Incomplete and Probabilistic Data Management
FOX Training, October 2009, Dortmund

Dan Olteanu, Oxford University Computing Laboratory

Purpose of this tutorial

1. How to represent incomplete/probabilistic data?
   1. Relational vs. XML data
      The relational case has received much more attention
   2. Incomplete vs. probabilistic data
      Incompleteness modeled by sets of possible instances, instead of just one complete instance
      (Discrete) probability distributions over possible instances
   3. Finite vs. infinite sets of possible instances

2. How to query such data?
   1. Query language
      Conjunctive queries, extensions with uncertainty-aware query constructs
   2. Exact vs. approximate query evaluation
      Complexity, dichotomy results, evaluation techniques
   3. Query optimization

Left out in this tutorial: Uncertainty-aware query languages, constraint processing, probabilistic streams, uncertain data mining, incomplete XML data, ...

Tutorial Outline

The plan: tutorial of 180 minutes (including exercises)

1. Incomplete relational data (70 min.)
2. Probabilistic relational data (90 min.)
3. Probabilistic XML data (20 min.)

1. Incomplete Relational Data

Outline of Part 1

- Sources of incompleteness
- Two approaches to deal with incompleteness
- Representation systems for incomplete data
  - Codd tables, v-tables, c-tables, or-sets, world-set decompositions
- Query evaluation on representation systems
- Expressiveness of world-set decompositions
- Complexity of decision problems

Sources of Incompleteness

Single-source problems

- schema level (lack of integrity constraints, poor schema design)
  uniqueness, referential integrity
- instance level (data entry errors)
  misspellings, redundancy/duplicates, contradictory values

Multi-source problems

- schema level (heterogeneous data models and schema design)
  naming and structural conflicts
- instance level (overlapping, contradicting, and inconsistent data)
  inconsistent aggregating, inconsistent timing

Sources of Incompleteness

Single-source problems at schema level

Scope	Problem	Dirty Data	Remarks
Attribute	illegal	bdate=30.13.70	out-of-range value
Record	dependency	age=22, bdate=12.02.70	age=now - bdate
Record type	uniqueness	emp1(SSN=1),emp2(SSN=1)	SSN is unique
Source	ref. integrity	emp(SSN=007)	007 not defined

Sources of Incompleteness

Single-source problems at instance level

Scope	Problem	Dirty Data
Attribute	missing value	null
	misspelling	Lizpig
	cryptic values	DB Prog.
	embedded values	name="Joe New York"
	misfielded values	city=Germany
Record	dependencies	city=SB, code=85764
Record type	transpositions	"D.Olteanu", "Olteanu D."
	duplicates	emp("Dan Olteanu"), emp("D.Olteanu")
	contradictions	emp(Olteanu,Koch), emp(Olteanu,Bock)

How to Cope with Incompleteness? (Approach 1)

Remove all instances (= worlds) that do not satisfy particular criteria.
The hope is to get one (clean) instance.
Data Cleaning deals with detecting and removing errors and inconsistencies from data in order to improve the quality of the data.
It usually consists of the following steps:

1. Data analysis (to detect the kind of occurring errors)
   data mining
2. definition of transformation and mapping rules
   declarative SQL-based languages
3. verification
4. transformation
5. backflow of cleaned data

Data cleaning is a complex semiautomatic approach to deal with incomplete data.

How to Cope with Incompleteness? (Approach 2)

Complementary approach: Provide support for

1. efficient (=succinct) representation of (possibly infinte) sets of worlds.
2. define processing (query evaluation, dependency chasing) on such succinct representations.

We further discuss this approach.

Example of Incomplete Information

Persons	Name	Salary	Room	Phone
	DAO	40K	228	?
	LRA	10K	?	?
	CEK	?	226	57328

? usually represented as **null** value in existing RDBMSs

SQL supports NULL values with constructs like IS (NOT) NULL.

Compare the answers to the following two queries
Q_1: SELECT FROM Persons WHERE Room $>$ 226;
Q_2: SELECT FROM Persons WHERE Room $>$ 226 OR room IS NULL;

Example of Incomplete Information

There are different types of nulls (?).

1. existing unknown values, e.g., DAO's phone or CEK's salary
2. nonexisting values, e.g., LRA's phone
3. no information is known about, e.g., LRA's room number

Persons	Name	Salary	Room	Phone
	DAO	40K	228	?
	LRA	10K	?	
	CEK	?	226	57328

We consider next nulls of the first kind.

Completeness versus Incompleteness

A relation with null values encodes a set of possible *worlds*.

Persons	Name	Salary	Room	Phone
	DAO	40K	228	57332
	LRA	10K	MPIRS-1	
	CEK	400K	226	57328

Persons	Name	Salary	Room	Phone
	DAO	40K	228	57332
	LRA	10K	MPIRS-2	
	CEK	500K	226	57328

.. and so on.

There is an infinite amount of possible worlds!!!

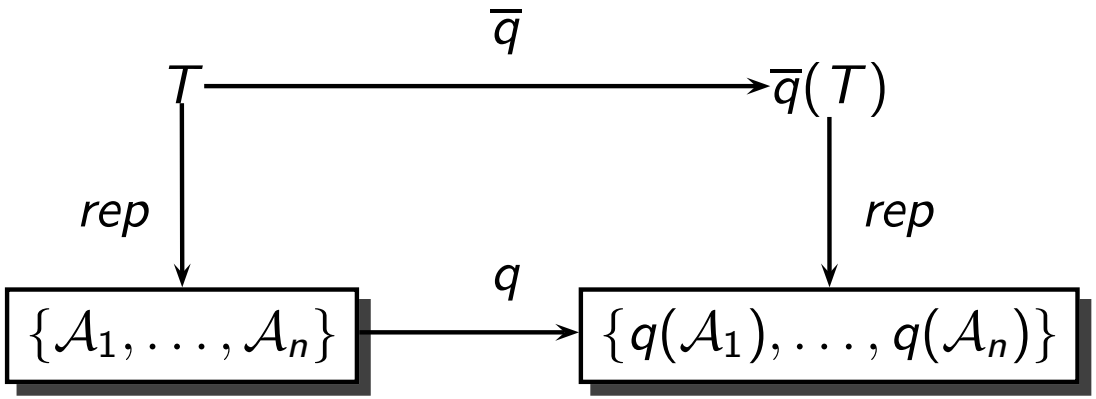
Represent intensionally the set of possible worlds

What is a representation system? [IL84,AHV95]

System to represent set of alternatives or *possible worlds*.

- World = (complete) database.
- Representation T (usually called *Table*)
- Function *rep* mapping T to the set of *possible worlds*.

Query evaluation under *possible world semantics*



Strong Representation Systems [IL84,AHV95]

Language $\mathcal{L}$ (e.g., relational algebra) and *table* T with $rep(T)$

- For a query $q \in \mathcal{L}$, collect the set of possible answers

$$\boxed{q(rep(T)) = \{q(I) \mid I \in rep(T)\}}$$

- *represent*! $q(rep(T))$ as a table $\overline{q}(T)$

$$\boxed{rep(\overline{q}(T)) = q(rep(T))}$$

If T is any table in a representation system τ and q any query in $\mathcal{L}$, then

$$\boxed{\tau \text{ is a } \textit{strong representation system} \text{ for } \mathcal{L}}$$

Weak Representation Systems [IL84,AHV95]

$\mathcal{L}$-Equivalence $\equiv_{\mathcal{L}}$ of Incomplete Databases
Language $\mathcal{L}$, two incomplete databases $\mathcal{I}$ and $\mathcal{J}$.

$$\boxed{\mathcal{I} \equiv_{\mathcal{L}} \mathcal{J} \Leftrightarrow \forall q \in \mathcal{L} : \bigcap\{q(I) \mid I \in \mathcal{I}\} = \bigcap\{q(J) \mid J \in \mathcal{J}\}}$$

$\bigcap\{q(J) \mid J \in \mathcal{J}\}$ is the *certain* answer (or the set of *sure* answer tuples)
$\mathcal{I}$ and $\mathcal{J}$ are equivalent if all we can ask for is the certain answer of $\mathcal{L}$-queries.

If T is any table of a representation system τ and q any query in $\mathcal{L}$, then

$$\boxed{\tau \text{ is a } \textit{weak representation system} \text{ for } \mathcal{L} \Leftrightarrow \mathit{rep}(\overline{q}(T)) \equiv_{\mathcal{L}} q(\mathit{rep}(T))}$$

Corollary: If a system is strong for $\mathcal{L}$, then it is also weak for $\mathcal{L}$.

(Codd) Tables

- Codd tables = Finite relations, where tuples can contain variables
- A variable can occur at most once per entire table
- A Codd table T represents the incomplete database (set of possible worlds)

$$\boxed{rep(T) = \{\nu(T) \mid \nu \text{ is a valuation of the variables in } T\}}$$

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

contains

R	A	B	C
	0	1	2
	2	0	1
	2	0	0

R	A	B	C
	0	1	2
	3	0	1
	2	0	5

$\ldots$

Querying Codd tables: Selection

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

$\xrightarrow{\sigma_{A=3}(R)}$

R	A	B	C
	3	z	1

R	A	B	C

There is no Codd table representing the set of all possible answers!
But there is a (empty) Codd table representing the *certain* answer!

Codd tables form no strong representation system for selection

Querying Codd tables: Projection

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

$\xrightarrow{\pi_A(R)}$

$\pi_A(R)$	A
	0
	y
	2

Codd tables form a strong representation system for projection

Querying Codd tables: Product and Join

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	D
	0
	1

$\xrightarrow{R \times S}$

$R \times S$	A	B	C	D
	0	1	x	0
	0	1	x	1
	y	z	1	0
	y	z	1	1
	2	0	v	0
	2	0	v	1

A variable can appear only once in a Codd table!

Codd tables form no strong representation system for product and join

Querying Codd tables: Union

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	A	B	C
	1	1	w

$\xrightarrow{R \cup S}$

$R \cup S$	A	B	C
	0	1	x
	y	z	1
	2	0	v
	1	1	w

A variable can appear only once in a Codd table!

Codd tables form a strong representation system for union

Querying Codd tables: Difference

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

S	A	B	C
	2	0	0

$\xrightarrow{R-S}$

$R-S$	A	B	C
	0	1	x
	y	z	1
	2	0	?

The value of ? can be anything but 0!

Codd tables form no strong representation system for difference

Certain Answers for Codd tables

For a table T and a query q, the certain answer is

$$\boxed{sure(q, T) = \bigcap\{q(I) \mid I \in rep(T)\}}.$$

- Sure facts appear in the answer for *every* possible world.
- Compute $sure(q, T)$ by dropping all tuples with variables in $q(rep(T))$.
- For our Codd table T, $sure(\sigma_{A=3}(R), T) = \emptyset$, thus representable as Codd table!
- Representing *only* the sure answer tuples is not sufficent!

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

- Consider $q = \sigma_{A=2}(R)$ and $q' = \pi_{AB}(R)$
- Then, $sure(q, T) = \emptyset \Rightarrow q'(sure(q, T)) = \emptyset$
- But, $sure(q'(q(rep(T)))) = \{(2, 0)\} \neq \emptyset$
- $\Rightarrow$ non-compositional query semantics!

How Weak are Codd tables?

R	A	B	C
	0	1	x
	y	z	1
	2	0	v

- Consider again $q = \sigma_{A=2}(R)$ and $q' = \pi_{AB}(R)$
- Choose projection $\overline{q}' = q'$ and selection $\overline{q}_\theta$ such that $\overline{q}_\theta(T) = \{t \mid t \in T, \forall \text{ valuations of vars in t } \mu : \theta(\mu(t))\}$
- Then, $\overline{q}(T) = \{(2, 0, v)\}$ and $\overline{q' \circ q}(T) = \{(2, 0)\}$.

Codd tables form a weak representation system for selections and projections

Quiz: Are Codd tables weak for SPU/SPJ?

Answer

Codd tables form no weak representation system for SPU/SPJ

Idea: Joins require equalities on variables.

Or-set Relations

Codd tables, where each variable takes values from a finite domain.

Census	SSN	Name	Marital Status
	{ 185, 785 }	Smith	{ 1, 2 }
	{ 185, 186 }	Brown	{ 1, 2, 3, 4 }

Number of represented worlds: $2 \cdot 1 \cdot 2 \cdot 2 \cdot 1 \cdot 4 = 32$.

C	SSN	Name	MS
	185	Smith	1
	185	Brown	1

C	SSN	Name	MS
	185	Smith	1
	185	Brown	2

C	SSN	Name	MS
	185	Smith	1
	185	Brown	3

C	SSN	Name	MS
	185	Smith	1
	185	Brown	4

and so on.

Naïve tables (v-tables)

v-tables are Codd tables, where a variable can occur several times.

R	A	B	C
	0	1	x
	x	z	1
	2	0	v

v-tables form a *weak representation system* for positive relational algebra

Proof Idea

- treat variables in v-tables as constants
- perform standard query evaluation on the v-table

Querying v-tables

R_1	A	B	C
	0	1	x
	x	z	1
	2	0	v

R_2	A	B	C
	1	1	x
	x	z	1

R_3	C	D
	1	1
	x	z

$\overline{\pi_B(R_1)}$	B
	1
	z
	0

$\overline{R_2 \bowtie R_3}$	A	B	C	D
	1	1	x	z
	x	z	1	1

$\overline{R_1 \cup R_2}$	A	B	C
	0	1	x
	x	z	1
	2	0	v
	1	1	x

$\overline{\sigma_{C=1}(R_3)}$	C	D
	1	1

Conditional tables (c-tables) [IL84,Gra91]

A **c-(multi)table** over schema $(R_1[U_1], \ldots, R_k[U_k])$ is a tuple

$$\mathcal{T} = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$$

where $R_i^{\mathcal{T}}$ are v-tables, $\phi^{\mathcal{T}}$ is a global condition, and $\lambda^{\mathcal{T}}$ maps each tuple to a local condition.
Condition: Boolean combination of equalities and inequalities over variables from a finite set and constants.

Semantics

$$\begin{aligned} \mathit{rep}(\mathcal{T}) &= \{(R_1^{\nu(\mathcal{T})}, \ldots, R_k^{\nu(\mathcal{T})}) \mid \nu : \text{valuation}, \nu(\phi^{\mathcal{T}}) \text{ is true}\} \\ R_i^{\nu(\mathcal{T})} &= \{\nu(t) \mid t \in R_i^{\mathcal{T}}, \nu(\lambda^{\mathcal{T}}(t)) \text{ is true}\} \end{aligned}$$

c-table Example

R	Student	Course	
			$x \neq math \wedge x \neq CS$
	Sally	*math*	$z = 0$
	Sally	*CS*	$z \neq 0$
	Sally	x	
	Alice	*bio*	$z = 0$
	Alice	*math*	$x = physics \wedge t = 0$
	Alice	*physics*	$x = physics \wedge t \neq 0$

global condition: $x \neq math \wedge x \neq CS$
local conditions: eg, $x = physics \wedge t = 0$

Example of a possible world:

R	Student	Course
	Sally	*math*
	Sally	*physics*
	Alice	*bio*
	Alice	*math*

How Strong are c-tables?

c-tables form a *strong representation system* for relational algebra

Proof Idea: all relational algebra operations are performed as usual with the addition that the local conditions of the input tuples are copied in the output tuples.

Consider a set of input tuples $\{(t, \lambda), (t_1, \lambda_1), \ldots, (t_n, \lambda_n)\}$. Then,

- for selection ϕ, $\phi(t_i, \lambda_i) = (t_i, \phi(t_i) \wedge \lambda_i)$.
- for projection π_X, $\pi_X(t_i, \lambda_i) = (\pi_X(t_i), \lambda_i)$.
- for product, $(t_i, \lambda_i) \times (t_j, \lambda_j) = (t_i \circ t_j, \lambda_i \wedge \lambda_j)$
- for difference, $(t, \lambda) - \{(t_i, \lambda_i)\} = (t, \lambda \bigwedge \neg\lambda_i)$, if there is a valuation under which all t_i are equal to t; otherwise, the output is (t, λ).

How Strong are c-tables?

T_1	B	C
	x	c

T_2	B	C	
	y	c	y=b
	z	w	

T_3	A	B
	a	y

$\overline{\pi_B(T_2)}$	B	
	y	$y = \mathrm{b}$
	z	

$\overline{T_1 \cup T_2}$	B	C	
	x	c	
	y	c	$y = \mathrm{b}$
	z	w	

$\overline{T_1 \bowtie T_3}$	A	B	C	
	a	y	c	$y = x$

$\overline{\sigma_{B=b}(T_1 \bowtie T_3)}$	A	B	C	
	a	y	c	$y = \mathrm{b} \wedge y = x$

$\overline{T_1 - T_2}$	B	C	
	x	c	$y \neq \mathrm{b} \wedge x \neq z$
	x	c	$y \neq \mathrm{b} \wedge w \neq \mathrm{c}$
	x	c	$y = \mathrm{b} \wedge x \neq \mathrm{b} \wedge x \neq z$
	x	c	$y = \mathrm{b} \wedge x \neq \mathrm{b} \wedge w \neq \mathrm{c}$

Quiz: transitive closure queries on c-tables

Are c-tables strong for transitive closure queries?
Compute the transitive closre of the following binary relation:

T	A	B
	a	b
	x	c
	c	d

Answer

c-tables can represent answers of transitive closure queries

T	A	B
	a	b
	x	c
	c	d

$\overline{tc}(T)$	A	B	
	a	b	
	x	c	
	c	d	
	a	c	$x = \mathrm{b}$
	x	d	
	c	c	$x = \mathrm{d}$
	a	d	$x = \mathrm{b}$

World-set Decompositions (WSDs) [OKA08]

Idea: Exploit independence and mutual exclussiveness present in the data.

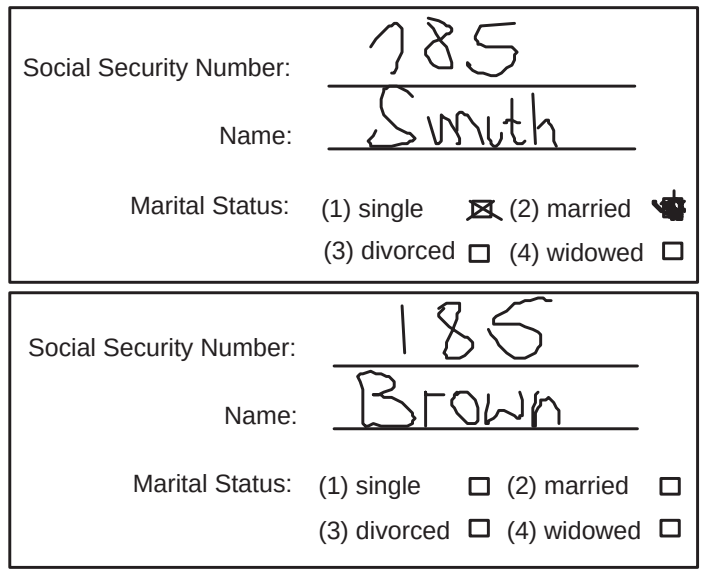
Properties of WSDs

- As expressive as c-tables, but with better complexity results for standard decision problems (to be discussed a bit later).
- (Corollary) Form a strong representation system for any relational query language.
- Any finite world-set can be represented as a WSD.

Note: The slides on WSDs form extra material and were not used in the tutorial!

Census data scenario

Suppose we have to enter the information from forms like these into a database.



R	SSN	N	M
t_1	null	Smith	null
t_2	null	Brown	null

Loss of information, e.g.

1. Smith's SSN is either 185 or 785.
2. Data cleaning: SSN is unique: The case that Smith and Brown both have SSN 185 is excluded.

World-set tables

- Tabular representation of set of possible worlds.
- Schema: The fields in a world. Rows: Alternative worlds.

$R.t_1.SSN$	$R.t_1.N$	$R.t_1.M$	$R.t_2.SSN$	$R.t_2.N$	$R.t_2.M$
185	Smith	1	186	Brown	3
185	Smith	1	$\bot$	$\bot$	$\bot$
185	Smith	2	186	Brown	1

- Pad with $\bot$-values to get uniform arity if not all worlds have the same number of tuples in each relation.
- This represents the world-set

World #1:

R	SSN	N	M
t_1	185	Smith	1
t_2	186	Brown	3

World #2:

R	SSN	N	M
t_1	185	Smith	1

World #3:

R	SSN	N	M
t_1	185	Smith	2
t_2	186	Brown	1

World-set decompositions (WSDs)

World-set decomposition (WSD) : Product decomposition of the world-set table.

$R.t_1.SSN$	$R.t_1.N$	$R.t_1.M$	$R.t_2.SSN$	$R.t_2.N$	$R.t_2.M$
185	Smith	1	186	Brown	1
185	Smith	1	186	Brown	2
185	Smith	1	186	Brown	3
185	Smith	1	186	Brown	4
185	Smith	2	186	Brown	1
		⋮			
785	Smith	2	186	Brown	4

=

$R.t_1.SSN$
185
785

×

$R.t_1.N$
Smith

×

$R.t_1.M$
1
2

×

$R.t_2.SSN$
185
186

×

$R.t_2.N$
Brown

×

$R.t_2.M$
1
2
3
4

Tabsets: Sets of tables

g-(multi)table: a c-(multi)table where ϕ: conjunction of (in)equalities , λ: maps all tuples to "true"

v-(multi)table: a g-(multi)table where ϕ: conjunction of equalities

c-(g-, v-)tabset T $= \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$: a finite set of c-(g-, v-)multitables.

$$rep(\mathbf{T}) = \bigcup_{\mathcal{T} \in \mathbf{T}} rep(\mathcal{T})$$

Inlining tabsets

g-tabset table (gTST) of a g-tabset **A** is a structure (W, λ):

- Turn each multitable into a single tuple:

$$\begin{aligned} W &= \{\text{inline}(\mathcal{A}) \mid \mathcal{A} \in \mathbf{A}\} \\ \text{inline}(\mathcal{A}) &= \text{inline}(R_1^{\mathcal{A}}) \circ \ldots \circ \text{inline}(R_k^{\mathcal{A}}) \\ \text{inline}(R^{\mathcal{A}}) &= t_1 \circ \cdots \circ t_{|R^{\mathcal{A}}|} \circ (\underbrace{\bot, \ldots\ldots\ldots\ldots, \bot}_{(|R|_{\max} - |R^{\mathcal{A}}|) \cdot ar(R)}) \end{aligned}$$

- λ: maps each tuple $\text{inline}(\mathcal{A})$ to the global condition of $\mathcal{A}$

$$rep(W, \lambda) = \bigcup \{rep(\text{inline}^{-1}(t), \lambda(t)) \mid t \in W\}$$

The gTSTs in which λ maps each tuple to a unique common global condition ϕ, i.e. $\lambda : \cdot \mapsto \phi$, capture the gTSTs.

Examples

- g-tabset:

(TID)	S	N	M
t_1	x	Smith	1
t_2	y	Brown	3

(TID)	S	N	M
t_1	185	z	2

- gTST:

$t_1.S$	$t_1.N$	$t_1.M$	$t_2.S$	$t_2.N$	$t_2.M$
x	Smith	1	y	Brown	3
185	z	2	$\perp$	$\perp$	$\perp$

gWSDs: WSDs with global conditions

Let (W, ϕ) be a gTST. Then the pair

$$(\{C_1, \ldots, C_m\}, \phi)$$

where $C_1 \times \ldots \times C_m = W$ is called an
attribute-level world-set m-decomposition (m-gWSD) of (W, ϕ).

An attribute-level gWSD is called a **tuple-level gWSD** if for any two attributes $A, B \in \text{sch}(R)$, and any tuple id t, $R.t.A, R.t.B$ are in the same component schema.

Attribute-level vs. tuple-level gWSDs

Trade-off between succinctness and efficiency

- Any v-tabset representation of the gWSD

$$\left\{ \begin{array}{c|c} C_1 & R.t_1.A \\ \hline & a_1 \\ & b_1 \end{array} \quad \cdots \quad \begin{array}{c|c} C_n & R.t_n.A \\ \hline & a_n \\ & b_n \end{array} \right\}$$

  where the a_i, b_i are distinct domain values takes space exponential in n.

- Given an attribute-level (g)WSD $\mathcal{W}$, checking whether the empty world is in $rep(\mathcal{W})$ is NP-complete.

- Tuple certainty is coNP-hard for attribute-level gWSDs.

gWSDs capture the c-tables: gWSDs $\Rightarrow$ c-tables

Given a tuple-level gWSD $\mathcal{W} = (\{C_1, \ldots, C_m\}, \phi)$ for a g-tabset over schema Σ

- Define a c-multitable over Σ
- Transform the component tuples into tuples of the c-multitable
- Tuples defined in the same component tuple are assigned the same local condition

gWSDs $\Rightarrow$ c-tables

Input: $\mathcal{W} = (\{C_1, \ldots, C_m\}, \phi)$: tuple-level m-gWSD for a g-tabset over schema $(R_1[U_1], \ldots, R_k[U_k])$

- If $|C_i| = 0$, then $rep(\mathcal{W}) = \emptyset$. The result is any c-multitable with unsatisfiable global condition, e.g. $(x \neq x)$.
- If $|C_i| > 0$ for all i, construct a c-multitable $\mathcal{T} = (R_1^{\mathcal{T}}, \ldots, R_k^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$ over schema $(R_1[U_1], \ldots, R_k[U_k])$.
- The global condition ϕ of the gWSD becomes global condition $\phi^{\mathcal{T}}$ of the c-multitable.

gWSDs $\Rightarrow$ c-tables

C_j	$R.t_1.A_1$	$\ldots$	$R.t_1.A_k$	$R.t_2.A_1$	$\ldots$	$R.t_2.A_k$	$R.t_3.A_1$ $\ldots$
				$\vdots$			
w_i	a_{11}	$\ldots$	a_{1k}	a_{21}	$\ldots$	a_{2k}	$\bot$ $\ldots$
				$\vdots$			

- Translate the ith tuple of C_j into tuples of the c-multitable

$R^{\mathcal{T}}$	A_1	$\ldots$	A_k	λ
		$\ldots$		
t_{1i}	a_{11}	$\ldots$	a_{1k}	$\lambda(t_{1i})$
t_{2i}	a_{21}	$\ldots$	a_{2k}	$\lambda(t_{2i})$
		$\ldots$		
		$\ldots$		

- Local conditions ($n_j = |C_j|$):

$$\lambda(t) = \begin{cases} \mathit{true} & n_j = 1 \\ (x_j = i) & 1 \leq i < n_j \text{ and } n_j > 1 \\ (x_j \neq 1 \wedge \ldots \wedge x_j \neq n_j - 1) & i = n_j \text{ and } n_j > 1. \end{cases}$$

gWSDs $\Rightarrow$ c-multitables: Example

- 1-gWSD $(\{C_1\}, \phi)$

C_1	$R.t_1.A$	$R.t.B$	$R.t_2.A$	$R.t_2.B$
w_1	x	y	$\perp$	$\perp$
w_2	1	z	z	3
w_3	1	2	$\perp$	$\perp$

$$\phi = (x \neq 1) \wedge (x \neq y) \wedge (z \neq 2)$$

- Equivalent c-table $(T, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$

T	A	B	$\lambda^{\mathcal{T}}$
	x	y	$(x_1 = 1)$
	1	z	$(x_1 = 2)$
	z	3	$(x_1 = 2)$
	1	2	$(x_1 \neq 1 \wedge x_1 \neq 2)$

$$\phi^{\mathcal{T}} = (x \neq 1) \wedge (x \neq y) \wedge (z \neq 2)$$

gWSDs capture the c-tables: c-tables $\Rightarrow$ gWSDs

Given a c-table $\mathcal{T} = (T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$

$T^{\mathcal{T}}$	A_1	$\ldots$	A_k	$cond$
				$\phi^{\mathcal{T}}$
t_1	$x_{1,1}$	$\ldots$	$x_{1,k}$	$\lambda_1^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
t_n	$x_{n,1}$	$\ldots$	$x_{n,k}$	$\lambda_n^{\mathcal{T}}$

Construct a 1-gWSD $C(t_1.A_1, \ldots, t_1.A_k, \ldots, t_n.A_1, \ldots, t_n.A_k)$

- Inline the c-table
- Get rid of the local conditions
  - generate the possible valuations of the variables in the local conditions
  - it is sufficient to consider only the consistent conjunctions of comparisons between constants and variables from $\mathcal{T}$

c-tables $\Rightarrow$ gWSDs

- Given a c-table $\mathcal{T} = (T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$

$T^{\mathcal{T}}$	A_1	$\ldots$	A_k	$cond$
				$\phi^{\mathcal{T}}$
t_1	$x_{1,1}$	$\ldots$	$x_{1,k}$	$\lambda_1^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
t_n	$x_{n,1}$	$\ldots$	$x_{n,k}$	$\lambda_n^{\mathcal{T}}$

- We define a corresponding 1-gWSD $(\{C\}, \phi')$ with gTST C of schema

$$C(t_1.A_1, \ldots, t_1.A_k, \ldots, t_n.A_1, \ldots, t_n.A_k)$$

c-tables $\Rightarrow$ gWSDs

- Given a c-table $\mathcal{T} = (T^{\mathcal{T}}, \phi^{\mathcal{T}}, \lambda^{\mathcal{T}})$

$T^{\mathcal{T}}$	A_1	$\ldots$	A_k	$cond$
				$\phi^{\mathcal{T}}$
t_1	$x_{1,1}$	$\ldots$	$x_{1,k}$	$\lambda_1^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
t_n	$x_{n,1}$	$\ldots$	$x_{n,k}$	$\lambda_n^{\mathcal{T}}$

- We define a corresponding 1-gWSD $(\{C\}, \phi')$ with gTST C of schema

$$C(t_1.A_1, \ldots, t_1.A_k, \ldots, t_n.A_1, \ldots, t_n.A_k)$$

- Let $\mathbf{X}_{\mathcal{T}}$ and $\mathbf{D}_{\mathcal{T}}$ be the set of all variables and the set of all constants in $\mathcal{T}$, respectively.
- We compute a set of consistent $\Theta = \bigwedge\{\tau\,\theta\,\tau' \mid \tau, \tau' \in \mathbf{X}_{\mathcal{T}} \cup \mathbf{D}_{\mathcal{T}}\}$ where $\theta \in \{=, \neq\}$ and $\Theta \vDash \phi^{\mathcal{T}}$.

c-tables $\Rightarrow$ gWSDs

$T^{\mathcal{T}}$	A_1	$\ldots$	A_k	cond
				$\phi^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
t_i	$x_{i,1}$	$\ldots$	$x_{i,k}$	$\lambda_i^{\mathcal{T}}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$

$\Rightarrow$

C	$\ldots$	$t_i.A_1$	$\ldots$	$t_i.A_k$	$\ldots$
			$\vdots$		
	$\ldots$	$s_{i,1}$	$\ldots$	$s_{i,k}$	$\ldots$
			$\vdots$		

- We have tuple $\langle s_{1,1}, \ldots, s_{1,k}, \ldots, s_{n,1}, \ldots, s_{n,k} \rangle \in C$ if and only if for some consistent Θ

$$s_{i,j} = \begin{cases} \bot & \ldots \quad \Theta \not\models \lambda_i^{\mathcal{T}} \\ c & \ldots \quad c \in \mathbf{D}_T, \Theta \models \lambda_i^{\mathcal{T}}, \Theta \models (x_{i,j} = c) \\ h([x_{i,j}]_=) & \ldots \quad \Theta \models \lambda_i^{\mathcal{T}}, \forall c \in \mathbf{D}_T \; \Theta \models (x_{i,j} \neq c) \end{cases}$$

$h([x_{i,j}]_=)$ is the representative element of the equivalent class of $x_{i,j}$ with respect to the equalities given by Θ.

c-tables $\Rightarrow$ gWSDs: Example

- c-table T with global condition ϕ

T	A	B	$cond$
			$\phi = (x \neq 1) \wedge (x = z)$
t_1	x	1	$(x \neq 2)$
t_2	z	y	$(y \neq 2)$

- Equivalent 1-gWSD $(\{C\}, \phi')$

C	$t_1.A$	$t_1.B$	$t_2.A$	$t_2.B$	Θ
	$\bot$	$\bot$	$\bot$	$\bot$	$x = 2 \wedge y = 2 \wedge z = 2$
	$\bot$	$\bot$	2	y	$x = 2 \wedge z = 2 \wedge y \neq 2$
	x	1	$\bot$	$\bot$	$x \neq 2 \wedge y = 2 \wedge x \neq 1 \wedge x = z$
	x	1	x	y	$x \neq 2 \wedge y \neq 2 \wedge x \neq 1 \wedge x = z$

global cond: $\phi' = (x \neq 1) \wedge (x \neq 2) \wedge (y \neq 2)$

Overview of Representation Systems

System	Is Weak For..	Is Strong For..
Codd tables	PSU	PU
or-sets	PSU	PU
v-tables	PS^+UJ	PU
c-tables	PSUJD	PSUJD
(g)WSDs	PSUJD	PSUJD

P = Projection, S = Selection, S^+ = pos S, U = Union, J = Join, D = Difference.

Decision Problems

Input: Representation system $\mathcal{W}$, instance $I = (R^I)$, tuple t.

Decision problems:

Tuple Possibility:	$\exists \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$
Tuple Certainty:	$\forall \mathcal{A} \in rep(\mathcal{W}) : t \in R^{\mathcal{A}}$
Instance Possibility:	$\exists \mathcal{A} \in rep(\mathcal{W}) : R^I = R^{\mathcal{A}}$
Instance Certainty:	$\forall \mathcal{A} \in rep(\mathcal{W}) : R^I = R^{\mathcal{A}}$
Tuple Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$
Tuple Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in rep(\mathcal{W}) : t \in Q(\mathcal{A})$
Instance Q-Possibility (query Q fixed):	$\exists \mathcal{A} \in rep(\mathcal{W}) : R^I = Q(\mathcal{A})$
Instance Q-Certainty (query Q fixed):	$\forall \mathcal{A} \in rep(\mathcal{W}) : R^I = Q(\mathcal{A})$

Example

R_1	Student	Course	
			$x \neq math \wedge x \neq CS$
	Sally	*math*	$z = 0$
	Sally	*CS*	$z \neq 0$
	Sally	x	
	Alice	*bio*	$z = 0$
	Alice	*math*	$x = physics \wedge t = 0$
	Alice	*physics*	$x = physics \wedge t \neq 0$

- Which of the following tuples is possible/certain?
  (Alice,bio), (Sally,math), (Sally,bio), (Sally,agriculture), (Banana,bio)
- Which of the following tuples is $\pi_{Student}(R)$-certain? (Sally), (Alice)
- Which of the following instances is possible/certain?
  $\emptyset$, {(Alice,bio),(Sally,CS)}, {(Sally,CS),(Sally,math)}

Complexity results for decision problems

	v-tables	(g)WSDs	c-tables
Tuple possibility	PTIME	PTIME	NP-compl.
Tuple certainty	PTIME	PTIME	coNP-compl.
Instance possibility	NP-compl.	NP-compl.	NP-compl.
Instance certainty	PTIME	PTIME	coNP-compl.
Tuple q-possibility *positive relational algebra*	NP-compl. PTIME	NP-compl. PTIME	NP-compl. NP-compl.
Tuple q-certainty *positive relational algebra*	coNP-compl. PTIME	coNP-compl. coNP-compl.	coNP-compl. coNP-compl.
Instance q-possibility	NP-compl.	NP-compl.	NP-compl.
Instance q-certainty *positive relational algebra*	coNP-compl. PTIME	coNP-compl. coNP-compl.	coNP-compl. coNP-compl.

Results for tuple-level gWSDs from [OKA08], for v- and c-tables from [AKG91,Gra91].

Quiz: Instance possibility for WSDs

Why is instance possibility for WSDs NP-hard?

Answer

Idea: Reduction from Exact Cover by 3-Sets.
Given a set X with $|X| = 3q$ and a collection C of 3-element subsets of X, the exact cover by 3-sets problem is to decide whether there exists a subset $C' \subseteq C$, such that every element of X occurs in exactly one member of C'.
Construction

- The set X is encoded as an instance consisting of a unary relation I_X over schema $I_X[A]$ with $3q$ tuples.
- The collection C is represented as a WSD $\mathcal{W} = \{C_1, \ldots, C_q\}$ encoding a relation R over schema $R[A]$, where $C_1, \ldots, C_q$ are component relations.
- The schema of a component C_i is $C_i[R.t_{j+1}.A, R.t_{j+2}.A, R.t_{j+3}.A]$, where $j = \lfloor \frac{i}{3} \rfloor$.
- Each 3-element set $c = \{x, y, z\} \in C$ is encoded as a tuple (x, y, z) in each of the components C_i.

The problem of deciding whether there is an exact cover by 3-sets of X is equivalent to deciding whether $I_X \in rep(\mathcal{W})$.

Example

Consider the set X and the collection of 3-element sets C defined as

$$X = \{1,2,3,4,5,6,7,8,9\}$$
$$C = \{\{1,5,9\},\{2,5,8\},\{3,4,6\},\{2,7,8\},\{1,6,9\}\}$$

I_X	A
	1
	2
	3
	4
	5
	6
	7
	8
	9

C_1	t_1.A	t_2.A	t_3.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

C_2	t_4.A	t_5.A	t_6.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

C_3	t_7.A	t_8.A	t_9.A
w_1	1	5	9
w_2	2	5	8
w_3	3	4	6
w_4	2	7	8
w_5	1	6	9

A possible cover of X, or equivalently, a world of $rep(\mathcal{W})$ equivalent to I_X, is the world inline$^{-1}(w_1 \circ w_3 \circ w_4)$ or, by resolving the record composition,

inline$^{-1}(t_1.A:1, t_2.A:5, t_3.A:9, t_4.A:3, t_5.A:4, t_6:A:6, t_7:2, t_8:7, t_9.A:8)$.

Literature on Incomplete Relational Data

[AHV95] Abiteboul, Hull, and Vianu. The section on incomplete information from *Foundations of Databases*. 1995.

[AKG91] Abiteboul, Kanellakis, Grahne. *On the representation and querying of sets of possible worlds*. TCS 1991.

[Gra91] Grahne. *The Problem of Incomplete Information in Relational Databases*. LNCS 554, 1991.

[IL84] Imielinski, Lipski. *Incomplete information in relational databases*. JACM 1984.

[OKA08] Olteanu, Koch, Antova. *World-set Decompositions: Expressiveness and Efficient Algorithms*. TCS 2008.

2. Probabilistic Relational Data

Outline of Part 2

- From incomplete to probabilistic data
- Complexity of query evaluation
- Dichotomy result for conjunctive queries without self-joins
- Exact evaluation techniques
  enhanced query plans for probability computation
- Approximate evaluation techniques
  Monte Carlo simulation, deterministic approximation algorithms

Where do probabilities come from?

- Probabilistic extraction models used to populate probabilistic databases
- OCR on manually filled census forms
- Unreliable sensor readings, ...

Possible segmentations of unstructured text [Sarawagi VLDB'06]

`52-A Goregaon West Mumbai 400 076`

ID	HouseNo	Area	City	PinCode	**Event**	**P**
1	52	Goregaon West	Mumbai	400 062	$x = 1$	**0.1**
1	52-A	Goregaon	West Mumbai	400 062	$x = 2$	**0.2**
1	52-A	Goregaon West	Mumbai	400 062	$x = 3$	**0.4**
1	52	Goregaon	West Mumbai	400 062	$x = 4$	**0.2**
...	...	...	...	...		...

- *Sound* confidence values obtained using probabilistic extraction models
- Output a ranked list of possible extractions
  Empty answer to query: Find movies filmed in 'West Mumbai'
- Several segmentations are required to cover most of the probability mass and improve recall

From Incomplete to Probabilistic Data

Syntax.

Probabilistic databases are c-(multi)tables for finite world-sets where

- There is a finite set of independent random variables $\mathbf{X} = \{x_1, \ldots, x_n\}$ with finite domains $\mathrm{Dom}_{x_1}, \ldots, \mathrm{Dom}_{x_n}$.
- There is a probability distribution over the assignments of each variable.
- Local conditions (called *lineage*) are conjunctions of atomic events of the form $x_i = a$ or $x_i \neq a$ where $x_i \in \mathbf{X}$ and $a \in \mathrm{Dom}_{x_i}$.

Semantics.

- Possible worlds defined (as for c-tables) by total assignments θ over $\mathbf{X}$.
- The world defined by assignment θ
  - consists of all tuples with lineage ϕ such that $\theta(\phi) = \text{true}$.
  - has probability defined by the product of probabilities of each assignment in θ.

A probabilistic database can represent any finite set of possible worlds.

Tuple-independent Probabilistic Databases

Tuple-independent: Tuples have independent lineage, or equivalently

- Each tuple t is associated with a Boolean random variable x_t.
- Tuple t is in the world defined by θ if $x_t = \mathit{true}$ holds in θ.

Cust

ckey	cname	V	P
1	Joe	x_1	0.1
2	Dan	x_2	0.2
3	Li	x_3	0.3
4	Mo	x_4	0.4

Ord

okey	ckey	odate	V	P
1	1	1995-01-10	y_1	0.1
2	1	1996-01-09	y_2	0.2
3	2	1994-11-11	y_3	0.3
4	2	1993-01-08	y_4	0.4
5	3	1995-08-15	y_5	0.5
6	3	1996-12-25	y_6	0.6

Item

okey	disc	ckey	V	P
1	0.1	1	z_1	0.1
1	0.2	1	z_2	0.2
3	0.4	2	z_3	0.3
3	0.1	2	z_4	0.4
4	0.4	2	z_5	0.5
5	0.1	3	z_6	0.6

Tuple-independent Probabilistic Databases

Consider the world $\mathcal{A}$ defined by a total assignment θ:

- x_1, y_1, z_1 are true, and
- all other variables are false.

Cust

ckey	cname	V	P
1	**Joe**	x_1	**0.1**
2	Dan	x_2	0.2
3	Li	x_3	0.3
4	Mo	x_4	0.4

Ord

okey	ckey	odate	V	P
1	**1**	**1995-01-10**	y_1	**0.1**
2	1	1996-01-09	y_2	0.2
3	2	1994-11-11	y_3	0.3
4	2	1993-01-08	y_4	0.4
5	3	1995-08-15	y_5	0.5
6	3	1996-12-25	y_6	0.6

Item

okey	disc	ckey	V	P
1	**0.1**	**1**	z_1	**0.1**
1	0.2	1	z_2	0.2
3	0.4	2	z_3	0.3
3	0.1	2	z_4	0.4
4	0.4	2	z_5	0.5
5	0.1	3	z_6	0.6

Tuple-independent Probabilistic Databases

Consider the world $\mathcal{A}$ defined by a total assignment θ:

- x_1, y_1, z_1 are true, and
- all other variables are false.

$\text{Cust}^{\mathcal{A}}$

ckey	cname
1	Joe

$\text{Ord}^{\mathcal{A}}$

okey	ckey	odate
1	1	1995-01-10

$\text{Item}^{\mathcal{A}}$

okey	disc	ckey
1	0.1	1

Probability of $\mathcal{A}$ = the product of the probabilities of the assignments in θ:

$$Pr(\mathcal{A}) = Pr(\theta) = Pr(x_1) \cdot Pr(y_1) \cdot Pr(z_1) \cdot \Pi\{Pr(\bar{v}) \mid v \in \{x_2, \ldots, x_4, y_2, \ldots, y_6, z_2, \ldots, z_6\}\}$$

Query Evaluation on Probabilistic Databases

- Follows query evaluation on c-tables,
- New challenge: Computation of probabilities of query answers.

Query asking for the dates of discounted orders shipped to customer 'Joe':

$Q(odate)$:- $\mathsf{Cust}(ckey, 'Joe'), \mathsf{Ord}(okey, ckey, odate), \mathsf{Item}(okey, disc, ckey), disc > 0$

odate	V_c	P_c	V_o	P_o	V_i	P_i	tuple probability
1995-01-10	x_1	0.1	y_1	0.1	z_1	0.1	$0.1 \cdot 0.1 \cdot 0.1$
1995-01-10	x_1	0.1	y_1	0.1	z_2	0.2	$0.1 \cdot 0.1 \cdot 0.2$

Probability of (1995-01-10) = Probability of associated lineage $x_1y_1z_1 + x_1y_1z_2$.

Challenge: Scalable probability computation for **distinct** answer tuples.

Complexity Class #P (Sharp P)

Class of functions $f(x)$ for which there exists a PTIME non-deterministic Turing machine M such that $f(x) =$ number of accepting computations of M on input x.

Informally: #P is the class of **counting problems** associated with decision problems in NP [Val79].

- NP-complete problem: SAT = "given formula ϕ, is ϕ satisfiable?"
- #P-complete problem: #SAT = "given formula ϕ, count # of satisfying assignments"

A PTIME machine with a #P oracle ($P^{\#P}$) can solve any problem in PH (polynomial hierarchy) with only one #P query [Toda91].

Our concern here:

- #SAT is #P-complete already for bipartite positive 2DNFs! [Val79,PB83]
- Probability computation for positive k-DNF formulas is thus #P-complete.

#P-hard Queries

The query $Q :- R(x), S(x, y), T(y)$ is #P-hard [Graedel98].

Proof idea. Reduction from #SAT for bipartite positive 2-DNF.

- Given 2-DNF ϕ over variable sets X and Y.
- Tuple-independent tables R and T have one distinct tuple for each variable in X and Y resp.
- Certain table S encodes the clauses of ϕ.
- For each variable, set its probability to $1/2$.
- Then, $\#\phi = P(Q) \cdot 2^{|Vars(\phi)|}$.

Example: $\phi = x_1y_1 + x_1y_2 + x_2y_2 + x_3y_3$.

R	A	V	P
	a_1	x_1	0.5
	a_2	x_2	0.5
	a_3	x_3	0.5

S	A	B
	a_1	b_1
	a_1	b_2
	a_2	b_2
	a_3	b_3

T	B	V	P
	b_1	y_1	0.5
	b_2	y_2	0.5
	b_3	y_3	0.5

Dichotomy Property

Discussed here [Dalvi&Suciu07]:

- Conjunctive queries without self-joins (CQ^1) on
- Tuple-independent databases.

> The data complexity of any CQ^1 query is either FP or #P-hard.

- FP is the class of functions that can be solved by a deterministic Turing machine in PTIME.
- Problems in FP can have *any* output that can be computed in PTIME, not only true/false.

Further tractability results not covered here:

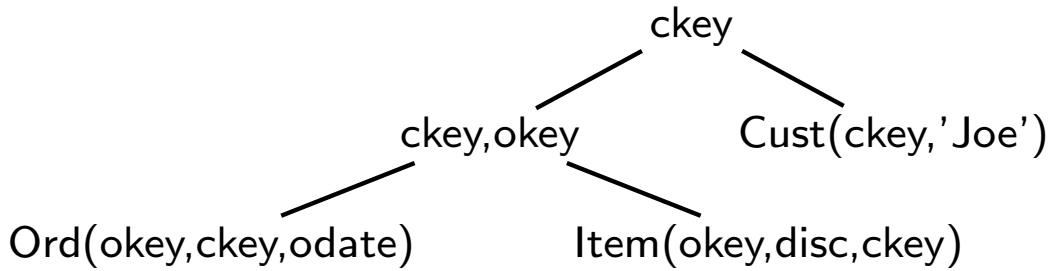
- Dichotomy for queries with self-joins [Dalvi&Suciu07b]
- Tractable queries with inequalities ($<, \leq, \neq$) [O.&Huang08,O.&Huang09]

Tractable CQ^1 queries are hierarchical

A query is *hierarchical* if for any two non-head variables, either their sets of subgoals are disjoint, or one set is contained in the other.

$Q(odate)$:- $\mathsf{Cust}(ckey, 'Joe'), \mathsf{Ord}(okey, ckey, odate), \mathsf{Item}(okey, disc, ckey), disc > 0.$
is hierarchical; also without odate as head variable.

subgoals(disc)={Item}, subgoals(okey)={Ord, Item}, subgoals(ckey)={Cust, Ord, Item}.
It holds that subgoals(disc)$\subseteq$ subgoals(okey)$\subseteq$ subgoals(ckey).



$Q'()$:- $\mathsf{Cust}(ckey, 'Joe'), \mathsf{Ord}(okey, ckey, odate), \mathsf{Item}'(okey, disc), disc > 0.$
is **not** hierarchical: subgoals(okey)={Ord, $\underline{\mathsf{Item}'}$}, subgoals(ckey) = {Ord, $\underline{\mathsf{Cust}}$}.

Queries with subqueries $R(\ldots, X, \ldots), S(\ldots, X, \ldots, Y, \ldots), T(\ldots, Y, \ldots)$
where X and Y are non-head query variables, are not hierarchical.

Quiz: Hierarchical Queries

- Is $Q : -R(x, y), S(y, a, u), T(y, y, v)$ hierarchical?
- Is $Q : -R(x, y), S(x, y, z), T(x, z)$ hierarchical?
- Is $Q(z) : -R(x, y), S(x, y, z), T(x, z)$ hierarchical?
- Is $Q : -R(x, a), S(y, u, x), T(u, y), U(x, y)$ hierarchical?
- Is $Q(x) : -R(x, a), S(y, u, x), T(u, y), U(x, y)$ hierarchical?
- Is $Q : -R(x, y, z), S(z, u, y), T(y, v, z, x), U(y)$ hierarchical?

Answer

- Is $Q : -R(x, y), S(y, a, u), T(y, y, v)$ hierarchical? YES
- Is $Q : -R(x, y), S(x, y, z), T(x, z)$ hierarchical?NO
- Is $Q(z) : -R(x, y), S(x, y, z), T(x, z)$ hierarchical?YES
- Is $Q : -R(x, a), S(y, u, x), T(u, y), U(x, y)$ hierarchical?NO
- Is $Q(x) : -R(x, a), S(y, u, x), T(u, y), U(x, y)$ hierarchical?YES
- Is $Q : -R(x, y, z), S(z, u, y), T(y, v, z, x), U(y)$ hierarchical?YES

Query Evaluation

Subsumed by general probabilistic inference! Move to next topic.

Query Evaluation

Subsumed by general probabilistic inference! Move to next topic.

Hold on! What about scalability?
Two fundamental aspects of databases are relevant here:

- the separation of (very large) data and (small and fixed) query, and
- the use of mature relational query engines.

A Toolbox of Query Evaluation Techniques [Excerpt]

Exact techniques

- MystiQ ("restricted" safe plans), SPROUT ("unrestricted" query plans) query plans for hierarchical queries and tractable queries with inequalities probabilistic inference for hard queries
- Inference algorithms in AI: variable elimination, junction trees, ... Tractable cases for bounded treewidth [Zabiyaka&Darwiche'06], [Huang&Darwiche'01]

Approximation techniques with error guarantees for arbitrary queries on c-table-like probabilistic databases

- MystiQ, MayBMS: Monte Carlo simulations using Karp-Luby FPTRAS
- SPROUT: deterministic algorithm that incrementally refines lower & upper bounds on the output probability using decomposition methods for DNF formulas

Exact Query Evaluation using SPROUT

Cast the query evaluation problem as a decision diagram construction problem.

- Given a query q and a probabilistic database D, each distinct tuple $t \in q(D)$ is associated with a DNF expression ϕ_t.
- Probability of t is probability of lineage ϕ_t.
- Compile ϕ_t into a propositional theory with efficient model counting. We use (among others) **binary decision diagrams (BDDs)**, for which probability computation can be done in one traversal.
- Probability of ϕ_t is then the probability of its BDD.

To achieve true scalability, SPROUT employs secondary-storage techniques for BDD construction and probability computation.

- the query structure and DB constraints are used to guide the search for good BDD variable orders
- the BDDs are not materialized, their probabilities are computed on the fly

BDDs

- Commonly used to represent compactly large Boolean expressions.
- Idea: Decompose Boolean expressions using variable elimination and avoid redundancy in the representation.
  Variable elimination by Shannon's expansion: $\phi = x \cdot \phi\,|_x + \bar{x} \cdot \phi\,|_{\bar{x}}$.
- Variable order π = order of variable eliminations;
  the same variable order on all root-to-leaf paths $\Rightarrow$ ordered BDDs or **OBDDs**
- An OBDD for ϕ is uniquely identified by the pair (ϕ, π).
- Supports linear-time probability computation.

$$\begin{aligned} Pr(\phi) &= Pr(x \cdot \phi\,|_x + \bar{x} \cdot \phi\,|_{\bar{x}}) \\ &= Pr(x \cdot \phi\,|_x) + Pr(\bar{x} \cdot \phi\,|_{\bar{x}}) \\ &= Pr(x) \cdot Pr(\phi\,|_x) + Pr(\bar{x}) \cdot Pr(\phi\,|_{\bar{x}}) \end{aligned}$$

Compilation example

R	A	B	V_r
	a_1	b_1	x_1
	a_2	b_1	x_2
	a_2	b_2	x_3
	a_3	b_3	x_4

S	A	C	V_s
	a_1	c_1	y_1
	a_1	c_2	y_2
	a_2	c_1	y_3
	a_4	c_2	y_4

q :- $R(A, B), S(A, C)$

	V_r	V_s
	x_1	y_1
	x_1	y_2
	x_2	y_3
	x_3	y_3

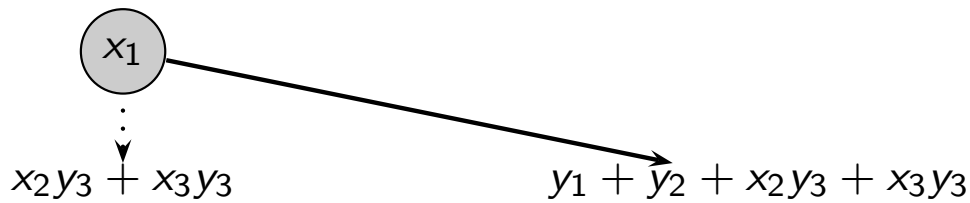
Query q has lineage $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$.
Assume variable order: $\pi = x_1y_1y_2x_2x_3y_3$.
Task: Construct the OBDD (ϕ, π).

Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 1: Eliminate variable x_1 in ϕ.

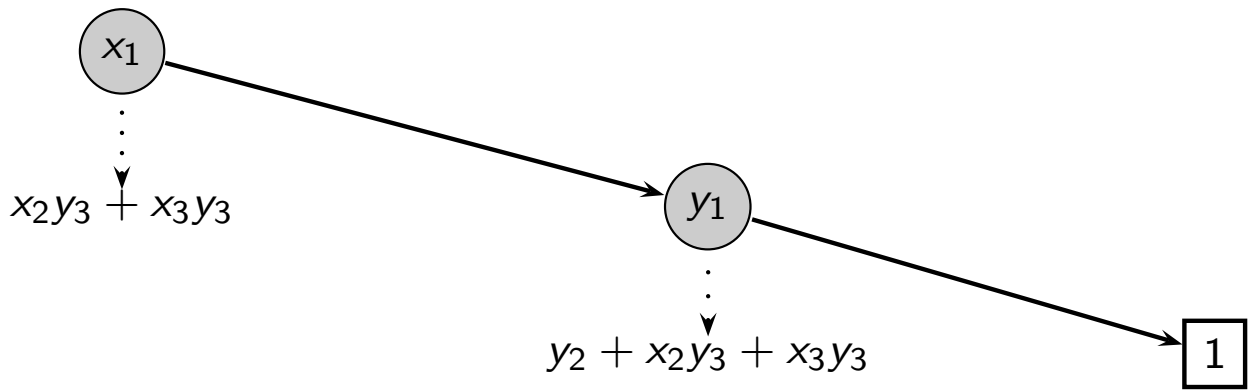


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 2: Eliminate variable y_1.

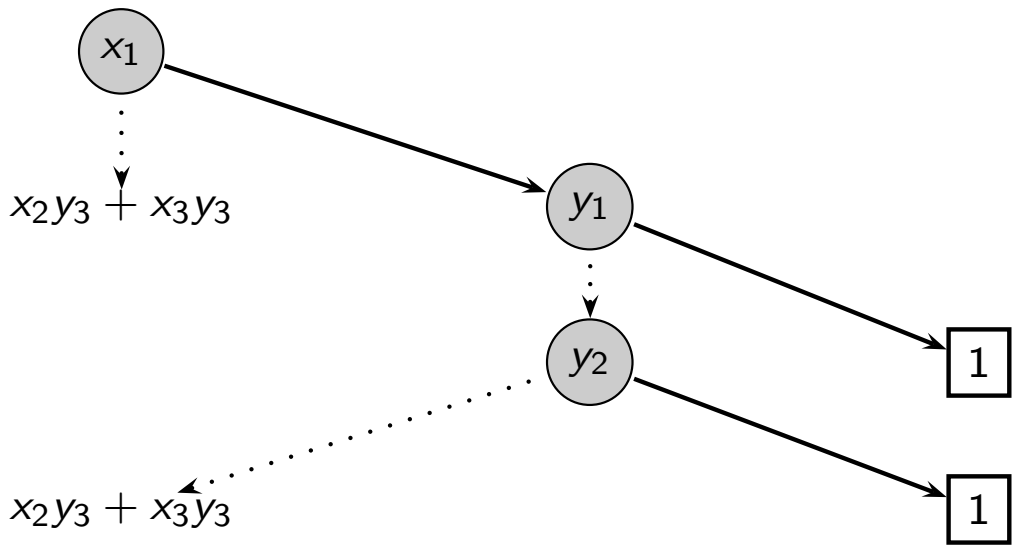


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 3: Eliminate variable y_2.



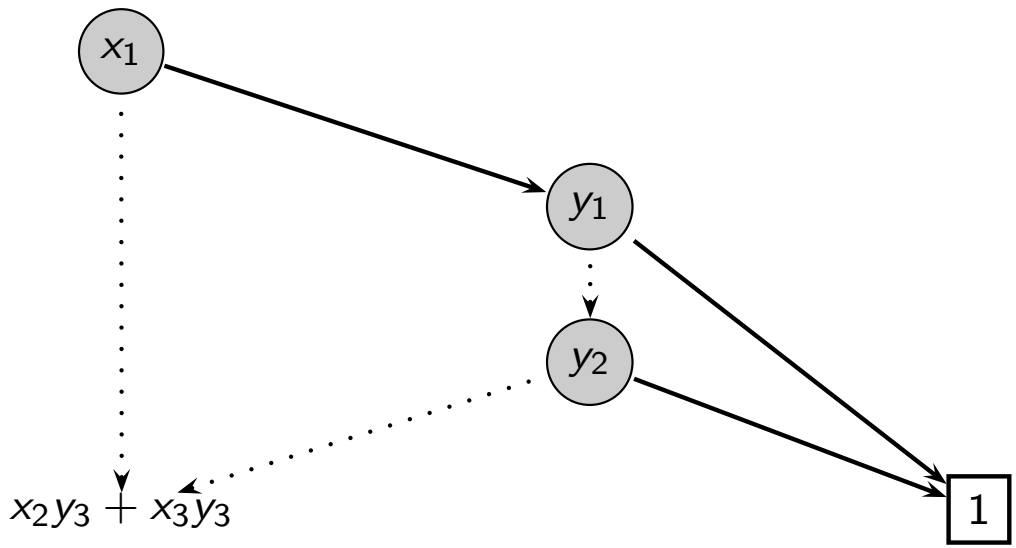
Some leaves have the same expressions $\Rightarrow$ Represent them only once!

Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 4: Merge leaves with the same expressions.

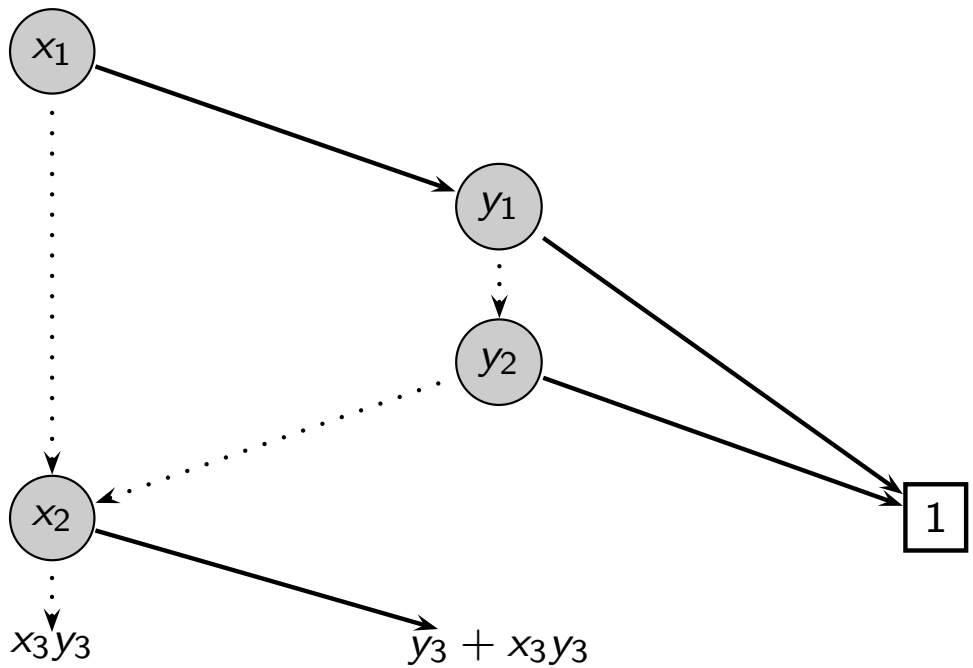


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 5: Eliminate variable x_2.

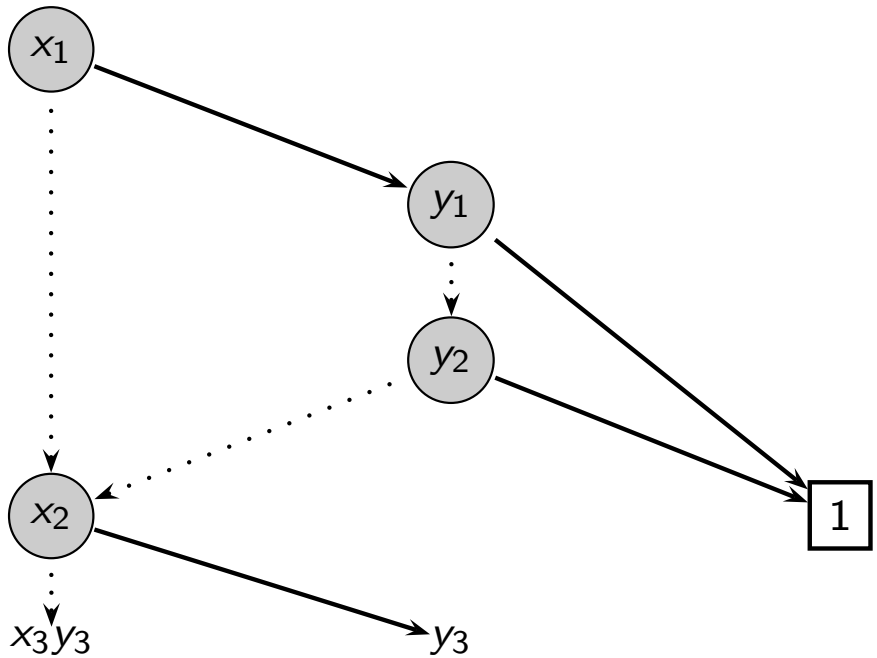


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 6: Replace $y_3 + x_3y_3$ by y_3.

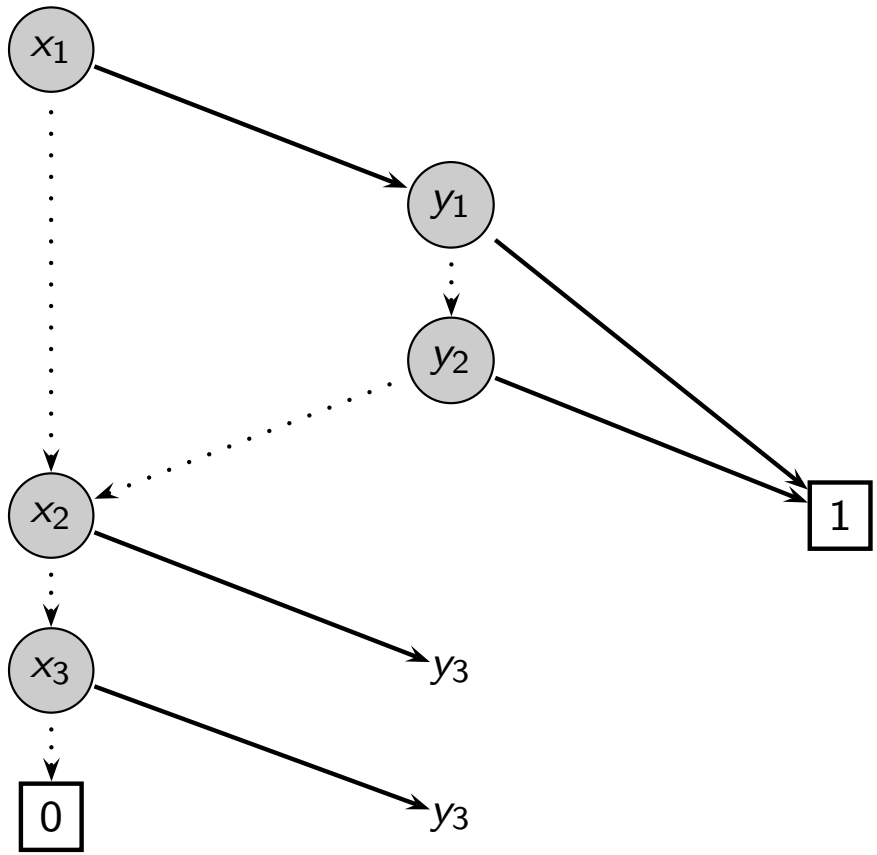


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 7: Eliminate variable x_3.

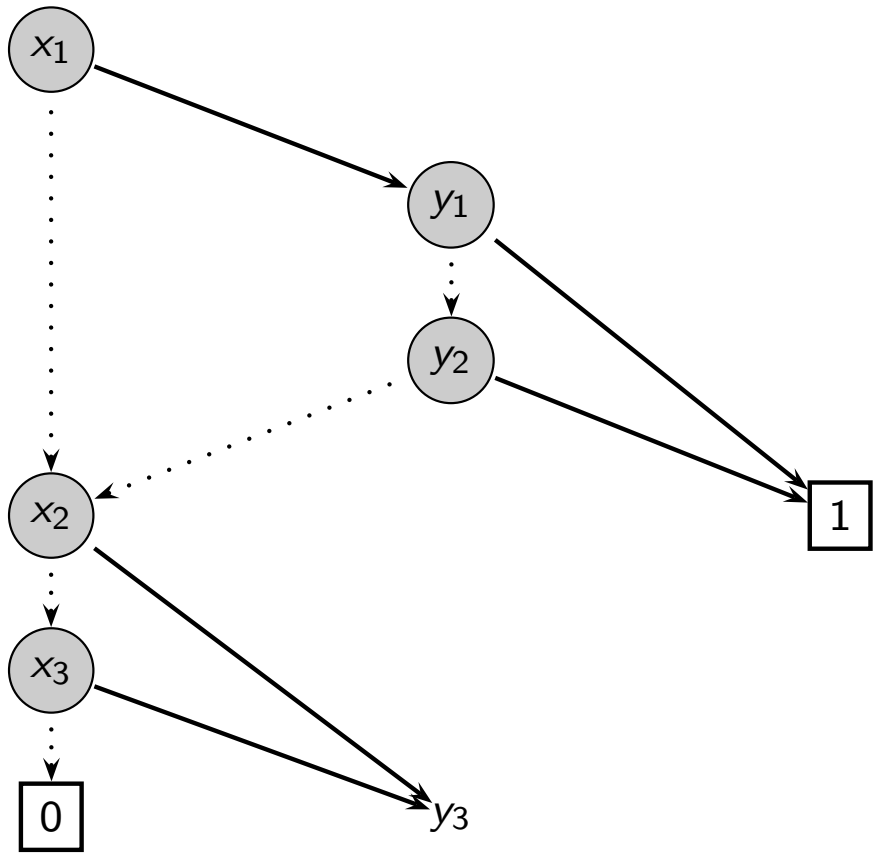


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 8: Merge leaves with the same expression y_3.

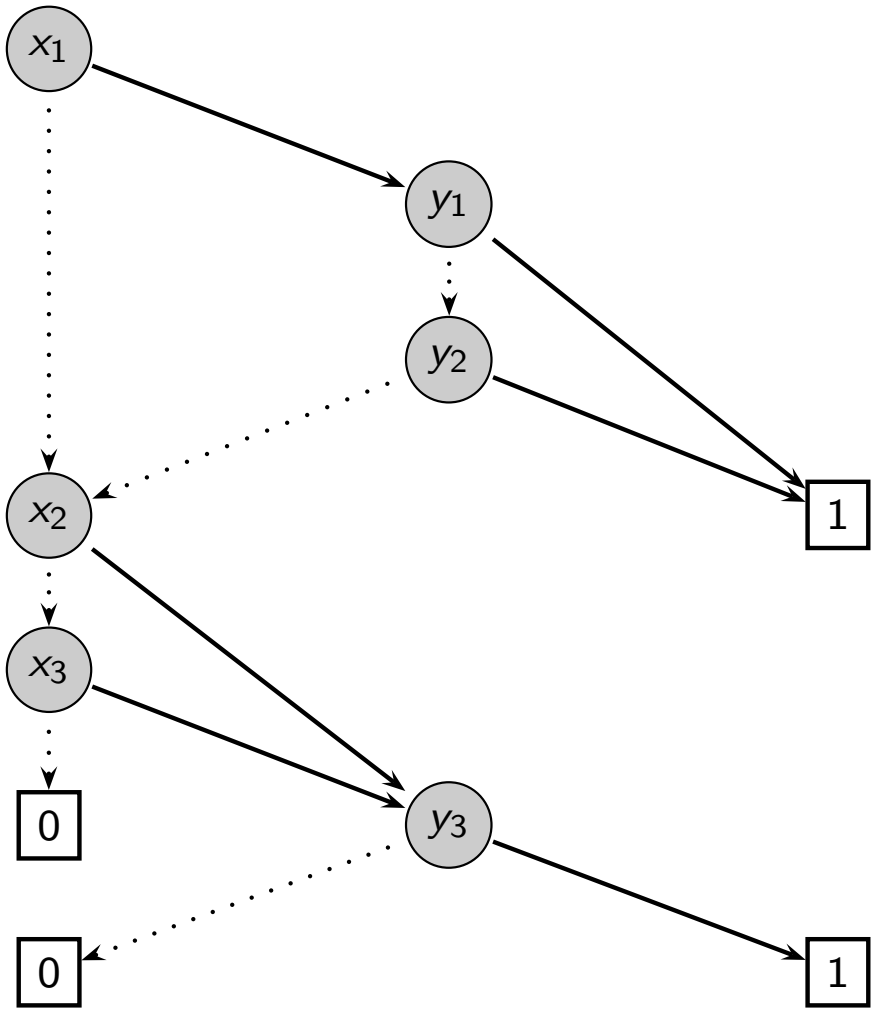


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 9: Eliminate variable y_3.

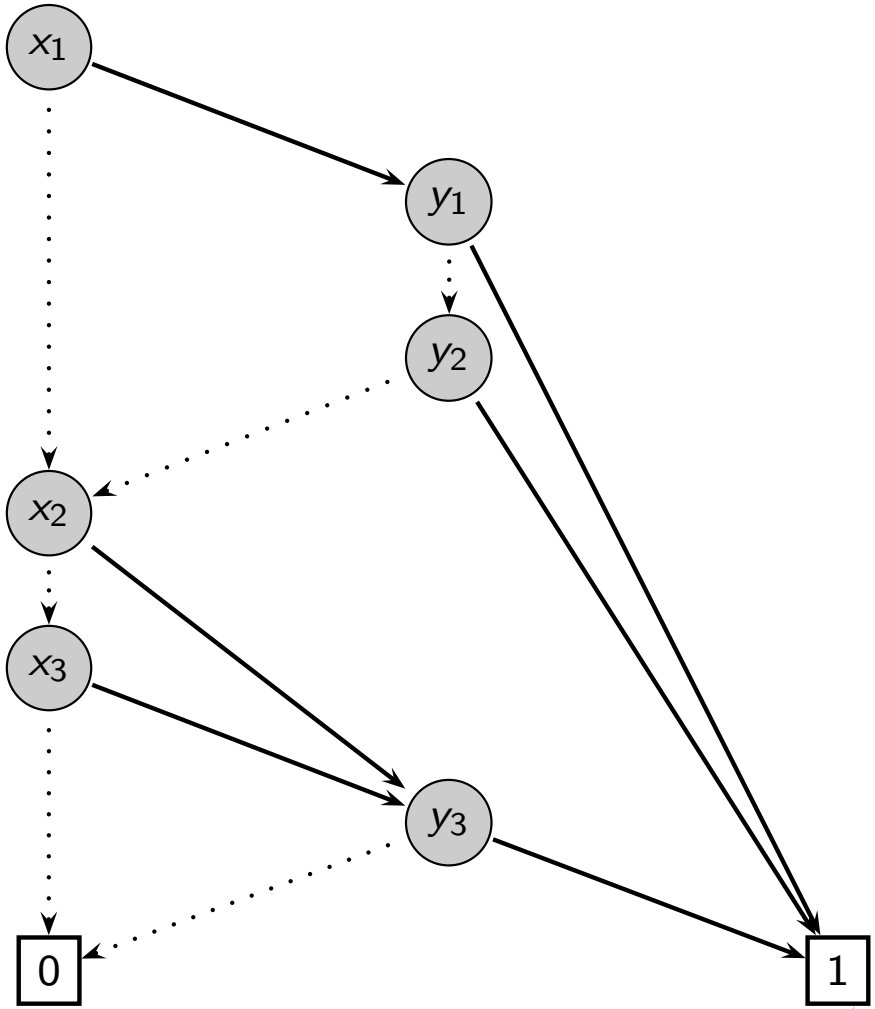


Compilation example

Task: Construct OBDD (ϕ, π), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- $\pi = x_1y_1y_2x_2x_3y_3$.

Step 10 (final): Merge leaves with the same expression (0 or 1).



Compilation example: Summing Up

OBDD (ϕ, π) has size bounded in the number of literals in ϕ.
(exactly one node per variable in ϕ in our example)

Questions

1. Is this property shared by the BDDs of many queries?
2. Can we directly and efficiently construct such succinct BDDs?
3. Can we efficiently find such *good* variable orders?

Compilation example: Summing Up

OBDD (ϕ, π) has size bounded in the number of literals in ϕ.
(exactly one node per variable in ϕ in our example)

Questions

1. Is this property shared by the BDDs of many queries?
2. Can we directly and efficiently construct such succinct BDDs?
3. Can we efficiently find such *good* variable orders?

The answer is in the affirmative for all of the three questions!

Quiz: BDDs

Task: Construct OBDD (ϕ, π'), where

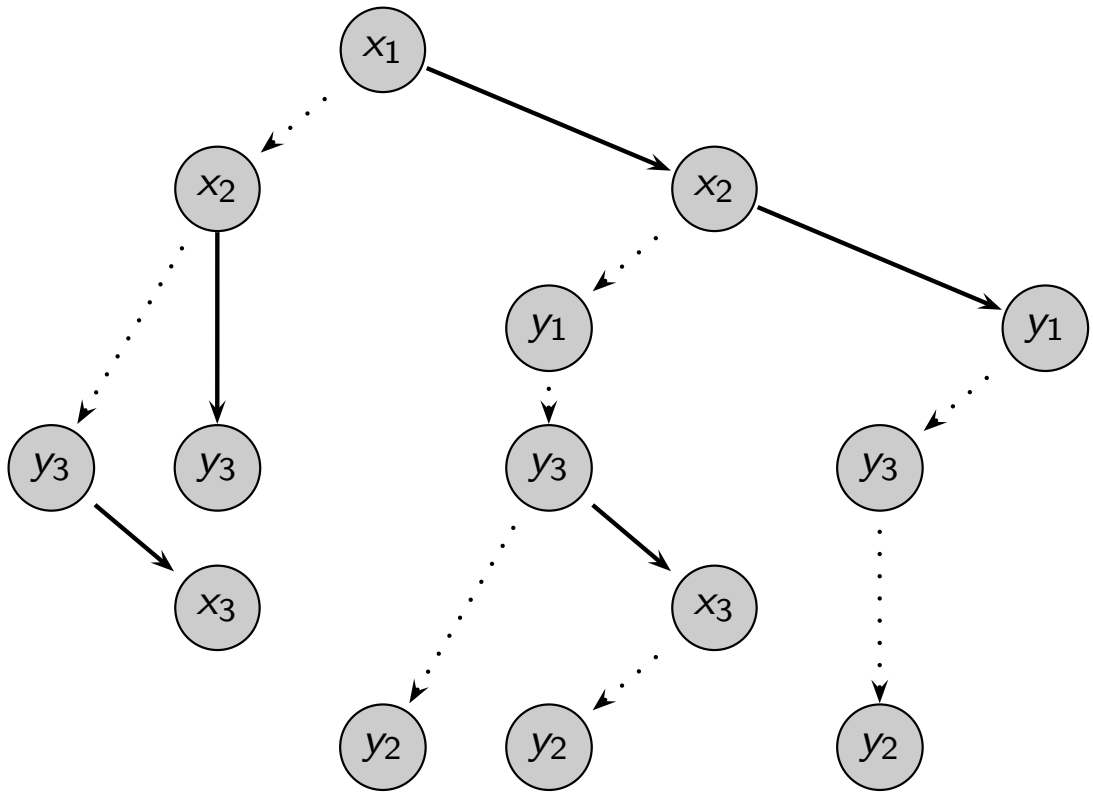
- $\phi = x_1 y_1 + x_1 y_2 + x_2 y_3 + x_3 y_3$ and
- π' is a suboptimal order for ϕ (and find such an order)

Answer

Task: Construct OBDD (ϕ, π'), where

- $\phi = x_1y_1 + x_1y_2 + x_2y_3 + x_3y_3$ and
- π' is a suboptimal order for ϕ (and find such an order)

$\pi' = x_1x_2y_1y_3x_3y_2$ is suboptimal since, eg, the decision on x_2 does not constrain the values for y_1 and y_2, and hence both true/false cases have to be considered under each branch of x_2. See below (the trivial nodes 0 and 1 are ommitted):



Tractable Queries and Succinct BDDs

O&Huang08: For any CQ^1 query q and database D, $\forall t \in q(D)$, and lineage ϕ_t,

- There is a variable order π computable in time $O(|\phi_t| \cdot \log^2 |\phi_t|)$ such that
- The OBDD (ϕ_t, π) has size $O(|\phi_t|)$ and can be computed in time $O(|\phi_t| \cdot \log |\phi_t|)$.

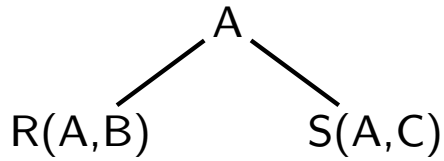
O&Huang09: BDD construction in polynomial time for a large class of tractable conjunctive queries with inequalities.

Good variable orders can be *statically* derived from the query structure!

Static Query Analysis: Query Signatures

Query signatures for TQ queries capture

- the structures of queries and
- the one/many-to-one/many relationships between the query tables;
- variable orders for succinct BDDs representing compiled lineage!



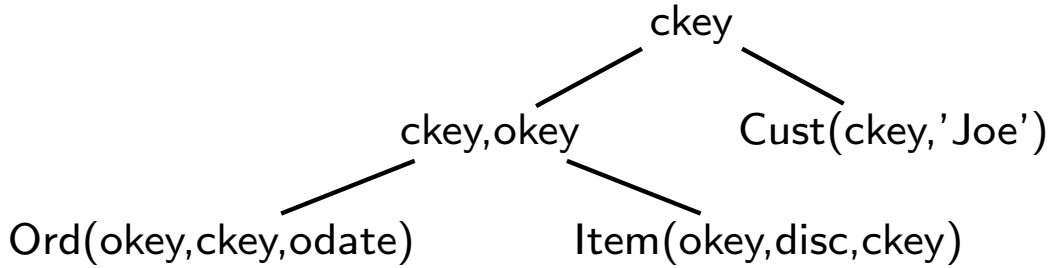
Query q :- $R(A, B), S(A, C)$ has signature $(R^*S^*)^*$.

- There may be several R-tuples with the same A-value, hence R^*
- There may be several S-tuples with the same A-value, hence S^*
- R and S join on A, hence R^*S^*
- There may be several A-values in R and S, hence $(R^*S^*)^*$

Variable orders captured by $(R^*S^*)^*$ (x_i's are from R, y_j's are from S):
$\{[x_1(y_1y_2)][(x_2x_3)y_3]\}$, $\{[(x_2x_3)y_3][x_1(y_1y_2)]\}$, $\{[y_3(x_3x_2)][x_1(y_2y_1)]\}$, etc.

Deriving Better Query Signatures

Q :- Cust($ckey$,$'$ Joe'), Ord($okey$, $ckey$, $odate$), Item($okey$, $disc$, $ckey$), $disc > 0$



Query Q has signature $(\text{Cust}^*(\text{Ord}^*\text{Item}^*)^*)^*$.

Database constraints can make the signature more precise

- If ckey is key in Cust, we obtain the signature $(\text{Cust}(\text{Ord}^*\text{Item}^*)^*)^*$.
  The many-to-many relationship between Cust and Ord is now one-to-many

- If in addition okey is key in Ord, we obtain the signature $(\text{Cust}(\text{Ord Item}^*)^*)^*$.

Query Rewriting under Functional Dependencies (FDs)

FDs on tuple-independent databases can help deriving better query signatures.

Given a set of FDs Σ and a conjunctive query of the form

$$Q = \pi_{\overline{A_0}}(\sigma_\phi(R_1(\overline{A_1}) \bowtie \ldots \bowtie R_n(\overline{A_n}))$$

where ϕ is a conjunction of unary predicates. Let $\Sigma_0 = CLOSURE_\Sigma(\overline{A_0})$. Then, the Boolean query

$$\pi_\emptyset(\sigma_\phi(R_1(CLOSURE_\Sigma(\overline{A_1}) - \Sigma_0) \bowtie \ldots \bowtie R_n(CLOSURE_\Sigma(\overline{A_n}) - \Sigma_0)))$$

is called the **FD-reduct** of Q under Σ.

If there is a sequence of chase steps under Σ that turns Q into a hierarchical query, then the fixpoint of the chase (the FD-reduct) is hierarchical.

Importance of FD-reducts

The signature of Q's FD-reduct captures the structure of Q's lineage.

Two relevant cases

1. Intractable queries may admit tractable FD-reducts.

   Under $X \rightarrow Y$, the hard query Q :- $R(X), S(X,Y), T(Y)$ admits the hierarchical FD-reduct Q' :- $R(X,Y), S(X,Y), T(Y)$ with signature $((RS)^* T)^*$.

2. FD-reducts have more precise query signatures.

   In the presence of keys ckey and okey, the query
   $Q(odate)$:- $\mathsf{Cust}(ckey, cname), \mathsf{Ord}(okey, ckey, odate), \mathsf{Item}(okey, disc, ckey)$
   with signature $(\mathsf{Cust}^*(\mathsf{Ord}^*\mathsf{Item}^*)^*)^*$ rewrites into

   Q' :- $\mathsf{Cust}(ckey, cname), \mathsf{Ord}(okey, ckey, cname), \mathsf{Item}(okey, disc, ckey, cname)$
   with signature $(\mathsf{Cust}(\mathsf{Ord}\ \mathsf{Item}^*)^*)^*$.

Case Study: TPC-H Queries

Considered the conjunctive part of each of the 22 TPC-H queries

- Boolean versions (B)
- with original selection attributes, but without aggregates (O)

Hierarchical in the absence of key constraints

- 8 queries (B)
- 13 queries (O)

Hierarchical in the presence of key constraints

- 8+4 queries (B)
- 13+4 queries (O)

In-depth study at

`http://www.comlab.ox.ac.uk/people/dan.olteanu/papers/icde09queries.html`

Secondary-storage Query Evaluation

Query evaluation in two logically-independent steps

1. Compute query answer using a good *relational* query plan of your choice
2. Compute probabilities of each distinct answer (or temporary) tuple

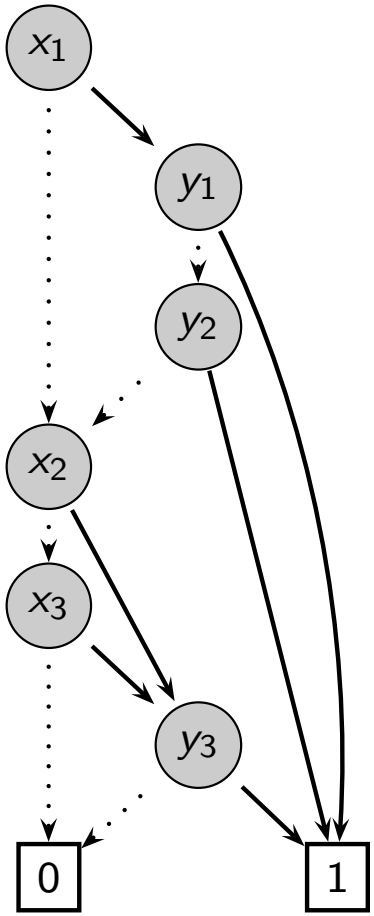
Probability computation supported by a new aggregation operator that can

- blend itself in any relational query plan
- be placed on top of the query plan, or *partially* pushed down past joins
- compute in parallel different fragments of the BDD for the lineage *without* materializing the BDD.

Our aggregation operator is a sequence of

- **aggregation** steps. Effect on query signature: $\alpha^* \rightarrow \alpha$
- **propagation** steps. Effect on query signature: $\alpha\beta \rightarrow \alpha$

Example of Probability Computation



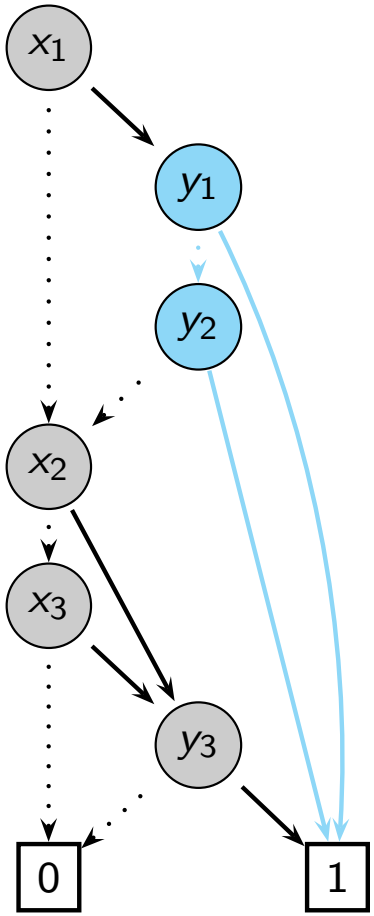
q :- $R(A, B), S(A, C)$

V_r	V_s
x_1	y_1
x_1	y_2
x_2	y_3
x_3	y_3

How to proceed?

1. **Sort query answer by** (V_r, V_s)**.**
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



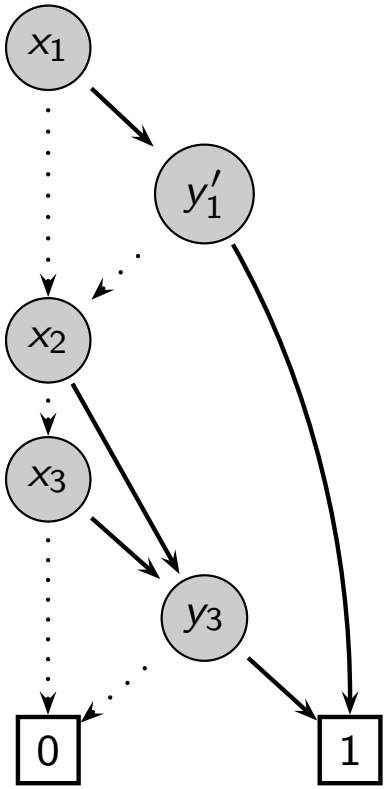
q :- $R(A, B), S(A, C)$

V_r	V_s
x_1	$y_1 + y_2$
x_2	y_3
x_3	y_3

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. **Apply aggregation step $S^* \longrightarrow S$.**
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



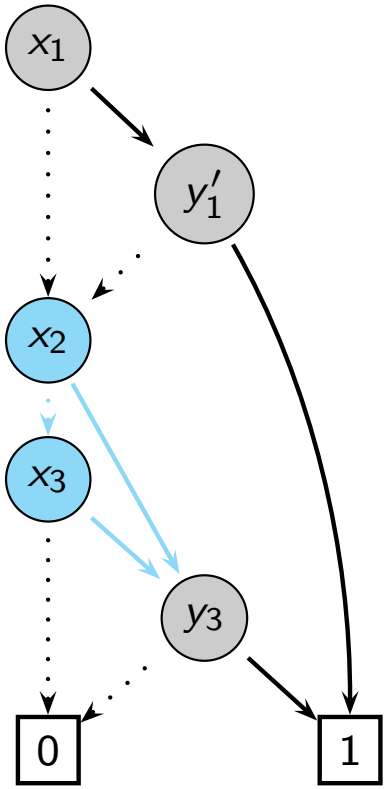
q :- $R(A, B), S(A, C)$

V_r	V_s
x_1	y'_1
x_2	y_3
x_3	y_3

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. **Apply aggregation step $S^* \longrightarrow S$.**
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



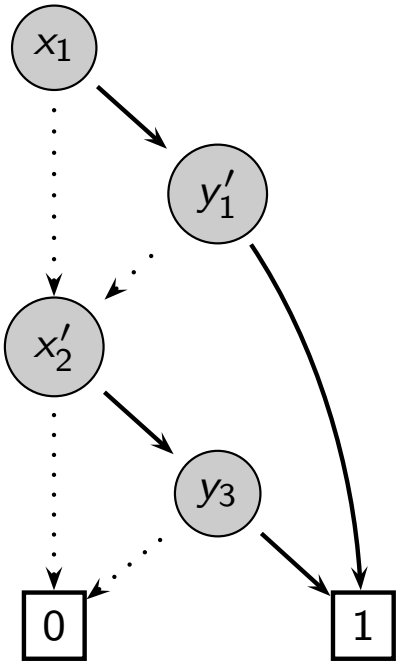
q :- $R(A, B), S(A, C)$

V_r	V_s
x_1	y'_1
$x_2 + x_3$	y_3

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. **Apply aggregation step** $R^* \longrightarrow R$**.**
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



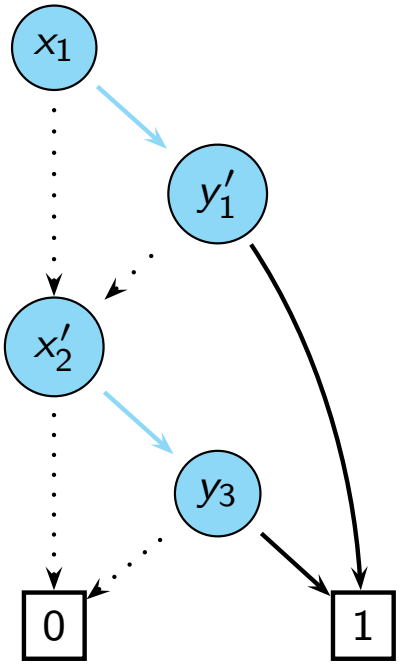
q :- $R(A, B), S(A, C)$

	V_r	V_s
	x_1	y_1'
	x_2'	y_3

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. **Apply aggregation step $R^* \longrightarrow R$.**
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



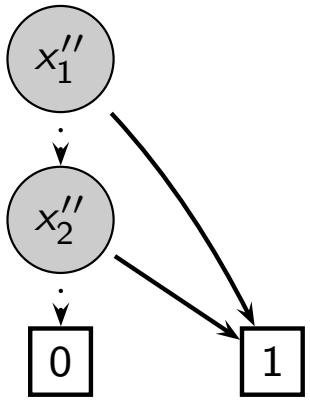
q :- $R(A, B), S(A, C)$

	V_r
	$x_1 y_1'$
	$x_2' y_3$

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. **Apply propagation step** $RS \longrightarrow R$**.**
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation



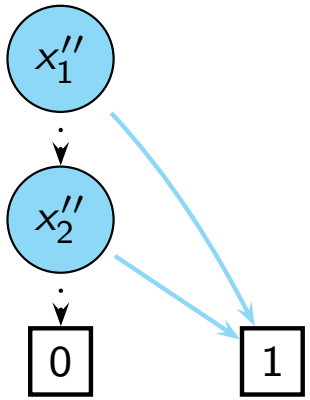
q :- $R(A, B), S(A, C)$

V_r
x_1''
x_2''

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. **Apply propagation step** $RS \longrightarrow R$**.**
   New signature: R^*
5. Apply aggregation step $R^* \longrightarrow R$.
   New signature: R

Example of Probability Computation

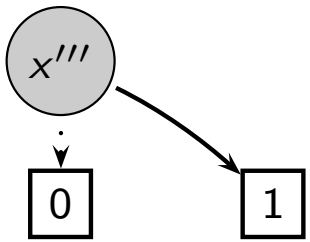


q :- $R(A,B), S(A,C)$
V_r
$x_1'' + x_2''$

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. **Apply aggregation step $R^* \longrightarrow R$.**
   New signature: R

Example of Probability Computation



q :- $R(A, B), S(A, C)$

V_r
x'''

Return the probability of x'''.

How to proceed?

1. Sort query answer by (V_r, V_s).
   Initial signature: $(R^* S^*)^*$
2. Apply aggregation step $S^* \longrightarrow S$.
   New signature: $(R^* S)^*$
3. Apply aggregation step $R^* \longrightarrow R$.
   New signature: $(RS)^*$
4. Apply propagation step $RS \longrightarrow R$.
   New signature: R^*
5. **Apply aggregation step $R^* \longrightarrow R$.**
   New signature: R

Grouping Aggregations and Propagations

Groups of aggregations/propagations can be computed in **one scan**.

Definition: A signature has the *1scan* property if each of its composite expressions is made up by concatenating signatures with the 1scan property and at least one table without (*).

Examples of 1scan signatures:

- $(RS^*)^*$ (last 3 steps in the previous example)
- R^*S^* (relational product)
- $\text{Nation}_1\text{Supp}(\text{Nation}_2(\text{Cust}(\text{Ord Item}^*)^*)^*)^*$ (conj. part of TPC-H query 7)

For signature α: $\#\text{scans}(\alpha) =$ one plus the number of its starred (*) subexpressions, including itself, without the 1scan property.

Proposition: An operator with signature α needs $\#\text{scans}(\alpha)$ scans.

Examples:

- $\#\text{scans}((R^*S^*)^*) = 2$
- $\#\text{scans}((\text{Cust}^*(\text{Ord}^*\text{Item}^*)^*)^*) = 3$, BUT $\#\text{scans}((\text{Cust}(\text{Ord Item}^*)^*)^*) = 1$

Can we leverage existing results on BDD construction?

- Generic AI compilation techniques construct BDDs whose sizes are exponential in the treewidth of the lineage [Huang&Darwiche01]

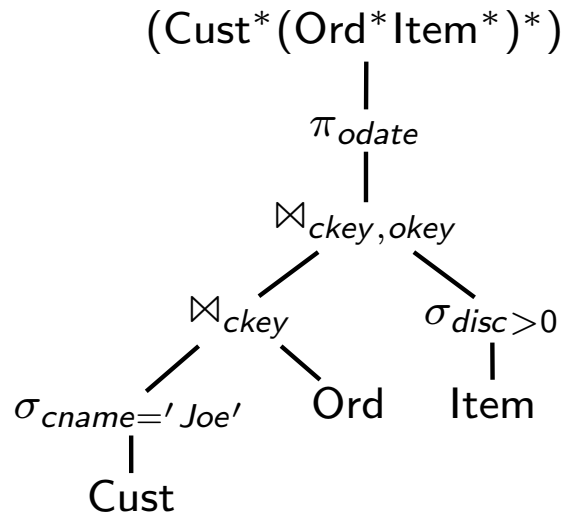
- Conjunctive queries **do** generate lineage of unbounded treewidth.
  - The product query Q :- $R(X), S(Y)$ generates lineage that has a clause for each pair of random variables of R and S $\Rightarrow$ unbounded treewidth.

- Reconciling the two techniques [Jha&**O**&Suciu10]:
  - Partition input query+data into a tractable subinstance and a (usually much smaller) hard subinstance.
  - Apply scalable database-specific techniques to the tractable part and generic compilation techniques to the hard part.

Query Optimization: Types of Query Plans

Our previous examples considered *lazy* plans

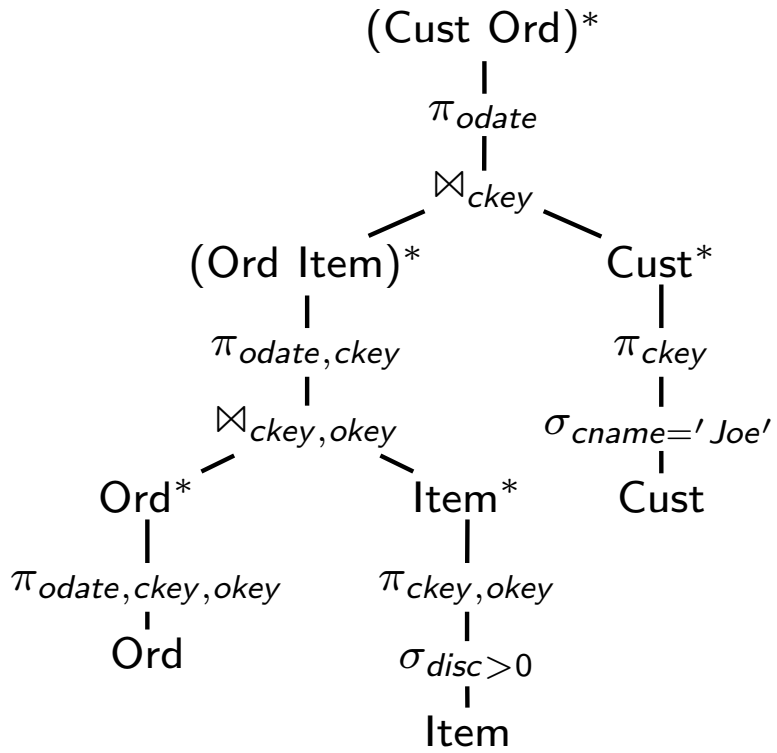
- probability computation done *after* the computation of answer tuples
- unrestricted search space for good query plans
- especially desirable when join conditions are selective (eg, TPC-H)!



BUT, we can push down probability computation!

Query Optimization: Types of Query Plans

Eager plans discard duplicates and compute probabilities on each temporary table.

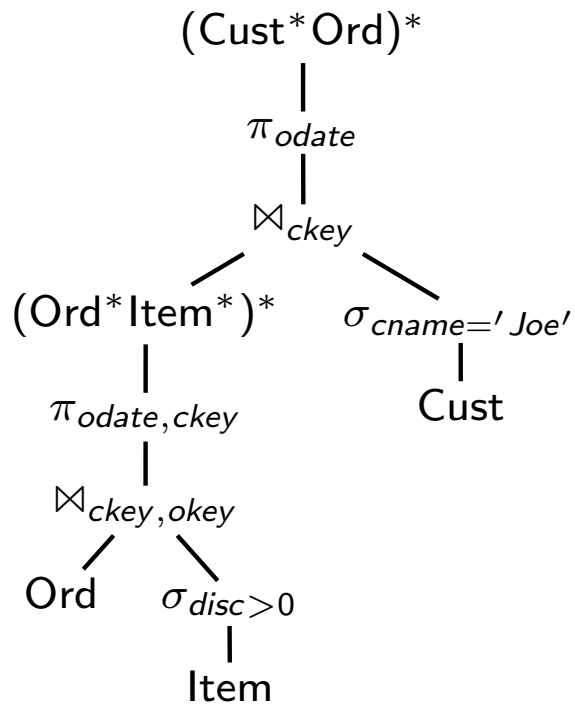


MystiQ's safe plans are special cases of eager plans!

- mirror the hierarchical structure of the query signature
- probability computation restricts join ordering!
- suboptimal join ordering, which is more costly than probability computation

Quiz: Query Optimization

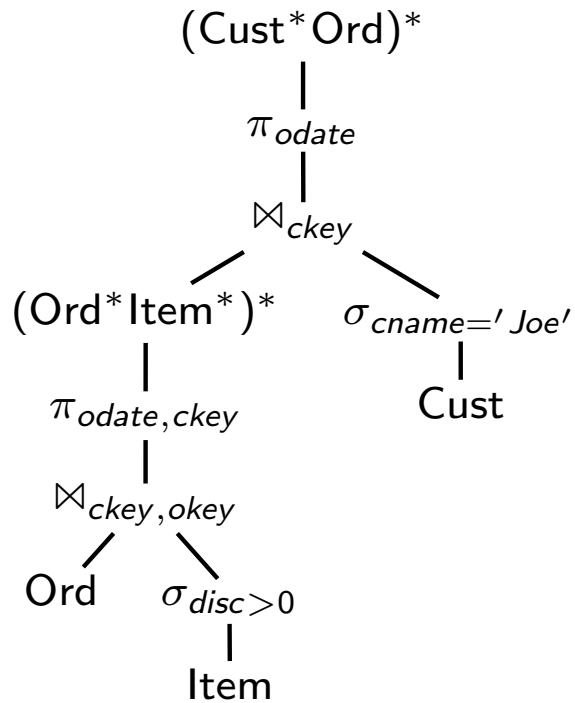
Is the following a valid query plan?



Answer

YES! It is a *hybrid* plan

- useful when selectivities of different joins differ significantly
- push down probability computation below unselective joins
- keep probability computation on top of selective joins



Approximation Algorithms

- Randomized
  - Naïve Monte Carlo simulation
  - Improved Monte Carlo

- Deterministic
  - Algorithmic guarantees (polynomial time, but with prohibitively large constants) [Trevisan'04], [Luby&Velickovic'91]
  - Incremental compilation scheme with polynomial time guaranteess for known tractable queries [O.,Huang,Koch'10]

Naïve Monte Carlo

Input: Boolean formula ϕ with variables $V(\phi)$

$Cnt \leftarrow 0$

repeat N times

  randomly choose a total valuation λ over $V(\phi)$

  if $\lambda(\phi) =$ true then $Cnt = Cnt + 1$

$P = Cnt/N$

return P/* $\approx Pr(\phi)$*/

If $N \geq (1/Pr(\phi)) \times (4ln(2/\delta)/\epsilon^2)$ then $Pr[\mid P/Pr(\phi) - 1 \mid > \epsilon] < \delta$.

Improved Monte Carlo

[KL83], [Graedel,Gurevitch, Hirsch98]

Input: Boolean formula in DNF $\phi = C_1 + \cdots + C_m$ with variables $V(\phi)$

$Cnt \leftarrow 0; S \leftarrow Pr(C_1) + \cdots + Pr(C_m)$

repeat N times

randomly choose $1 \leq i \leq m$ with probability $Pr(C_i)/S$

randomly choose a total valuation λ over $V(\phi)$ such that $\lambda(C_i) =$ true

if $\forall 1 \leq j < i : \lambda(C_j) =$ false then $Cnt = Cnt + 1$

$P = Cnt/N \times S/2^{|V(\phi)|}$

return P/* $\approx Pr(\phi)$*/

If $N \geq (1/m) \times (4ln(2/\delta)/\epsilon^2)$ then $Pr[\mid P/Pr(\phi) - 1 \mid > \epsilon] < \delta$.

- Slightly modified algorithms are used in MayBMS, MystiQ, and MCDB.
- Veeeery sloooow in practice: SPROUT query plans for tractable queries are about two orders of magnitude faster that the improved (and optimized!) Monte Carlo.

Approximate Evaluation with SPROUT

- Monte Carlo simulations are very powerful and generic
  only require sampling the formula, no knowledge of its structure

- Why not exploit the structure of the input formula?
  - compile it into an equivalent decomposed form that allows for efficient probability computation
  - in practice, good approximations are obtained after a few decomposition steps

D-trees: Decomposition Trees of DNF Formulas

Given DNF formula Φ.

- Independent-or $\otimes$: Partition Φ into independent DNFs $\Phi_1, \Phi_2 \subset \Phi$ such that Φ is equivalent to $\Phi_1 \vee \Phi_2$.
- Independent-and $\odot$: Partition Φ into independent DNFs $\Phi_1, \Phi_2 \subset \Phi$ such that Φ is equivalent to $\Phi_1 \wedge \Phi_2$.
- Exclusive-or $\oplus$: Choose a variable x in Φ. Replace Φ by

$$\bigoplus_{a \in \mathrm{Dom}_x, \Phi|_{x=a} \neq \emptyset} \left(\{\{x = a\}\} \odot \Phi \mid_{x=a} \right)$$

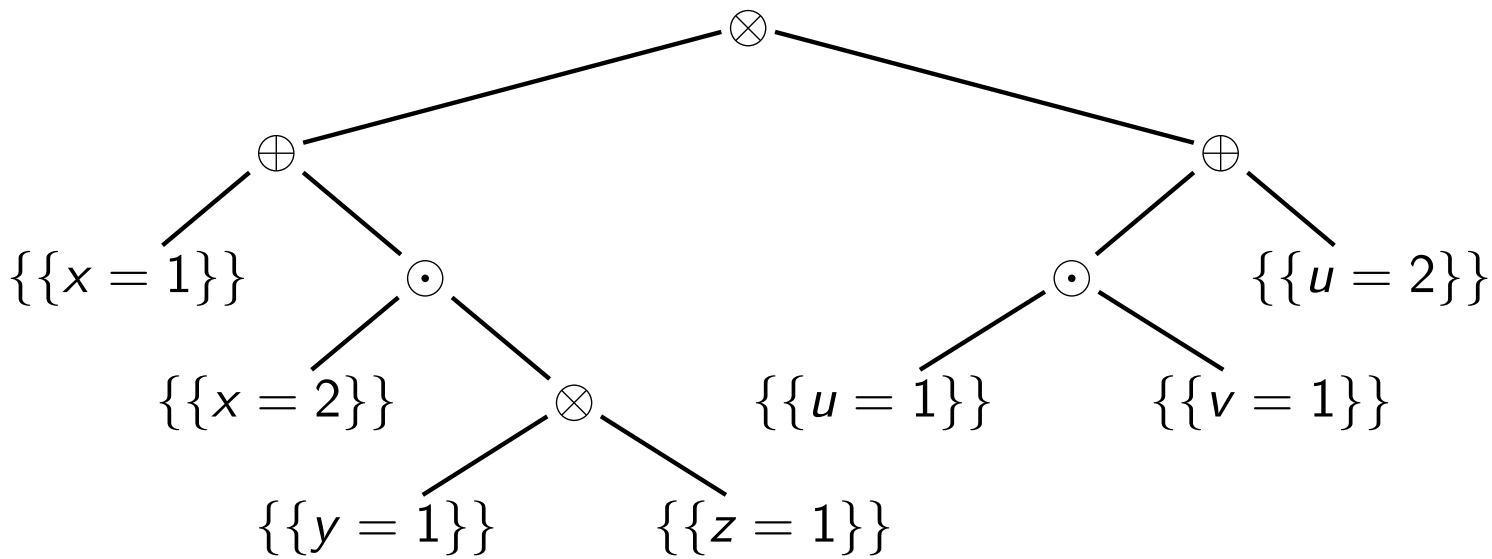
The decompositions preserve equivalence and are efficient for query lineage.

A *d-tree* is a formula constructed from $\otimes$, $\oplus$, $\odot$ and nonempty DNFs (as "leaves"). If each leaf holds one clause, the d-tree is *complete*.

D-tree Example

D-tree for DNF

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



Approximation with error guarantees

Underlying ideas [**O**&Huang&Koch10]:

- Incremental refinement of the leaves of a d-tree for a DNF Φ
  - stop when desired approximation is reached

- Fast approximation of probabilities at d-tree leaves
  - Choose a maximal subset S of pairwise independent clauses in leaf Ψ
  - $P(S)$ is a lower bound for $P(\Psi)$
  - $\min(1, P(S) + \sum_{c \in (\Psi - S)} (P(c)))$ is an upper bound for $P(\Psi)$

- Once lower and upper bounds are known at each leaf, the bounds of the entire d-tree can be computed very efficiently

- Additional property of d-tree compilation:
  Lineage of CQ^1 queries can be compiled efficiently into complete d-trees.

Literature on Probabilistic Relational Data

[AJKO08] Antova, Jansen, Koch, Olteanu. *Fast and Simple Relational Processing of Uncertain Data*. ICDE 2008.

[DS07] Dalvi, Suciu *Efficient Query Evaluation on Probabilistic Databases*. VLDBJ 2007.

[DS07b] Dalvi, Suciu. *The Dichotomy of Conjunctive Queries on Probabilistic Structures*. PODS 2007.

[DS07c] Dalvi, Suciu. *Management of Probabilistic Data: Foundations and Challenges*. PODS 2007.

[KLM89] Karp, Luby, Madras. *Monte-Carlo Approximation Algorithms for Enumeration Problems*. J. Algorithms 1989.

[OH08] Olteanu, Huang. *Using OBDDs for Efficient Query Evaluation on Probabilistic Databases*. SUM 2008.

Literature on Probabilistic Relational Data

[OH09] Olteanu, Huang. *Secondary-Storage Confidence Computation for Conjunctive Queries with Inequalities*. SIGMOD 2009.

[OHK09] Olteanu, Huang, Koch. *SPROUT: Lazy vs. Eager Query Plans for Tuple-Independent Probabilistic Databases*. ICDE 2009.

[OHK10] Olteanu, Huang, Koch. *Approximate Confidence Computation in Probabilistic Databases*. ICDE 2010.

[RDS07] Ré, Dalvi, Suciu. *Efficient Top-k Query Evaluation on Probabilistic Data*. ICDE 2007.

[SD07] Sen, Deshpande. *Representing and Querying Correlated Tuples in Probabilistic Databases*. ICDE 2007.

[Val79] Valiant. *The Complexity of Enumeration and Reliability Problems*. SIAM J. Comput. 1979.

3. Probabilistic XML Data

Outline of Part 3

Same topics as in the relational case

- Plethora of data models
  Uncertainty constructs of these models very much resemble those from the relational case.

- Expressiveness and succinctness of these models

- Query Evaluation
  Tractability for twig queries over various models

Probabilistic XML Data Models

P-document = unordered tree with two types of nodes

1. ordinary nodes (like in standard XML trees)
2. **distributional** nodes
   only used to define probability distributions over subsets of their children.

How to generate a random document?

(a1) Each distributional node randomly chooses a subset of its children
(for some nodes, the choices are not necessarily independent)

(a2) All of the unchosen nodes and their descendants are deleted

(b) Remove all distributional nodes and connect ordinary nodes to their closest ordinary ancestor

Types of Distributional Nodes

1. det : always (nondeterministically) choose all children
2. ind : independent choice of children
3. mux : mutual exclusive choice of children
4. exp : explicit specification of the probability distribution over subsets of children

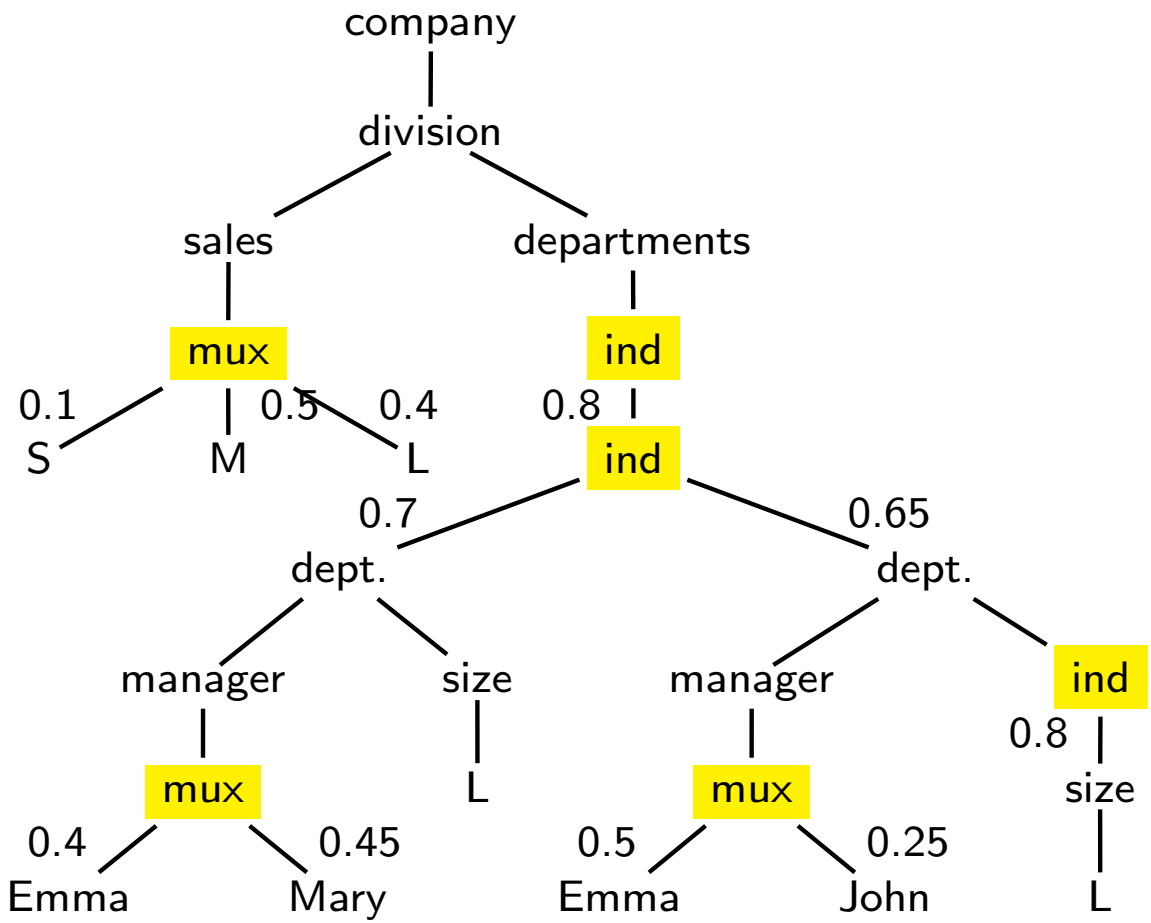
Underlying assumption for all types above: Choices of different distributional nodes are probabilistically independent.

5. cie : each node can be associated with a conjunction of (possibly negated) independent event variables.

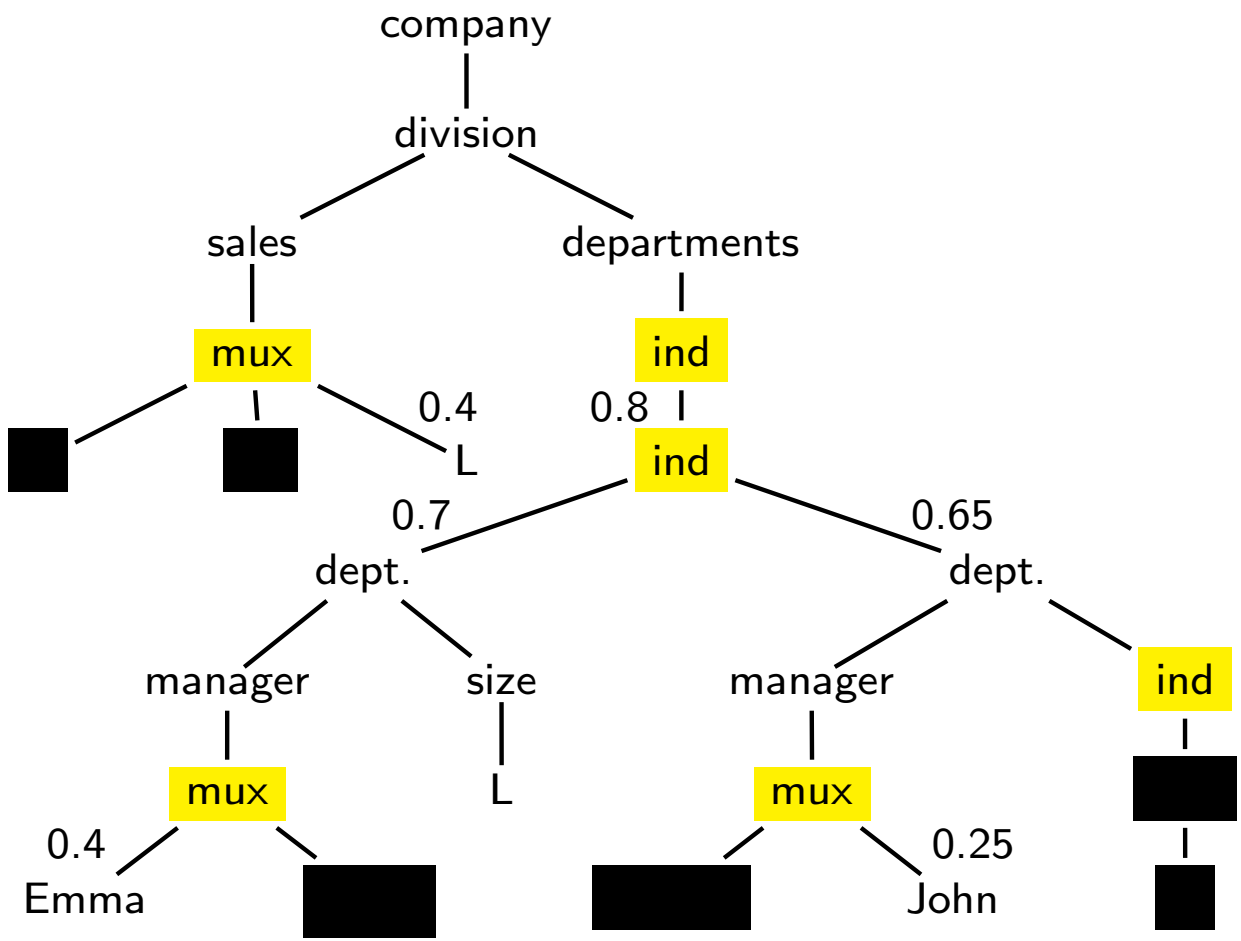
How to compute the probability of a possible world in a p-document?

- without cie nodes: Product of probabilities of choices made at each distributional node (P).
- with cie nodes: $P\times$ probability of a total valuation over cie variables that defines that world.

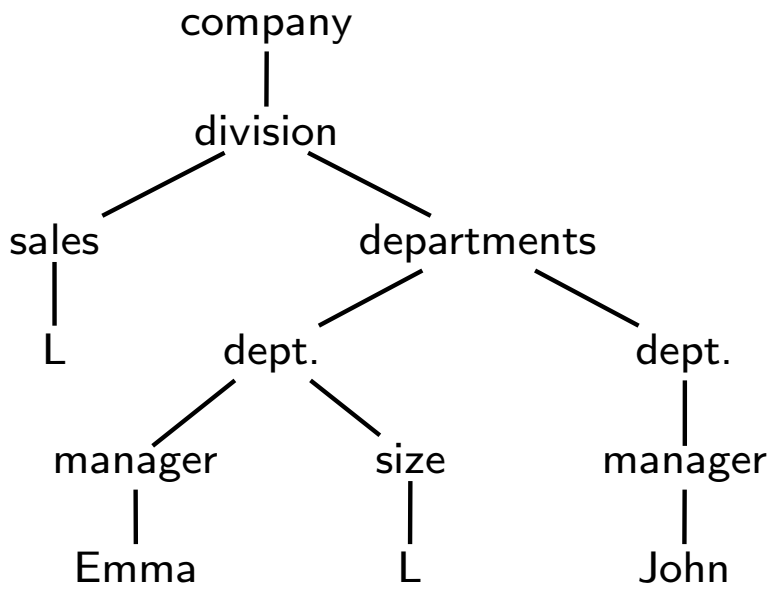
Example of a p-document [AKSS09]



Choosing a world defined by our p-document



Choosing a world defined by our p-document



Expressiveness and Succinctness [AKSS09]

Plethora of families of p-documents:

- $\text{PrXML}^{t_1,\ldots,t_n}$ = those p-documents with distributional nodes $t_1, \ldots, t_n$ only.
- (dis)allow hierarchies of distributional nodes.

Some results

- $\text{PrXML}^{\text{mux}}$ is complete
- $\text{PrXML}^{\text{ind,mux}}$(non-hier) can be translated efficiently to $\text{PrXML}^{\text{mux}}$
- $\text{PrXML}^{\text{ind,mux}}$ can be translated efficiently to $\text{PrXML}^{\text{mux,det}}$
- ...

Open: Is $\text{PrXML}^{\text{exp}}$ efficiently translatable into $\text{PrXML}^{\text{mux,det}}$?

Query Evaluation

Considered queries:

- Syntax: Twig patterns with child/descendant edges and unary conditions at nodes.
- Semantics: Evaluate the twig in each world and return the set of all possible answers

Matches of twigs in trees = mappings that

- are root-preserving,
- are structure-preserving (from pattern to p-document nodes), and
- satisfy all node conditions.

Query evaluation task now also computes probabilities of query answers!

- Compute a function p over all the mappings μ such that $p(\mu)$ is the probability that μ is a match of the input twig in a random world.
- Set of answers includes all μ such that $p(\mu) > 0$.

Major challenge: Handling queries with projection!

Complexity Results for Query Evaluation [KS08]

Query& data complexity: NP-hard to determine whether a twig (without projection) evaluates to a nonempty result in $\text{PrXML}^{\text{mux}}$.

Data complexity

- Every *nontrivial* Boolean twig over $\text{PrXML}^{\text{cie}}$ is #P-complete.

- Efficient evaluation of
  - Every Boolean twig over $\text{PrXML}^{\text{exp}}$ or $\text{PrXML}^{\text{ind,mux}}$.
  - Twigs nested with count (or min, max) aggregates.

Open issues:

- Are there natural restrictions of $\text{PrXML}^{\text{cie}}$ beyond $\text{PrXML}^{\text{mux,ind}}$ with efficient query evaluation?
- What happens beyond twigs extended with count aggregates?

Last Quiz: Stochastic Context-free Grammars (SCFGs)

A SCFG is a CFG where there is a probability distribution over the choices in the rhs of each rule. Example:

$$S \rightarrow aSa(0.6) \mid bSb(0.3) \mid \epsilon(0.1)$$

Consider the problem of computing the probability that a word is in the generated language (can be casted as a trivial query evaluation problem on SCFGs).

Questions:

- How many bits can such probabilities require?
- Can these probabilities be irrational numbers?

K. Etessami and M. Yannakakis. "Recursive Markov Chains, Stochastic Grammars, and Monotone Systems of Nonlinear Equations". JACM: 56(1), 2009.

Answer (1)

The probabilities may require an exponential number of bits (in the size of the SCFG).

$$S_i \rightarrow S_{i-1}S_{i-1}, \quad 1 \leq i \leq n$$
$$S_0 \rightarrow \epsilon(1/2) \mid a(1/2)$$

The probability of producing the empty string from nonterminal S_i is 2^{-2^i}.

Answer (2)

The probabilities can be irrational numbers.

$$S \rightarrow \epsilon(1/2) \mid b(2/6) \mid SSSSS(1/6)$$

The probability of producing the empty string from nonterminal S is x with $x = 1/2 + 1/6x^5$. This polynomial has no rational roots.

Literature on Probabilistic XML Data

[AKSS09] Abiteboul, Kimelfeld, Senellart, Sagiv. *On the Expressiveness of Probabilistic XML Models*. VLDBJ 2009.

[KS07] Kimelfeld, Sagiv. *Matching Twigs in Probabilistic XML*. VLDB 2007.

[KSS08] Kimelfeld, Kosharovsky, Sagiv. *Query Efficiency in Probabilistic XML Models*. SIGMOD 2008.

[SA07] Senellart, Abiteboul. *On the Complexity of Managing Probabilistic XML Data*. PODS 2007.