto 1\}$	1	John
	$\{j' \mapsto 1, b \mapsto 4\}$	4	Bill
	$\{j' \mapsto 1, b \mapsto 7\}$	7	Bill

Data cleaning example: Update the Database

W	Var	Dom	P
	j	1	.2
	j	7	.8
	b	4	.3
	b	7	.7

U_R	WSD	SSN	NAME
	$\{j \mapsto 1\}$	1	John
	$\{j \mapsto 7\}$	7	John
	$\{b \mapsto 4\}$	4	Bill
	$\{b \mapsto 7\}$	7	Bill





The U-relational database after conditioning (b' and j are useless and removed):

W'	Var	Dom	P
	b	4	.3
	b	7	.7
	j'	1	$.2/.44$
	j'	7	$.8 \cdot .3/.44$

$U_R' = U_1' \cup U_2'$	WSD	SSN	NAME
	$\{j' \mapsto 1\}$	1	John
	$\{j' \mapsto 7, b' \mapsto 4\}$	1	John
	$\{j' \mapsto 1, b \mapsto 4\}$	4	Bill
	$\{j' \mapsto 1, b \mapsto 7\}$	7	Bill
	$\{j' \mapsto 7, b' \mapsto 4\}$	4	Bill

Experiments

Tuple-independent TPC-H Data

Queries

1. **select distinct** true **from** customer c, orders o, lineitem l **where** c.mktsegment = 'BUILDING' **and** c.custkey = o.custkey and o.orderkey = l.orderkey **and** o.orderdate > '1995-03-15'
2. **select distinct** true **from** lineitem **where** shipdate **between** '1994-01-01' **and** '1996-01-01' **and** discount **between** '0.05' **and** '0.08' **and** quantity < 24

Query	Size of ws-desc.	TPC-H Scale	#Input Vars	Size of ws-set	User Time(s)
Q_1	3	0.01	77215	9836	5.10
		0.05	382314	43498	99.76
		0.10	765572	63886	356.56
Q_2	1	0.01	60175	3029	0.20
		0.05	299814	15545	8.24
		0.10	600572	30948	33.68

Tractable cases of query evaluation on probabilistic databases beyond safe plans:

- Using OBDDs for Efficient Query Evaluation on Probabilistic Databases.
  O. and Huang. In Proc. SUM 2008.
- Lazy versus Eager Query Plans for Tuple-Independent Probabilistic Databases.
  O., Huang, and Koch. 2008.

#P-hard cases

Input: ws-sets similar to those associated with the answers of non-safe Boolean queries on probabilistic databases.

Compared agorithms for confidence computation

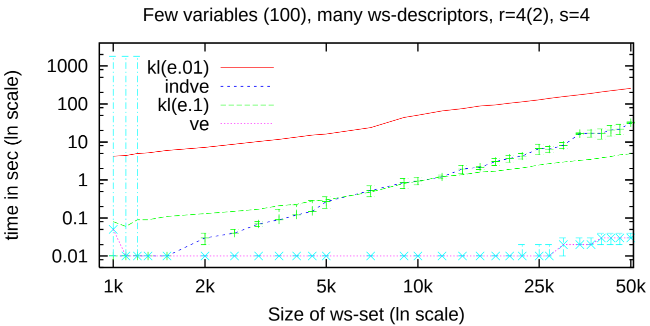
- INDVE: independence partitioning and variable elimination
- VE: only variable elimination
- KL: (adapted) optimal Monte Carlo simulation based on Karp-Luby FPRAS for DNF counting
  Given a DNF formula with m clauses, compute an (ϵ, δ)-approximation $\hat{c}$ of the number of solutions c of the DNF formula such that

  $$\Pr[|c - \hat{c}| \leq \epsilon \cdot c] \geq 1 - \delta$$

  for any given $0 < \epsilon < 1$, $0 < \delta < 1$. It does so within $\lceil 4 \cdot m \cdot \log(2/\delta)/\epsilon^2 \rceil$ iterations of an efficiently computable estimator.

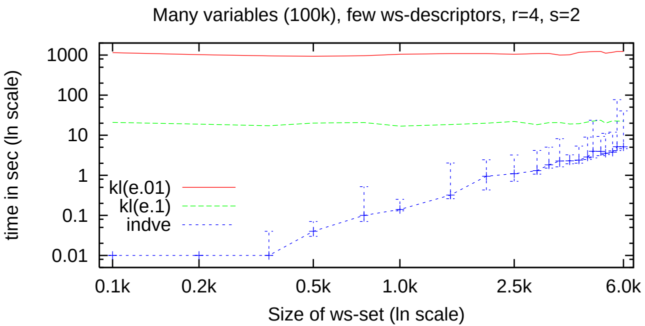
INDVE is now part of the MayBMS engine!

#variables and #wsds differ by orders of magnitude



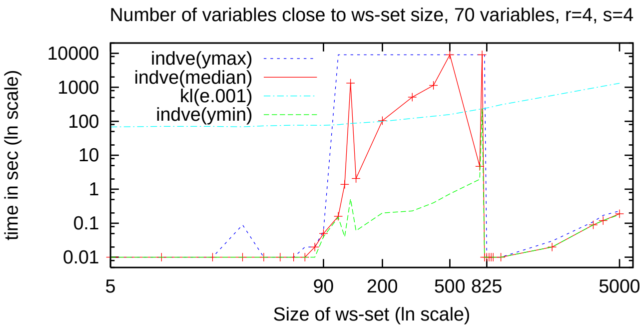
r = domain size of variables; s = size of wsds = #joins used to produce them.

#variables and #wsds differ by orders of magnitude



r = domain size of variables; s = size of wsds = #joins used to produce them.

#variables and #wsds are close: Easy-hard-easy pattern

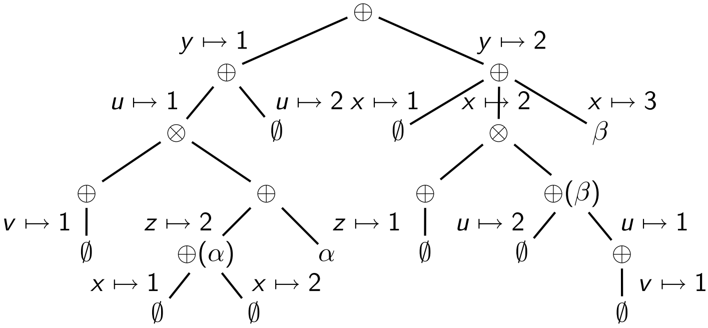


Known that the computation becomes harder in this case. The hard area is smaller for SAT than for #SAT.

Thanks!

Order of Variable Elimination Matters!

$S = \{\{x \mapsto 1\}, \{x \mapsto 2, y \mapsto 1\}, \{x \mapsto 2, z \mapsto 1\}, \{u \mapsto 1, v \mapsto 1\}, \{u \mapsto 2\}\}$
Assume $\text{dom}_x = \{1, 2, 3\}$ and $\text{dom}_y = \text{dom}_z = \text{dom}_u = \text{dom}_v = \{1, 2\}$.



Different ws-tree for the same ws-set S!