Scalable Uncertainty Management

01 – Introduction

Rainer Gemulla

April 20, 2012

Information & Knowledge Management Circa 1988

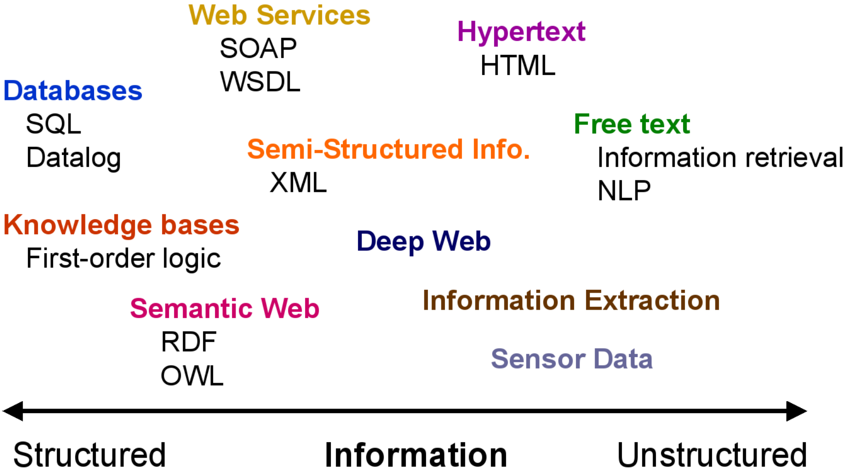
Databases
SQL
Datalog

Knowledge bases
First-order logic

Free text
Information retrieval
NLP

Structured **Information** Unstructured

Information & Knowledge Management Today



Web Services
SOAP
WSDL

Hypertext
HTML

Databases
SQL
Datalog

Free text
Information retrieval
NLP

Semi-Structured Info.
XML

Knowledge bases
First-order logic

Deep Web

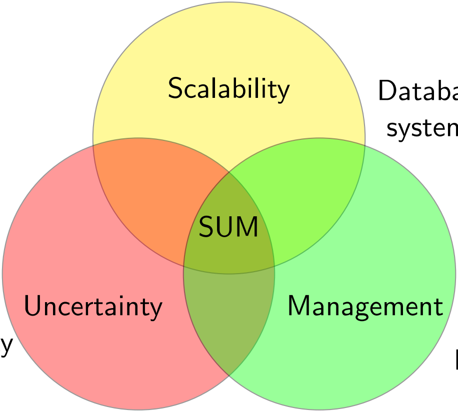
Information Extraction

Semantic Web
RDF
OWL

Sensor Data

Structured **Information** Unstructured

Overview



Distributed systems

Scalability

Database systems

SUM

Uncertainty

Management

Probability theory

Logic

Artificial intelligence
Machine learning

SUM is about managing large amounts of uncertain data.

Outline

1. Uncertainty in the Real World

2. Managing Uncertainty

Sources of uncertainty

Certain data	**Uncertain data**	
The temparature is 25.634589 °C.	Sensor reported 25 ± 1 °C.	Precision of devices
Bob works for Yahoo.	Bob works for Yahoo or Microsoft.	Lack of information
MPII is located in Saarbrücken.	MPII is located in Saarland.	Coarse-grained information
Mary sighted a finch.	Mary sighted either a finch (80%) or a sparrow (20%).	Ambiguity
It will rain in Saarbrücken tomorrow.	There is a 60% chance of rain in Saarbrücken tomorrow.	Uncertainty about future
John's age is 23.	John's age is in [20,30].	Anonymization
Paul is married to Amy.	Paul is married to Amy. Amy is married to Frank.	Inconsistent data

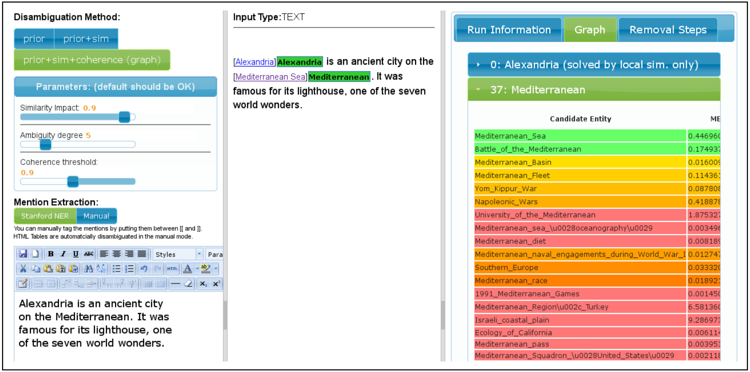
Where does uncertainty arise?

Everywhere!

- Information extraction (D5 research)
- Sensor networks
- Business intelligence & predictive analytics
- Forecasting
- Scientific data management
- Privacy preserving data mining
- Data integration
- Data deduplication
- Social network analysis

Entity disambiguation (AIDA)

Disambiguate each mention of an entity in a piece of text.



Disambiguation Method:

Similarity Impact: 0.9

Ambiguity degree 5

Coherence threshold: 0.9

Mention Extraction:

You can manually tag the mentions by putting them between [[and]].
HTML Tables are automatcially disambiguated in the manual mode.

Alexandria is an ancient city on the Mediterranean. It was famous for its lighthouse, one of the seven world wonders.

Input Type:TEXT

[Alexandria] **Alexandria** is an ancient city on the [Mediterranean Sea] **Mediterranean**. It was famous for its lighthouse, one of the seven world wonders.

Candidate Entity	ME
Mediterranean_Sea	0.44696
Battle_of_the_Mediterranean	0.17493
Mediterranean_Basin	0.01600
Mediterranean_Fleet	0.11436
Yom_Kippur_War	0.08780
Napoleonic_Wars	0.41887
University_of_the_Mediterranean	1.87532
Mediterranean_sea_\u0028oceanography\u0029	0.00349
Mediterranean_diet	0.00818
Mediterranean_naval_engagements_during_World_War_I	0.01274
Southern_Europe	0.03332
Mediterranean_race	0.01892
1991_Mediterranean_Games	0.00145
Mediterranean_Region\u002c_Turkey	6.58136
Israeli_coastal_plain	9.28697
Ecology_of_California	0.00611
Mediterranean_pass	0.00395
Mediterranean_Squadron_\u0028United_States\u0029	0.00211

Example

- Find web pages concerning “The King of Rock’n’Roll” (*entity search*)
- How much fuzz about “Santorum” in each month of 2012? (*entity tracking*)

AIDA website

Text segmentation

Segment a piece of text into fields. E.g., “52-A Goregaon West Mumbai 400 062”.

Id	House no	
1	52	C
1	52-A	
1	52-A	C
1	52	

The YAGO-NAGA approach to knowledge discovery [3 citations — 0 self]

by Gjergji Kasneci , Maya Ramanath , Fabian Suchanek , Gerhard Weikum
SIGMOD Rec
Add To MetaCart

DOWNLOAD:
http://suchanek.name/work/publications/sigmodrec20
CACHED:

Abstract:

This paper gives an overview on the YAGO-NAGA approach to information extraction for building a conveniently searchable, large-scale, highly accurate knowledge base of common facts. YAGO harvests infoboxes and category names of Wikipedia for facts about individual entities, and it reconciles these with the taxonomic backbone of WordNet in order to ensure that all entities have proper classes and the class system is consistent. Currently, the YAGO knowledge base contains about 19 million instances of binary relations for about 1.95 million entities. Based on intensive sampling, its accuracy is estimated to be above 95 percent. The paper presents the architecture of the YAGO extractor toolkit, its distinctive approach to consistency checking, its provisions for maintenance and further growth, and the query engine for YAGO, coined NAGA. It also discusses ongoing work on extensions towards integrating fact candidates extracted from natural-language text sources. 1.

Citations

155 Two-stage language models for information retrieval – Zhai, Lafferty - 2002

149 Unsupervised named-entity extraction from the web: an experimental study – Etzioni, Cafarella, et al. - 2005

106 DBpedia: A Nucleus for a Web of Open Data – Auer, Bizer, et al. - 2008

Add a tag:

No tags have been applied to this document.

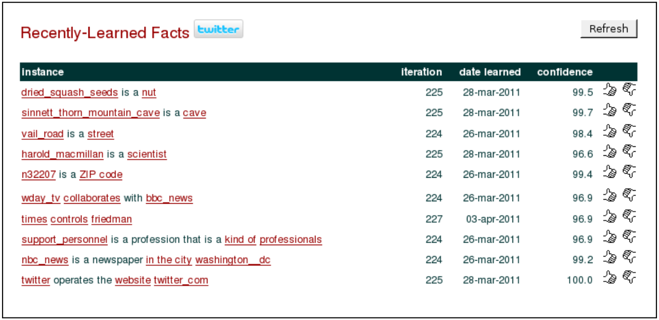
```
@ARTICLE{Kasneci_theyago-naga,
   author = {Gjergji Kasneci and Maya Ramanath
and Fabian Suchanek and Gerhard Weikum},
   title = {The YAGO-NAGA approach to knowledge
discovery},
   journal = {SIGMOD Rec},
   year = {},
   pages = {03}
}
```

Example

- Send a promotion to customers in West Mumbai.
- Find all papers containing YAGO in the title (*faceted search*)

Relation extraction (NELL / Yago2)

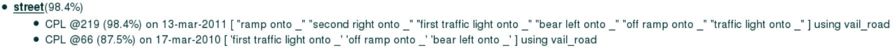
Extract structured relations from the web.



Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence	
dried_squash_seeds is a nut	225	28-mar-2011	99.5	👍 👎
sinnett_thorn_mountain_cave is a cave	225	28-mar-2011	99.7	👍 👎
vail_road is a street	224	26-mar-2011	98.4	👍 👎
harold_macmillan is a scientist	225	28-mar-2011	96.6	👍 👎
n32207 is a ZIP code	224	26-mar-2011	99.4	👍 👎
wday_tv collaborates with bbc_news	224	26-mar-2011	96.9	👍 👎
times controls friedman	227	03-apr-2011	96.9	👍 👎
support_personnel is a profession that is a kind of professionals	224	26-mar-2011	96.9	👍 👎
nbc_news is a newspaper in the city washington__dc	224	26-mar-2011	99.2	👍 👎
twitter operates the website twitter_com	225	28-mar-2011	100.0	👍 👎

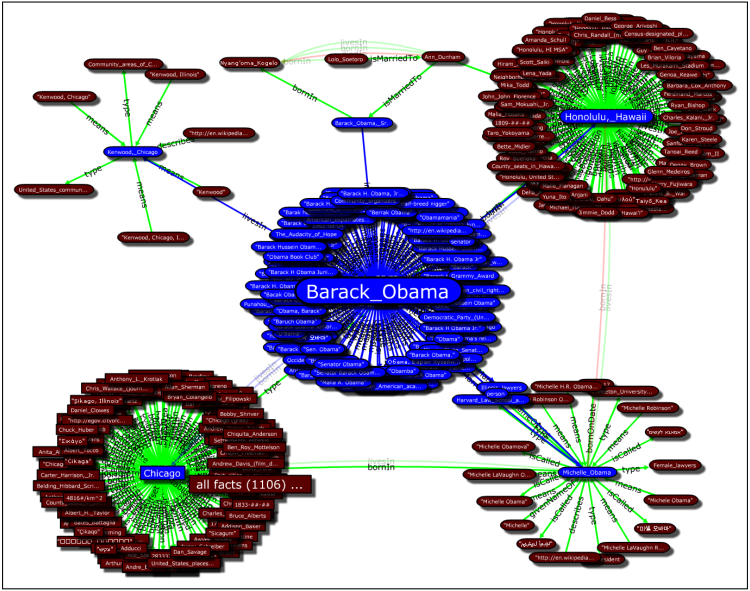


- **street**(98.4%)
  - CPL @219 (98.4%) on 13-mar-2011 ["ramp onto _" "second right onto _" "first traffic light onto _" "bear left onto _" "off ramp onto _" "traffic light onto _"] using vail_road
  - CPL @66 (87.5%) on 17-mar-2010 ['first traffic light onto _' 'off ramp onto _' 'bear left onto _'] using vail_road

Example

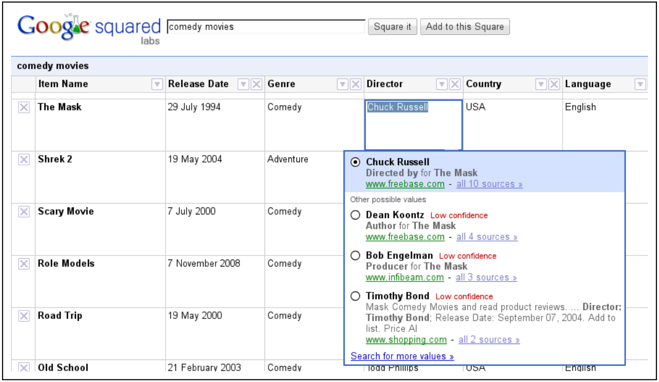
- What is known about Albert Einstein? (*fact search*)
- Who has won a Nobel Prize and is born in Ulm? (*question answering*)

Reasoning with uncertainty (URDF)



Google Squared (discontinued)

Find and describe items of a given category.



comedy movies

Item Name	Release Date	Genre	Director	Country	Language
The Mask	29 July 1994	Comedy	Chuck Russell	USA	English
Shrek 2	19 May 2004	Adventure			
Scary Movie	7 July 2000	Comedy			
Role Models	7 November 2008	Comedy			
Road Trip	19 May 2000	Comedy			
Old School	21 February 2003	Comedy	Todd Phillips	USA	English

● **Chuck Russell**
Directed by for **The Mask**
www.freebase.com - all 10 sources »

Other possible values

○ **Dean Koontz** Low confidence
Author for **The Mask**
www.freebase.com - all 4 sources »

○ **Bob Engelman** Low confidence
Producer for **The Mask**
www.infibeam.com - all 3 sources »

○ **Timothy Bond** Low confidence
Mask Comedy Movies and read product reviews. ... **Director: Timothy Bond**; Release Date: September 07, 2004. Add to list. Price Al
www.shopping.com - all 2 sources »

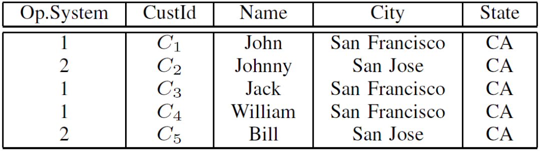
Search for more values »

Example

- Directors that directed at least one comedy movie?
- Birthplaces of directors of comedy movies with a budget of over $20M?

Information integration

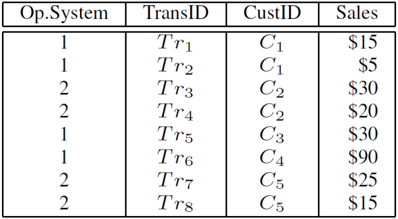
Same?



Op.System	CustId	Name	City	State
1	C_1	John	San Francisco	CA
2	C_2	Johnny	San Jose	CA
1	C_3	Jack	San Francisco	CA
1	C_4	William	San Francisco	CA
2	C_5	Bill	San Jose	CA

(a) Customer Data

Which one?



Op.System	TransID	CustID	Sales
1	Tr_1	C_1	\$15
1	Tr_2	C_1	\$5
2	Tr_3	C_2	\$30
2	Tr_4	C_2	\$20
1	Tr_5	C_3	\$30
1	Tr_6	C_4	\$90
2	Tr_7	C_5	\$25
2	Tr_8	C_5	\$15

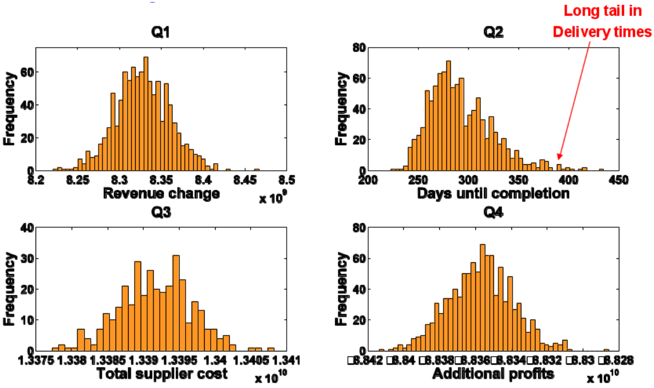
(b) Transaction Data

Example

- Turnover in San Francisco? And in California? (*OLAP*)

Sismanis et al., ICDE09.

Predictive analytics

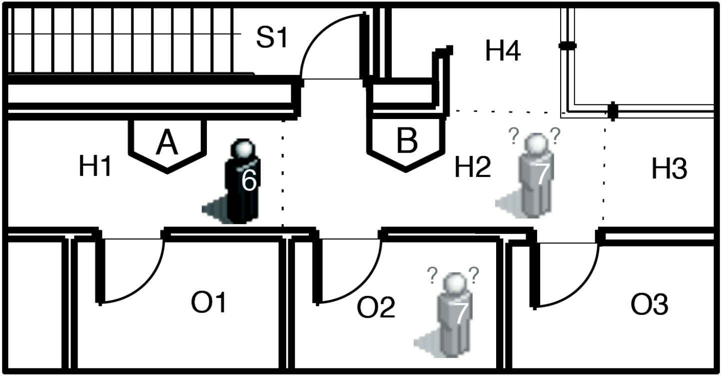

Long tail in Delivery times

Example

- What is the effect of changing the price on future sales?
- What is the risk associated with my portfolio?

Haas, MUD10.

RFID & moving objects

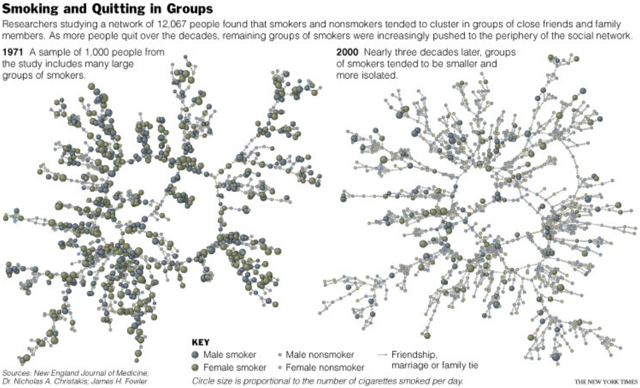


Example

- How many people are attending John's lecture?
- Where are choke points when moving items through my storage facility?

Ré et al., SIGMOD08.

Statistical & uncertain rules



Smoking and Quitting in Groups

Researchers studying a network of 12,067 people found that smokers and nonsmokers tended to cluster in groups of close friends and family members. As more people quit over the decades, remaining groups of smokers were increasingly pushed to the periphery of the social network.

1971 A sample of 1,000 people from the study includes many large groups of smokers.

2000 Nearly three decades later, groups of smokers tended to be smaller and more isolated.

Circle size is proportional to the number of cigarettes smoked per day.

Sources: New England Journal of Medicine; Dr. Nicholas A. Christakis; James H. Fowler

THE NEW YORK TIMES

Example

- Does John smoke? (*social network analysis*)
- "Mississippi" most often refers to the state of Mississippi. (*entity disambiguation*)

Kolata, The New York Times, 2008.

Anonymized data

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	1305*	≤ 40	$*$	Heart Disease
4	1305*	≤ 40	$*$	Viral Infection
9	1305*	≤ 40	$*$	Cancer
10	1305*	≤ 40	$*$	Cancer
5	1485*	> 40	$*$	Cancer
6	1485*	> 40	$*$	Heart Disease
7	1485*	> 40	$*$	Viral Infection
8	1485*	> 40	$*$	Viral Infection
2	1306*	≤ 40	$*$	Heart Disease
3	1306*	≤ 40	$*$	Viral Infection
11	1306*	≤ 40	$*$	Cancer
12	1306*	≤ 40	$*$	Cancer

Example

- Medical research, trend analysis, allocation of public funds, ...

Machanavajjhala et al., TKDD07.

Outline

1. Uncertainty in the Real World

2. Managing Uncertainty

How to deal with uncertainty? (1)

Clean it (then deny it)!

- E.g., data warehouse systems
- Advantages
  - Lots of expertise and tools for cleaning data
  - Can be stored and queried in traditional DBMS
- Disadvantages
  - Loss of information
  - No risk assessment
  - High expense of cleaning
  - New data may “break” the clean database
- Important, but not covered in this lecture!

Customers

Same!

Sys	Cust	Name	City	State
1	C_1	John	SFO	CA
2	C_2	Johnny	SJ	CA
1	C_3	Jak	SFO	CA

CleanedCustomers

Cust	Name	City	State
C_{12}	Johnny	SFO	CA
C_3	Jak	SFO	CA

How to deal with uncertainty? (2)

Manage it!

Approach I: Incomplete databases

- A data integration scenario

Customers

Same!

Sys	Cust	Name	City	State
1	C_1	John	SFO	CA
2	C_2	Johnny	SJ	CA
1	C_3	Jak	SFO	CA

Transactions

Sys	TransID	Cust	Sales
1	T_1	C_1	$15
1	T_2	C_1	$5
2	T_3	C_2	$30
1	T_4	C_3	$30

- Resolving entities via an incomplete database

ResolvedCustomers

Ent	Name	City	State
E_1	John $\|$ Johnny	SFO $\|$ SJ	CA
E_2	Jak	SFO	CA

ResolvedTransactions

TransID	Ent	Sales
T_1	E_1	$15
T_2	E_1	$5
T_3	E_1	$30
T_4	E_2	$30

- Some query results

Sales by city

City	Sum(Sales)	Status
SFO	$30-$80	guaranteed
SJ	$50	non-guaranteed

Sales by state

State	Sum(Sales)	Status
CA	$80	guaranteed

Sismanis et al., ICDE09

Approach II: Probabilistic databases

- Bird watcher's observations

Sightings

	Name	Bird	Species
t_1:	Mary	Bird-1	Finch: 0.8 ∥ Toucan: 0.2
t_2:	Susan	Bird-2	Nightingale: 0.65 ∥ Toucan: 0.35
t_3:	Paul	Bird-3	Humming bird: 0.55 ∥ Toucan: 0.45

- Which species exist in the park?

ObservedSpecies

Species		
Finch: 0.8	?	$(t_1, 1)$
Toucan: 0.714	?	$(t_1, 2) \vee (t_2, 2) \vee (t_3, 2)$
Nightingale: 0.65	?	$(t_2, 1)$
Humming bird: 0.55	?	$(t_3, 1)$

DistinctSpecies

#		
1: 0.0315	?	...
2: 0.2230	?	...
3: 0.7455	?	...

- Observe: Cleaning up data by most likely choice would miss Toucan!

Das Sarma, Stanford Info Blog, 2008.

Approach III: Probabilistic graphical models

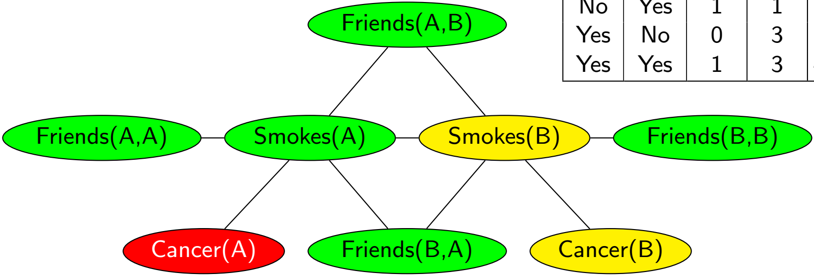
- Anna and Bob are friends. Anna smokes, but does not have cancer. What do we know about Bob?
- Uncertain knowledge

$$1.5 \left\{ \begin{array}{l} \text{Smoking causes cancer} \\ \forall x.\mathsf{Smokes}(x) \implies \mathsf{Cancer}(x) \end{array} \right.$$

$$1.1 \left\{ \begin{array}{l} \text{Friends have similar smoking habits} \\ \forall x.\forall y.\mathsf{Friends}(x, y) \implies (\mathsf{Smokes}(x) \iff \mathsf{Smokes}(y)) \end{array} \right.$$

- Build a graphical model & perform inference

S(B)	C(B)	#R1	#R2	w	Prob.
No	No	1	1	2.6	7.7%
No	Yes	1	1	2.6	7.7%
Yes	No	0	3	3.3	15.4%
Yes	Yes	1	3	4.8	69.2%



How to deal with uncertainty? (2)

Manage it!

- Advantages
  - No or little loss of information
  - Uncertainty might be resolved more accurately at query time
  - Risk assessment is possible
  - Less upfront effort
  - Arrival of new data handled gracefully
- Disadvantages
  - Increased cost of data processing
  - Active research area with lots of open issues (and interesting results)
  - No commercial DBMS systems available!
- This lecture!

Course overview

- Modelling uncertainty
    - Incomplete databases
    - Probabilistic databases
    - Probabilistic graphical models for relational data

- Managing uncertain data
    - Languages (relational algebra, datalog, relational calculus)
    - Provenance
    - Algorithms
    - Complexity
    - Approximation techniques
    - Systems

- Applications
    - Information extraction, sensor networks, business intelligence & predictive analytics, forecasting, scientific data management, privacy preserving data mining, data integration, data deduplication, social network analysis, . . .

Suggested reading

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 1)
  Springer, 2009.

- Daphne Koller, Nir Friedman
  Probabilistic Graphical Models: Principles and Techniques (Chapter 1)
  The MIT Press, 2009

- Dan Suciu, Dan Olteanu, Christopher Ré, Christoph Koch
  Probabilistic Databases (Chapter 1)
  Morgan & Claypool, 2011

- Charu C. Aggarwal, Philip S. Yu
  A Survey of Uncertain Data Algorithms and Applications
  IEEE Transactions of Knowledge and Data Engineering, 21(5), pp. 609–623, May 2009

Scalable Uncertainty Management

02 – Incomplete Databases

Rainer Gemulla

April 27, 2012

Overview

In this lecture

- Refresh relational algebra
- What is an incomplete database?
- How can incomplete information be represented?
- How expressive are these representations?
- How to query incomplete databases?
- How to query their representations?

Not in this lecture

- Complexity
- Efficiency
- Applications

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Notation

- Set of *attributes* $\mathscr{A}$ (countably infinite, totally ordered)
- *Domain* $\mathscr{D}$ of values for the attributes (countably infinite)
- Elements of $\mathscr{D}$ are called *constants*
- Per-attribute domain denoted $\mathrm{dom}(A)$
- Set of *relation names* $\mathscr{R}$, each associated with a finite set of attributes $\alpha(R) \subset \mathscr{A}$ (countably infinite names per finite set of attributes)
- A *schema* is a finite set of attributes (symbols U, W, V)
- A *relation schema* is a relation name (symbols R, S)
- A *database schema* is a nonempty finite set of relation names

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- $\mathscr{A} = \{ A, B, C, D, \ldots \} = ABCD \ldots$
- $\mathscr{D} = \{ a_1, b_1, c_1, a_2, \ldots \}$
- $\mathrm{dom}(A) = \{ a_1, a_2, \ldots \}$
- $\mathscr{R} = \{ R, S, \ldots \}$
- $\alpha(R) = ABC$; write $R[ABC]$

The Named Perspective

- Let $U \subset \mathscr{A}$ be a schema
- *Tuple* t over U is a function $t : U \to \mathscr{D}$ (also called *U-tuple*)
- $\alpha(t)$ denotes the schema of t
- *Value* of attribute $A \in U$ of U-tuple t is denoted $t(A)$ or $t.A$
- *Restriction* of U-tuple t to values in $V \subseteq U$ is denoted $t[V]$
- *Relation instance* $I(R)$ of R is a finite <u>set</u> of tuples over $\alpha(R)$
- *Database instance* $\mathbf{I}$ of database schema $\mathbf{R}$ maps each relation name in $R \in \mathbf{R}$ to a relation instance $\mathbf{I}(R)$

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- t_1 is a tuple over ABC
- $t_1 = \langle A\colon a_1, B\colon b_2, C\colon c_1 \rangle = a_1 b_2 c_1$
- $\alpha(t_1) = ABC$
- $t_1(A) = t_1.A = a_1$
- $t_1[AB] = a_1 b_2$ is a tuple over AB
- $I(R) = \{\, t_1, t_2 \,\} = \{\, a_1 b_2 c_1, a_2 b_1 c_1 \,\}$ is relation instance over ABC

The Unnamed Perspective

- *Tuple* t is an ordered n-tuple $(n \geq 0)$ of constants, i.e., $t \in \mathscr{D}^n$
- *Value* of i-th coordinate denoted $t(i)$
- Natural correspondence to named perspective
  - n-tuples can be viewed as functions with domain $\{1, \ldots, n\}$
  - U-tuples can be viewed as $|U|$-tuples by using total order of attributes

Example

R

t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_1

- $t_1 = \langle a_1, b_2, c_1 \rangle = a_1 b_2 c_1$
- $t_1(1) = a_1$

For now, we will mostly use the named perspective.

Relational algebra (1)

- Relation name R
- Single-tuple, single-attribute *constant relations* (`VALUES` clause)

$$\{\,\langle A\colon a\rangle\,\}$$

  for $A \in \mathscr{A}$, $a \in \mathrm{dom}(A)$
- *Selection* σ (`WHERE` clause)

$$\sigma_{A=a}(I) = \{\, t \in I \mid t.A = a \,\}$$
$$\sigma_{A=B}(I) = \{\, t \in I \mid t.A = t.B \,\}$$

  for $A, B \in \alpha(I)$ and $a \in \mathrm{dom}(A)$.

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_2
t_3:	a_1	b_1	c_1

$\{\,\langle A\colon a\rangle\,\}$

A
a

$\sigma_{A=a_1}(R)$

A	B	C
a_1	b_2	c_1
a_1	b_1	c_1

$\sigma_{A=a_3}(R)$

A	B	C

Relational algebra (2)

- *Projection* π (`SELECT DISTINCT` clause)

  $$\pi_U(I) = \{\, t[U] \mid t \in I \,\}$$

  for $U \subseteq \alpha(R)$

- *Natural Join* $\bowtie$ (`FROM clause`)

  $$I \bowtie J = \{\, t \text{ over } U \cup V \mid t[U] \in I \wedge t[V] \in J \,\},$$

  where $U = \alpha(I)$, $V = \alpha(J)$

Example

R

	A	B	C
t_1:	a_1	b_2	c_1
t_2:	a_2	b_1	c_2
t_3:	a_1	b_1	c_1

S

	A	D
t_4:	a_1	d_1
t_5:	a_3	d_2
t_6:	a_1	d_3

$\pi_{AC}(R)$

A	C
a_1	c_1
a_2	c_2

$R \bowtie S$

A	B	C	D
a_1	b_2	c_1	d_1
a_1	b_2	c_1	d_3
a_1	b_1	c_1	d_1
a_1	b_1	c_1	d_3

Relational algebra (3)

- *Renaming* of attributes ρ (`AS` clause)

$$\rho_{A_1 \ldots A_n \to B_1 \ldots B_n}(I) = \{\, t \text{ over } V \mid (\exists u \in I)(\forall i \in [1,n])\, u.A_i = t.B_i \,\},$$

where $\alpha(I) = \{\, A_1, \ldots, A_n \,\}$, $V = \{\, B_1, \ldots, B_n \,\}$

- Short notation: only list attributes being renamed

Example

R

	A	B
t_1:	a_1	b_2
t_2:	a_2	b_1
t_3:	a_1	b_1

$\rho_{AB \to CD}(R)$

C	D
a_1	b_2
a_2	b_1
a_1	b_1

$\rho_{AB \to BA}(R)$

B	A
a_1	b_2
a_2	b_1
a_1	b_1

$R \bowtie \rho_{B \to C}(R)$

A	B	C
a_1	b_2	b_2
a_1	b_2	b_1
a_2	b_1	b_1
a_1	b_1	b_1
a_1	b_1	b_2

Relational algebra (4)

- *Union* $\cup$ (`UNION` clause)

  $$I \cup J = \{\, t \mid t \in I \vee t \in J \,\}$$

  for $\alpha(I) = \alpha(J)$
- *Difference* $-$ (`EXCEPT` clause)

  $$I - J = \{\, t \mid t \in I \wedge t \notin J \,\}$$

  for $\alpha(I) = \alpha(J)$

Example

R

	A	B
t_1:	a_1	b_2
t_2:	a_2	b_1
t_3:	a_1	b_1

S

	A	B
t_4:	a_1	b_2
t_5:	a_2	b_1
t_6:	a_3	b_2

$R \cup S$

A	B
a_1	b_2
a_2	b_1
a_1	b_1
a_3	b_2

$R - S$

A	B
a_1	b_1

$\mathscr{L}$-expression

Definition

Let $\mathscr{L} \subseteq$ SPJRUD be an algebra. An *$\mathscr{L}$-expression* is any well-formed relational algebra expression composed of only relation names, constant relations, and the operations in $\mathscr{L}$. Algebra $\mathscr{L}$ is *positive* if it does not contain the difference operator.

Example

- $\pi_A(\pi_{AB}(R))$ is a P-expression but not an S-expression
- $\sigma_{A=a}(R)$ is both an S-expression and a PS-expression, but not a P-expression
- R is an $\emptyset$-expression
- All of the above expressions are positive, but $R - S$ is not

Generalized Selection

- Relational algebra
  - $\sigma_{A=a}(R)$ for $A \in \alpha(R)$ and $a \in \text{dom}(A)$
  - $\sigma_{A=B}(R)$ for $A, B \in \alpha(R)$
  - $A = a$ and $A = B$ are called *predicates*
- Generalized selection operators extend the class of predicates
- Positive conjunction

$$\sigma_{P_1 \wedge P_2}(R) = \sigma_{P_1}(\sigma_{P_2}(R))$$

- Positive disjunction (S^+)

$$\sigma_{P_1 \vee P_2}(R) = \sigma_{P_1}(R) \cup \sigma_{P_2}(R)$$

- Negation (S^-, not positive)

$$\sigma_{\neg P}(R) = R - \sigma_P(R)$$

- Note: Union and difference can simulate generalized selection but not vice versa! → S^+ and S^- variants of S

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Examples of incomplete information

Certain data

Uncertain data

Paul owns a car.

Name	Object
Paul	Car

Paul may own a car.

Name	Object
Paul	Car

or

Name	Object

Tuple-level uncertainty

Bob works for Yahoo.

Name	Company
Bob	Yahoo

Bob works for either Yahoo or Microsoft.

Name	Company
Bob	Yahoo

or

Name	Company
Bob	Microsoft

Attribute-level uncertainty

Mary sighted a finch.
Paul sighted a finch.

Name	Bird
Mary	Finch
Paul	Finch

Mary sighted a finch or a sparrow.
Paul sighted what Mary sighted.

Name	Bird
Mary	Finch
Paul	Finch

or

Name	Bird
Mary	Sparrow
Paul	Sparrow

Correlations

Paul's favorite number is 17.

Name	Num
Paul	17

Paul has a favorite number,
but it is unknown.

Name	Num
Paul	1

or

Name	Num
Paul	2

or . . .

Infinity

We need a precise way to *model* and *represent* incomplete information.

Examples of incomplete databases

Certain data | **Uncertain data**

Paul owns a car.

Name	Object
Paul	Car

Paul may own a car.

{

Name	Object
Paul	Car

,

Name	Object

}

Tuple-level uncertainty

Bob works for Yahoo.

Name	Company
Bob	Microsoft

Bob works for either Yahoo or Microsoft.

{

Name	Company
Bob	Yahoo

,

Name	Company
Bob	Microsoft

}

Attribute-level uncertainty

Mary sighted a finch.
Paul sighted a finch.

Name	Bird
Mary	Finch
Paul	Finch

Mary sighted a finch or a sparrow.
Paul sighted what Mary sighted.

{

Name	Bird
Mary	Finch
Paul	Finch

,

Name	Bird
Mary	Sparrow
Paul	Sparrow

}

Correlations

Paul's favorite number is 17.

Name	Num
Paul	17

Paul has a favorite number, but it is unknown.

{

Name	Num
Paul	1

,

Name	Num
Paul	2

, . . . }

Infinity

An incomplete database is a set of "possible worlds" (i.e., DB instances).

Incomplete database

$\mathcal{N}_U = \{\, I \mid I$ is a (finite) relation instance over schema $U \,\}$

Definition

- An *incomplete relation* (*i-relation*) $\mathcal{I}$ over U is a set of possible relation instances over U, i.e., $\mathcal{I} \subseteq \mathcal{N}_U$.
- An *incomplete database* (*i-database*) of a database schema $\mathbf{R}$ maps each relation name $R \in \mathbf{R}$ to an incomplete relation over $\alpha(R)$.

- "Incomplete" refers to incomplete information
- Focus on one relation → use i-relation and i-database synonymously
- Usual relation instances: $\mathcal{I} = \{\, I \,\}$
- *No-information* or *zero-information database* over U: $\mathcal{I} = \mathcal{N}_U$
- Incomplete databases can be *infinite* even though every relation instance is finite; e.g., $\{\, \boxed{a_1}, \boxed{a_2}, \boxed{a_3}, \ldots \,\}$
- $\mathcal{N}_U$ is (countably) infinite
- Set of all incomplete relations is uncountable

Representation system

- Incomplete databases are in general infinite
- Even if finite, they can be very large

→ Need compact representation!

Definition

A *representation system* consists of a set (a “language”) $\mathscr{T}$ whose elements we call *tables*, and a function Mod that associates to each table $T \in \mathscr{T}$ an incomplete database $\mathrm{Mod}(T)$.

- Again, we'll assume a single relation (reformulation for multiple relations possible)
- $\mathrm{Mod}(T)$ can be thought of as the set of database instances consistent with T (called the *possible worlds*)
- T can be viewed as logical assertion; $\mathrm{Mod}(T)$ are *models* of T

Codd tables

- Missing values are indicated by a single, untyped *null value* @
- Each occurrence of @ stands for a value of the attribute's domain
- Different occurrences may or may not refer to the same value

Example

SUPPLIER	LOCATION	PRODUCT	QUANTITY
Smith	London	Nails	@
Brown	@	Bolts	@
Jones	@	Nuts	40,000

Definition

An *@-tuple* on U is an extended tuple in which each attribute $A \in U$ takes values in $\mathrm{dom}(A) \cup \{ \texttt{@} \}$. A *Codd table* is a finite set of @-tuples.

Models of Codd tables (1)

Definition

Under the *closed world interpretation*, a Codd table represents the set of relations obtained by replacing @-symbols by valid values.

Example

Suppose $\text{dom}(A) = \{ a_1, a_2 \}$ and $\text{dom}(B) = \{ b_1, b_2 \}$.

$$\text{Mod}\left(\begin{array}{|c|c|}\hline a_1 & @ \\ @ & b_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array} \right\}$$

Let $R^* \in$ RHS of the example:

- There is no certain tuple, i.e., $\nexists t \forall R^* \; t \in R^*$
- The first column contains a_1, the second b_2
- R^* has at least one and at most 2 tuples
- $a_2 b_1$ is not in R^*
- ...

Negative information *can* be represented.

Models of Codd tables (2)

Definition

Under the *open world interpretation*, a Codd table represents the set of relations obtained by replacing @-symbols by valid values and adding arbitrarily many additional tuples.

Equivalently, this means $S \in \text{MOD}(T) \iff (\exists R)\, R \in \text{Mod}(T) \wedge S \supseteq R$.

Example

$$\text{MOD}\left(\overset{T}{\begin{array}{|c|c|}\hline a_1 & @ \\ @ & b_2 \\ \hline\end{array}}\right) = \left\{ \begin{array}{l} \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \\ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ a_2 & b_1 \\ a_2 & b_2 \\ \hline\end{array} \end{array} \right\}$$

Models of Codd tables (3)

Example

$$\mathrm{MOD}\Bigg(\overset{T}{\begin{array}{|c|c|}\hline a_1 & \text{@} \\ \text{@} & b_2 \\ \hline\end{array}}\Bigg) = \left\{ \begin{array}{c} \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_2} \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_1} & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_2} & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_2} \\ \textcolor{red}{a_2} & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_2} \\ a_2 & b_1 \\ \hline\end{array}, \\ \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_1} & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_1} & b_2 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_2} & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_2} \\ \textcolor{red}{a_2} & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & \textcolor{red}{b_1} \\ \textcolor{red}{a_1} & b_2 \\ a_2 & b_1 \\ a_2 & b_2 \\ \hline\end{array} \end{array} \right\}$$

Let $R^* \in$ RHS of the example:

- There is no certain tuple, i.e., $\nexists t \forall R^* \; t \in R^*$
- The first column contains a_1, the second b_2
- R^* has at least one tuple
- Every tuple is possible, i.e., $\forall t \exists R^* \; t \in R^*$
- ...

Negative information *cannot* be represented.

v-Tables

- Missing values are indicated by *marked null values* or variables
- $V(A)$ = set of *variables* for attribute A (countably infinite)
- $V(A) \cap V(B) = \emptyset$ if $\mathrm{dom}(A) \neq \mathrm{dom}(B)$; otherwise $V(A) = V(B)$

Example

Course	Teacher	Weekday
Databases	x	Monday
Programming	y	Tuesday
Databases	x	Thursday
FORTRAN	Smith	z

Definition

A *v-tuple* on U is an extended tuple in which each attribute $A \in U$ takes values in $dom(A) \cup V(A)$. A *v-table* is a finite set of *v*-tuples.

Models of v-tables

Example

Suppose dom$(A) = \{ a_1, a_2 \}$, dom$(B) = \{ b_1, b_2 \}$, dom$(C) = \{ c_1, c_2 \}$.

$$\text{Mod}\left(\begin{array}{|c|c|}\hline a_1 & x \\ y & b_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array} \right\}$$

$$\text{Mod}\left(\begin{array}{|c|c|}\hline c_1 & z \\ z & c_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline c_1 & c_1 \\ c_1 & c_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_1 & c_2 \\ c_2 & c_2 \\ \hline\end{array} \right\}$$

$$\text{Mod}\left(\begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline c_1 & c_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_1 & c_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_2 & c_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline c_2 & c_2 \\ \hline\end{array} \right\}$$

- Var$(T) = \{ x \mid$ variable x occurs in T $\}$
- *Valuation* v : Var$(T) \to \mathscr{D}$ assigns (valid) values to each variable
- $v(T)$ is the relation obtained by replacing all variables by their values
- Mod$(T) = \{ v(T) \mid v$ is a valuation for Var$(T) \}$

Codd tables $\equiv$ v-tables in which each variable occurs at most once.

v-Tables and view updates

v-tables appear naturally when updating relational views.

Example

SL

Supplier	Location
Smith	London
x	New York
y	Los Angeles

SP

Supplier	Product
Smith	Nails
x	Bolts
y	Nuts

$\pi_{\text{Location,Product}}(\text{SL} \bowtie \text{SP})$

Location	Product
London	Nails
New York	Bolts
Los Angeles	Nuts

c-Tables

- *c-tables* are v-tables with an additional *condition* column *con*, indicating a "tuple existence condition" → *conditional table*
- *Conditions* taken from a set $\mathscr{C}$ composed of
  - false, true
  - $x = a$ for $x \in V(A)$ and $a \in \mathrm{dom}(A)$ for some $A \in \mathscr{A}$
  - $x = y$ for $x, y \in V(A)$ for some $A \in \mathscr{A}$
  - negation $\neg$, disjunction $\vee$, conjunction $\wedge$
- *Positive conditions* do not contain negations (set $\mathscr{C}^+$)

Example

Supplier	Location	Product	con
x	London	Nails	$x =$ Smith
Brown	New York	Nails	$x \neq$ Smith

Definition

A *c-tuple* t on U is an extended tuple over $U \cup \{con\}$ such that $t[U]$ is a v-tuple and $t(con) \in \mathscr{C}$. A *c-table* is a finite set of c-tuples.

Models of c-Tables

Example

Suppose $dom(x) = dom(y) = \{1, 2\}$.

$$\text{Mod}\left(\begin{array}{|c|c|c|}\hline A & B & con \\ \hline a_1 & b_1 & x = 1 \\ a_2 & b_1 & x \neq 1 \\ a_3 & b_2 & y = 1 \wedge x \neq 1 \\ a_4 & b_2 & y \neq 1 \vee x = 1 \\ \hline \end{array}\right) = \left\{ \overset{\text{x1y1}}{\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array}}, \overset{\text{x1y2}}{\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array}}, \overset{\text{x2y1}}{\begin{array}{|c|c|}\hline a_2 & b_1 \\ a_3 & b_2 \\ \hline \end{array}}, \overset{\text{x2y2}}{\begin{array}{|c|c|}\hline a_2 & b_1 \\ a_4 & b_2 \\ \hline \end{array}} \right\}$$

$$= \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_4 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_3 & b_2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ a_4 & b_2 \\ \hline \end{array} \right\}$$

- Valuation check conditions: $v(T) = \{\, v(t[U]) \mid v(t(con)) = \text{true} \,\}$
- $\text{Mod}(T) = \{\, v(T) \mid v \text{ is a valuation for } \text{Var}(T) \,\}$

v-tables are equivalent to *c*-tables in which each condition equals true.

Finite representation systems

Definition

In a *finite-domain* Codd-table, v-table, or c-table T, each variable $x \in \mathrm{Var}(T)$ is associated with a finite domain $\mathrm{dom}(x)$.

- Important in practice
- Sometimes easier to study
- Basis for most probabilistic databases
- Incomplete database is finite
  (but attribute domain and no. variables still countably infinite)

Other finite representation systems

All of these models can be seen as special cases of finite-domain c-tables.

Example

In *?-tables*, tuples are marked with ? if they may not exist.

$$\mathrm{Mod}\left(\begin{array}{|c|c|c}\cline{1-2} a_1 & b_1 & \\ a_1 & b_2 & ? \\ \cline{1-2}\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array} \right\}$$

In *or-set tables*, $t.A$ takes values in a finite subset of $\mathrm{dom}(A)$.

$$\mathrm{Mod}\left(\begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \| b_2 \\ a_2 & b_1 \| b_2 \\ \hline\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_2 \\ a_2 & b_2 \\ \hline\end{array} \right\}$$

Equivalent to finite-domain Codd tables.

In a *?-or-set table*, both are combined.

$$\mathrm{Mod}\left(\begin{array}{|c|c|c}\cline{1-2} a_1 & b_1 & \\ a_2 & b_1 \| b_2 & ? \\ \cline{1-2}\end{array}\right) = \left\{ \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array} \right\}$$

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. **Strong representation systems**
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Possible answer set semantics

Definition

The *possible answer set* to a query q on an incomplete database $\mathcal{I}$ is the incomplete database $q(\mathcal{I}) = \{\, q(I) \mid I \in \mathcal{I} \,\}$.

Example

Let $q(R) = \sigma_{A=a_1}(R)$.

$$q\left(\left\{\ \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ a_2 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_2 & b_1 \\ \hline\end{array}\right\}\right) = \left\{\begin{array}{|c|c|}\hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array}, \begin{array}{|c|c|}\hline a_1 & b_1 \\ \hline\end{array}, \begin{array}{|c|c|}\hline & \\ \hline\end{array}\right\}$$

Can we compute the representation of the possible answer set to a query from the representation of an incomplete database?

Strong representation systems

Definition

- A representation system is *closed* under a query language if for any query q and any table T there is a table $\bar{q}(T)$ that represents $q(\mathrm{Mod}(T))$.
- If $\bar{q}(T)$ can always be computed from q and T, the representation system is called *strong* under the query language.

$$\begin{tikzcd}
T \arrow[r, "\mathrm{Mod}"] \arrow[d, "\bar{q}"'] & \mathcal{I} \arrow[d, "q"] \\
\bar{q}(T) \arrow[r, "\mathrm{Mod}"] & q(\mathcal{I})
\end{tikzcd}$$

Intuitively, this means that the query language is "fully supported" by the representation system: query answers can be both computed and represented.

Normalized c-tables

Definition

A c-table T on U is *normalized* if $t[U] \neq t'[U]$ for all pairs of distinct c-tuples $t, t' \in T$.

Example

Not normalized

a_1	b_1	$x = 1$
a_1	b_1	$x = 2$
a_2	b_2	true

Normalized

a_1	b_1	$x = 1 \vee x = 2$
a_2	b_2	true

To normalize a c-table, repeatedly apply rule 3 (next slide).

We'll assume normalized c-tables throughout.

Mod-equivalence

Definition

Two tables T and T' are Mod-*equivalent* (or just *equivalent*) if $\text{Mod}(T) = \text{Mod}(T')$. We write $T \equiv_{\text{Mod}} T'$.

Mod-equivalent transformations on c-table T on U:

1. Replace a condition by an equivalent condition; e.g., $(x = 1 \wedge y = 1) \vee (x \neq 1 \wedge y = 1)$ by $y = 1$
2. Remove tuples in which condition is unsatisfiable; e.g., $x = 1 \wedge x = 2$
3. Merge tuples $t_1, \ldots, t_k \in T$ with $t_1[U] = \cdots = t_k[U]$ into a new tuple t' s.t. $t'[U] = t_1[U]$ and $t'.con = t_1.con \vee \cdots \vee t_k.con$.

Mod-equivalent transformations can be used to simplify c-tables.

c-Tables are strong

Theorem

c-tables, finite-domain c-tables, and Boolean c-tables are strong under $\mathcal{RA}$.

Proof.

Given a $\mathcal{RA}$ query q, construct $\bar{q}$ by replacing in q the operators π, σ, $\bowtie$, $\cup$, and $-$ by the respective operators $\bar{\pi}, \bar{\sigma}, \bar{\bowtie}, \bar{\cup}, \bar{-}$ of the *c-table algebra*. Then $v(\bar{q}(T)) = q(v(T))$ for all valuations v for $\mathrm{Var}(T)$. $\square$

- We assume and produce normalized c-tables
- Boolean c-table: all variables are boolean
- $T(t)$ denotes $t.con$ if $t \in T$; false otherwise
- $T[]$ drops condition column of normalized c-table
- Relational algebra operations on $T[]$ treat variables as normal values

c-Projection

Definition

$$\bar{\pi}_U(T)[] = \pi_U(T[])$$

$$\bar{\pi}_U(T)(t) = \bigvee_{t' \in T \text{ s.t. } t'[U]=t} T(t')$$

Example

Sightings

Name	*Species*	*con*
Anna	Guan	$x = 1$
Anna	Humming bird	$x = 2$
Bob	y	$x = 3$
z	Guan	$x = 4$

$\bar{\pi}_{Name}$(Sightings)

Name	*con*
Anna	$x = 1 \vee x = 2$
Bob	$x = 3$
z	$x = 4$

c-Selection

Definition

$$\begin{aligned}\bar{\sigma}_P(T)[] &= T[] \\ \bar{\sigma}_P(T)(t) &= T(t) \wedge P(t),\end{aligned}$$

where $P(t)$ replaces in P each occurence of an attribute A by $t.A$ and evaluates subexpressions of form $a = b$ (to false) and $a = a$ (to true).

Example

Sightings

N	S	con
A	G	$x = 1$
A	H	$x = 2$
B	y	$x = 3$
z	G	$x = 4$

$\bar{\sigma}_{Species=\mathrm{Guan}}$(Sightings)

N	S	con
A	G	$x = 1 \wedge \mathrm{true}$
A	H	$x = 2 \wedge \mathrm{false}$
B	y	$x = 3 \wedge y = \mathrm{G}$
z	G	$x = 4 \wedge \mathrm{true}$

$\bar{\sigma}_{S=\mathrm{G}}$(Sightings) (simpl.)

N	S	con
A	G	$x = 1$
B	y	$x = 3 \wedge y = \mathrm{G}$
z	G	$x = 4$

c-Union

Definition

$$(T_1 \bar{\cup} T_2)[] = T_1[] \cup T_2[]$$
$$(T_1 \bar{\cup} T_2)(t) = T_1(t) \vee T_2(t)$$

Example

Sightings

N	con
A	$x = 1$
B	$x = 2$
C	$x = 3$

VIPs

N	con
B	$y = 1$
C	$y = 2$
z	$y = 3$

Sightings $\bar{\cup}$ VIPs

N	con
A	$x = 1 \vee \text{false}$
B	$x = 2 \vee y = 1$
C	$x = 3 \vee y = 2$
z	$\text{false} \vee y = 3$

S$\bar{\cup}$V (simplified)

N	con
A	$x = 1$
B	$x = 2 \vee y = 1$
C	$x = 3 \vee y = 2$
z	$y = 3$

c-Join (1)

Definition

Set $U_1 = \alpha(T_1)$, $U_2 = \alpha(T_2)$, and denote by $V = U_1 \cap U_2 = A_1 \ldots A_k$ the join attributes. Let $V' = A'_1 \ldots A'_k$ be a fresh set of attributes (of matching domains). Set $T'_2 = \rho_{V \to V'}(T_2)$ and $U'_2 = \alpha(T'_2)$.

$$(T_1 \bar{\bowtie}_{V \to V'} T_2)[] = T_1[] \bowtie T'_2[]$$

$$(T_1 \bar{\bowtie}_{V \to V'} T_2)(t) = T_1(t[U_1]) \wedge T'_2(t[U'_2]) \bigwedge_{A \in V} t.A = t.A'$$

$$T_1 \bar{\bowtie} T_2 = \bar{\pi}_{U_1 \cup U_2}(T_1 \bar{\bowtie}_{V \to V'} T'_2).$$

c-Join (2)

Example

Sightings

N	S	con
A	G	$x = 1$
A	H	$x = 2$
z_1	K	$x = 3$
z_2	L	$x = 4$

VIPs

N	con
A	$y = 1$
B	$y = 2$
z_1	$y = 3$

VIPs′

N'	con
A	$y = 1$
B	$y = 2$
z_1	$y = 3$

Sightings$\bar{\bowtie}_{N \to N'}$VIPs

N	S	N'	con
A	G	A	$x = 1 \wedge y = 1 \wedge \text{true}$
A	H	A	$x = 2 \wedge y = 1 \wedge \text{true}$
z_1	K	A	$x = 3 \wedge y = 1 \wedge z_1 = \text{A}$
z_2	L	A	$x = 4 \wedge y = 1 \wedge z_2 = \text{A}$
A	G	B	$x = 1 \wedge y = 2 \wedge \text{false}$
A	H	B	$x = 2 \wedge y = 2 \wedge \text{false}$
z_1	K	B	$x = 3 \wedge y = 2 \wedge z_1 = \text{B}$
z_2	L	B	$x = 4 \wedge y = 2 \wedge z_2 = \text{B}$
A	G	z_1	$x = 1 \wedge y = 3 \wedge z_1 = \text{A}$
A	H	z_1	$x = 2 \wedge y = 3 \wedge z_1 = \text{A}$
z_1	K	z_1	$x = 3 \wedge y = 3 \wedge z_1 = z_1$
z_2	L	z_1	$x = 4 \wedge y = 3 \wedge z_2 = z_1$

c-Join (3)

Example (continued)

Sightings

N	S	con
A	G	$x1$
A	H	$x2$
z_1	K	$x3$
z_2	L	$x4$

VIPs

N	con
A	$y1$
B	$y2$
z_1	$y3$

VIPs′

N'	con
A	$y1$
B	$y2$
z_1	$y3$

Sightings$\bar{\bowtie}_{N \rightarrow N'}$VIPs (simplified)

N	S	N'	con
A	G	A	$x1y1$
A	H	A	$x2y1$
z_1	K	A	$x3y1 \wedge z_1 = \text{A}$
z_2	L	A	$x4y1 \wedge z_2 = \text{A}$
z_1	K	B	$x3y2 \wedge z_1 = \text{B}$
z_2	L	B	$x4y2 \wedge z_2 = \text{B}$
A	G	z_1	$x1y3 \wedge z_1 = \text{A}$
A	H	z_1	$x2y3 \wedge z_1 = \text{A}$
z_1	K	z_1	$x3y3$
z_2	L	z_1	$x4y3 \wedge z_2 = z_1$

Sightings$\bar{\bowtie}$VIPs (simplified)

N	S	con
A	G	$x1y1 \vee (x1y3 \wedge z_1 = \text{A})$
A	H	$x2y1 \vee (x2y3 \wedge z_1 = \text{A})$
z_1	K	$(x3y1 \wedge z_1 = \text{A}) \vee (x3y2 \wedge z_1 = \text{B}) \vee x3y3$
z_2	L	$(x4y1 \wedge z_2 = \text{A}) \vee (x4y2 \wedge z_2 = \text{B}) \vee (x4y3 \wedge z_2 = z_1)$

c-Difference

Definition (c-Table difference)

$$(T_1 \bar{-} \text{VIPs})[] = T_1[]$$

$$(T_1 \bar{-} \text{VIPs})(t) = T_1(t) \bigwedge_{t' \in \text{VIPs}} \neg(t = t' \wedge \text{VIPs}(t'))$$

Example

Sightings

A	con
A	$x1$
B	$x2$
C	$x3$

VIPs

A	con
B	$y1$
C	$y2$
z	$y3$

Sightings $\bar{-}$ VIPs (simplified)

A	con
A	$x1 \wedge \neg(z = \text{A} \wedge y3)$
B	$x2 \wedge \neg y1 \wedge \neg(z = \text{B} \wedge y3)$
C	$x3 \wedge \neg y2 \wedge \neg(z = \text{C} \wedge y3)$

Sightings $\bar{-}$ VIPs

A	con
A	$x1 \wedge \neg(\text{false} \wedge y1) \wedge \neg(\text{false} \wedge y2) \wedge \neg(z = \text{A} \wedge y3)$
B	$x2 \wedge \neg(\text{true} \wedge y1) \wedge \neg(\text{false} \wedge y2) \wedge \neg(z = \text{B} \wedge y3)$
C	$x3 \wedge \neg(\text{false} \wedge y1) \wedge \neg(\text{true} \wedge y2) \wedge \neg(z = \text{C} \wedge y3)$

Many representation systems are not closed

Theorem

Codd tables, v-tables, finite-domain Codd tables, finite-domain v-tables, ?-tables, or-set tables, and ?-or-set tables are not closed under $\mathcal{RA}$.

Proof.

By counterexample. Consider:

- Codd tables / v-tables (standard and finite-domain), or-set tables, ?-or-set tables:

$$\sigma_{A \neq B}\left(\begin{array}{|c|c|}\hline A & B \\ \hline x & y \\ \hline\end{array}\right) \qquad \text{where } \mathrm{dom}(x) = \mathrm{dom}(y) \text{ and } |\mathrm{dom}(x)| > 1.$$

- ?-tables:

$$\begin{array}{|c|c|}\hline A & B \\ \hline a_1 & b_1 \\ a_1 & b_2 \\ \hline\end{array} \bowtie \begin{array}{|c|}\hline A \\ \hline a_1 \\ \hline\end{array}?$$

We will see: these systems are still very useful!

$\square$

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. **Completeness**
5. Weak Representation Systems
6. Completion
7. Summary

Expressive power

Key question: How expressive is a given representation system?

Theorem

Neither Codd tables, v-tables, nor c-tables can represent all possible incomplete databases.

Proof.

Set of incomplete databases is uncountable, set of tables is countable. □

- E.g., zero-information database $\mathscr{N}_U$ cannot be represented with closed world assumption
- Need to study weaker forms of expressiveness
  1. $\mathcal{RA}$-completeness
  2. Finite completeness

$\mathcal{RA}$-definability (1)

- $\mathcal{Z}_V = \{\, \{\, t \,\} \mid \alpha(t) = V \,\}$
- $\mathcal{Z}_V$ is the minimal-information database for instances of cardinality 1

Example

Let $V = B_1 B_2$, where $\mathrm{dom}(B_1) = \mathrm{dom}(B_2) = \{\, 1, 2, \dots \,\}$.

$$\mathcal{Z}_V = \left\{ \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 2 \\ \hline \end{array}, \dots \right\}$$

Definition

An incomplete database $\mathcal{I}$ over U is *$\mathcal{RA}$-definable* if there exists a relational algebra query q such that $\mathcal{I} = q(\mathcal{Z}_V)$ for some V.

$\mathcal{RA}$-definability (2)

Theorem

If $\mathcal{I}$ is representable by some c-table T, then $\mathcal{I}$ is $\mathcal{RA}$-definable.

Proof.

Let $\alpha(T) = U = A_1 \dots A_n$. Let $x_1, \dots, x_k$ denote the variables in T and let $V = B_1 \dots B_k$ be a set of attributes such that $\mathrm{dom}(B_j) = \mathrm{dom}(x_j)$. Consider the query

$$q(Z) = \bigcup_{t \in T} \pi_U \left(\sigma_{\rho_{x_1 \dots x_k \to B_1 \dots B_k}(t.con)} \left[A_1(t) \bowtie \cdots \bowtie A_n(t) \bowtie Z \right] \right),$$

where

$$A_i(t) = \begin{cases} \{\, \langle A_i : a \rangle \,\} & \text{if } t.A_i = a \\ \rho_{B_j \to A_i}(\pi_{B_j}(Z)) & \text{if } t.A_i = x_j \end{cases}$$

We have $q(\mathcal{Z}_V) = \mathcal{I}$. □

$\mathcal{RA}$-definability (3)

Example

T

A_1	A_2	*con*
a_1	b_1	$x = 1$
a_2	b_1	$x \neq 1$
a_3	b_2	$y = 1 \wedge x \neq 1$
a_4	b_2	$y \neq 1 \vee x = 1$

$$\mathcal{Z}_V = \left\{ \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 1 & 2 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline B_1 & B_2 \\ \hline 2 & 2 \\ \hline \end{array}, \ldots \right\}$$

$$\begin{aligned}
q(Z) := \quad & \pi_{A_1 A_2}\left(\sigma_{B_1=1} \left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_1 & b_1 \\ \hline \end{array} \bowtie Z \right]\right) \\
\cup \quad & \pi_{A_1 A_2}\left(\sigma_{B_1\neq 1} \left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_2 & b_1 \\ \hline \end{array} \bowtie Z \right]\right) \\
\cup \quad & \pi_{A_1 A_2}\left(\sigma_{B_2=1\wedge B_1\neq 1} \left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_3 & b_2 \\ \hline \end{array} \bowtie Z \right]\right) \\
\cup \quad & \pi_{A_1 A_2}\left(\sigma_{B_2\neq 1\vee B_1=1} \left[\begin{array}{|c|c|}\hline A_1 & A_2 \\ \hline a_4 & b_2 \\ \hline \end{array} \bowtie Z \right]\right)
\end{aligned}$$

$\mathcal{RA}$-completeness

Definition

A representation system is *$\mathcal{RA}$-complete* if it can represent any $\mathcal{RA}$-definable incomplete database.

Theorem

c-tables are $\mathcal{RA}$-complete.

Proof.

Let $\mathcal{I}$ be $\mathcal{RA}$-definable using query $q(\mathcal{Z}_V)$. Let T be a c-table representing $\mathcal{Z}_V$, i.e., set

$T =$

B_1	B_2	$\dots$	B_k	con
x_1	x_2	$\dots$	x_k	true

Since c-tables are closed under $\mathcal{RA}$, $\bar{q}(T)$ produces a c-table that represents $\mathcal{I}$. □

Finite completeness (1)

Definition

A representation system is *finitely complete* if it can represent any finite incomplete database.

Theorem

Boolean c-tables (and hence finite-domain and standard c-tables) are finitely complete.

Corollary

Every $\mathcal{RA}$-complete representation system is finitely complete.

Finite completeness (2)

Proof.

Let $\mathcal{I} = \{ I^0, \ldots, I^{n-1} \}$ be a finite incomplete database and assume wlog that $n = 2^m$ for some positive integer m. Let $\mathbf{x} = (x_{m-1}, \ldots, x_0)$ be a vector of boolean variables. There are 2^m possible values of $\mathbf{x}$; assign a unique one to each I^w, $w \in \{ 0, \ldots, n-1 \}$. Let $c_w(\mathbf{x})$ be a Boolean formula that checks whether $\mathbf{x}$ takes the value assigned to I^w. Then set

$$T[] = \bigcup_{w} I^w$$

$$T(t) = \bigvee_{w \text{ s.t. } t \in I^w} c_w(\mathbf{x}).$$

We have $\mathrm{Mod}(T) = \mathcal{I}$. □

Finite completeness (3)

Example

$\mathcal{I} = \Bigg\{$

I^0

A	B
a_1	b_1

,

I^1

A	B
a_2	b_2
a_3	b_3

,

I^2

A	B
a_1	b_1
a_2	b_2

,

I^3

A	B

$\Bigg\}$

Instance	$\mathbf{x} = (x_1, x_0)$	$c_w(\mathbf{x})$
I^0	(F, F)	$\neg x_1 \wedge \neg x_0$
I^1	(F, T)	$\neg x_1 \wedge x_0$
I^2	(T, F)	$x_1 \wedge \neg x_0$
I^3	(T, T)	$x_1 \wedge x_0$

$T =$

A	B	con
a_1	b_1	$(\neg x_1 \wedge \neg x_0) \vee (x_1 \wedge \neg x_0)$
a_2	b_2	$(\neg x_1 \wedge x_0) \vee (x_1 \wedge \neg x_0)$
a_3	b_3	$(\neg x_1 \wedge x_0)$

Incompleteness results

Theorem

Codd tables, v-tables, finite-domain Codd tables, finite-domain v-tables, ?-tables, or-set tables, and ?-or-set tables are not finitely complete (and thus not $\mathcal{RA}$-complete).

Proof.

By counterexample. Consider the finite incomplete database

$$\mathcal{I} = \left\{ \begin{array}{|c|c|} \hline A_1 & A_2 \\ \hline a_1 & a_1 \\ \hline \end{array}, \begin{array}{|c|c|} \hline A_1 & A_2 \\ \hline a_2 & a_3 \\ \hline \end{array} \right\}.$$

□

Due to their simplicity (and completion properties), these representation systems are very useful in practice. This motivates the study of weak representation systems.

A note on compactness

In practice, compactness of representation is important!

Example

Let $x_1, \ldots, x_k$ be variables with domain $\{1, 2, \ldots, n\}$. Consider the finite-domain v-table

A_1	A_2	$\ldots$	A_k
x_1	x_2	$\ldots$	x_k

.

The corresponding Boolean c-table has n^k rows!

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Certain answer tuple semantics (1)

Definition

Let $\mathcal{I}$ be an incomplete database and q a relational algebra query. The *q-information* $\mathcal{I}^q$ is given by the set of certain tuples in $q(\mathcal{I})$, i.e., $\mathcal{I}^q = \cap_{I \in q(\mathcal{I})} I$. Note that $\mathcal{I}^q$ is a certain database; it constitutes the query result under the *certain answer tuple semantics*.

Example

- $\mathcal{I} = \left\{ \overset{I^1}{\begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Guan} \\ \hline \end{array}}, \overset{I^2}{\begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Hb} \\ \hline \end{array}} \right\}$
- $\mathcal{I}^R = I^1 \cap I^2 = \begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \hline \end{array}$
- $\mathcal{I}^{\pi_S(R)} = \pi_S(I^1) \cap \pi_S(I^2) = \begin{array}{|c|}\hline \text{Guan} \\ \hline \end{array}$
- $\mathcal{I}^{\pi_N(R)} = \pi_N(I^1) \cap \pi_N(I^2) = \begin{array}{|c|}\hline \text{Anna} \\ \text{Bob} \\ \hline \end{array}$

Different relational queries expose more or less information about certain tuples!

Certain answer tuple semantics (2)

Definition

Let T be a table and q a relational algebra query. The *q-information* T^q is given by the set of certain tuples in $q(\mathrm{Mod}(\mathcal{I}))$, i.e., $T^q = \cap_{I \in q(\mathrm{Mod}\,(\mathcal{I}))} I$. Note that T^q is a certain database.

Example

Suppose $\mathrm{dom}(x) = \{\,\mathrm{A}, \mathrm{B}\,\}$ and $\mathrm{dom}(y) = \{\,\mathrm{G}, \mathrm{H}\,\}$.

$$\mathrm{Mod}\left(\overset{T}{\begin{array}{|c|c|}\hline \mathrm{A} & y \\ x & \mathrm{H} \\ \hline\end{array}}\right) = \left\{ \begin{array}{|c|c|}\hline \mathrm{A} & \mathrm{G} \\ \mathrm{A} & \mathrm{H} \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathrm{A} & \mathrm{G} \\ \mathrm{B} & \mathrm{H} \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathrm{A} & \mathrm{H} \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathrm{A} & \mathrm{H} \\ \mathrm{B} & \mathrm{H} \\ \hline\end{array} \right\}$$

- $T^R = \emptyset$
- $T^{\pi_N(R)} = \{\,\mathrm{A}\,\}$
- $T^{\pi_S(R)} = \{\,\mathrm{H}\,\}$

Intuition: Uncertain tuples that remain after “applying” q are omitted.

$\mathscr{L}$-equivalency

Definition

Two sets of incomplete databases $\mathcal{I}$ and $\mathcal{J}$ are $\mathscr{L}$*-equivalent*, denoted $\mathcal{I} \equiv_{\mathscr{L}} \mathcal{J}$ if $\mathcal{I}^q = \mathcal{J}^q$ for all $\mathscr{L}$-expressions q.

Example

$$\mathcal{I} = \left\{ \begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Hum. bird} \\ \hline \end{array}, \begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \text{Bob} & \text{Kingfisher} \\ \hline \end{array} \right\}$$

$$\mathcal{J} = \left\{ \begin{array}{|c|c|}\hline \text{Anna} & \text{Guan} \\ \hline \end{array} \right\}$$

- $\mathcal{I}$ and $\mathcal{J}$ are $\emptyset$-equivalent
- But: $\mathcal{I}$ and $\mathcal{J}$ are *not* P-equivalent (consider π_A)

$\mathscr{L}$-equivalent databases are indistinguishable w.r.t. the certain tuples in the query result.

More examples of $\mathscr{L}$-equivalency

Example

$$\mathcal{I} = \left\{ \begin{array}{|c|c|c|}\hline a_1 & b_1 & c_1 \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline a_1 & b_2 & c_2 \\ a_2 & b_1 & c_2 \\ \hline \end{array} \right\}$$

$$\mathcal{J} = \left\{ \begin{array}{|c|c|c|}\hline a_1 & b_1 & c_1 \\ \hline \end{array}, \begin{array}{|c|c|c|}\hline a_1 & b_2 & c_2 \\ a_2 & b_1 & c_3 \\ \hline \end{array} \right\}$$

- $\mathcal{I}$ and $\mathcal{J}$ are $\emptyset$-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are P-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are J-equivalent
- $\mathcal{I}$ and $\mathcal{J}$ are *not* PJ-equivalent; e.g., set

  $$q(R) = \pi_{AB}(\pi_{AC}(R) \bowtie \pi_{BC}(R)).$$

  Then $a_1 b_1 \in \mathcal{I}^q$ but $a_1 b_1 \notin \mathcal{J}^q$.

Weak representation system

Definition

A representation system is *weak* under a query language $\mathscr{L}$ if for any $\mathscr{L}$-expression q and any table T there is a computable table $\bar{q}(T)$ that $\mathscr{L}$-represents $q(\mathrm{Mod}(T))$.

$$\mathrm{Mod}(\bar{q}(T)) \equiv_{\mathscr{L}} q(\mathrm{Mod}(T)).$$

$$\begin{tikzcd}
T \arrow[rr, "\mathrm{Mod}"] \arrow[dd, "\bar{q}"'] & & \mathrm{Mod}(T) \arrow[d, "q"] \\
 & & q(\mathrm{Mod}(T)) \\
\bar{q}(T) \arrow[r, "\mathrm{Mod}"] & \mathrm{Mod}(\bar{q}(T)) \arrow[ur, dash, dashed, "\equiv_{\mathscr{L}}"'] &
\end{tikzcd}$$

Weak representation systems correctly determine the certain tuples under $\mathscr{L}$.

PS on Codd-Tables

Theorem

Codd tables are weak under PS.

$$\bar{\sigma}_P(T) = \{\, t \mid t \in T \textit{ and } P(v(t)) \textit{ for all valuations for } \mathrm{Var}(T) \,\}$$

$$\bar{\pi}_U(T) = \pi_U(T)$$

Example

T

Name	*Species*	*Location*
Anna	Guan	@
@	@	Paris
Bob	Kingf.	@

$\bar{\sigma}_{N=\mathrm{B}}(T)$

N	*S*	*L*
B	K	@

$\bar{\pi}_{NS}(T)$

N	*S*
A	G
@	@
B	K

$\bar{\pi}_{NS}(\bar{\sigma}_{N=\mathrm{B}}(T))$

N	*S*
B	K

These are single-relation queries!

PJ/PSU on Codd-Tables

Theorem

Codd tables are not weak under PJ or PSU.

Proof (for PJ).

- Consider Codd table T and set $\mathcal{I} = \text{Mod}(T)$
- Set $q(R) = \pi_{AC}(R) \bowtie \pi_B(R)$
- c-table $T_{q,c}$ represents $\mathcal{I}_q = q(\text{Mod}(T))$.
- Suppose Codd table T_q PJ-represents $\mathcal{I}_q$
- Consider $q' = \pi_{AC}(\pi_{AB}(R) \bowtie \pi_{BC}(R))$
- For each valuation v, T_q must contain tuples t_1, t_2 s.t. $t_1.A = a_2$, $t_2.C = c_1$, and $v(t_1).B = v(t_2).B$
  1. $t_1 = t_2$, then $a_2 c_1 \in T_q^{\pi_{AC}}$ but $a_2 c_1 \notin \mathcal{I}_q^{\pi_{AC}}$ $\rightarrow \lightning$
  2. $t_1 \neq t_2$, then $t_1.B = t_2.B = b$, then $a_2 b \in T_q^{\pi_{AB}}$ for some b but $\mathcal{I}_q^{\pi_{AB}} = \emptyset$ $\rightarrow \lightning$

T

A	B	C
a_1	x	c_1
a_2	y	c_2

$\mathcal{I}_q^{q'}$

A	C
a_1	c_1
a_1	c_2
a_2	c_1
a_2	c_2

$T_{q,c}$

A	B	C
a_1	x	c_1
a_1	y	c_1
a_2	x	c_2
a_2	y	c_2

□

Null values in SQL

SQL null semantics is related but not equal to Codd tables → Be careful!

Example

On PostgreSQL.

- $\sigma_{B=1}(T) \rightarrow$ `SELECT * FROM T WHERE B=1`
- $\pi_{AC}(T) \rightarrow$ `SELECT DISTINCT A, C FROM T`

T

A	B	C
1	null	1
2	null	2

$\sigma_{B=1}(T)$

A	B	C

$\sigma_{B\neq 1}(T)$

A	B	C

$\sigma_{B=1 \vee B\neq 1}(T)$

A	B	C

$\pi_{AC}(T)$

A	C
1	1
2	2

$\pi_{B}(T)$

B
null

$T_q = \pi_{AC}(T) \bowtie \pi_{B}(T)$

A	B	C
1	null	1
2	null	2

$T_q^{q'} = \pi_{AC}(\pi_{AB}(T_q) \bowtie \pi_{BC}(T_q))$

A	C

Positive $\mathcal{RA}$ on v-Tables

Theorem

v-tables are weak under the positive $\mathcal{RA}$. To obtain $\bar{q}$, simply treat variables as distinct constants and use standard $\mathcal{RA}$ operators.

Example

Sightings

N	*S*
A	G
A	H
z_1	K
z_2	L

VIPs

N
A
B
z_1

$\bar{\sigma}_{N=A}(S)$

N	*S*
A	G
A	H

$S \bar{\bowtie} V$

N	*S*
A	G
A	H
z_1	K

$\bar{\pi}_S(S \bar{\bowtie} V)$

S
G
H
K

Easy to do in an off-the-shelf relational database system!

PS^- on v-tables

Theorem

v-tables are not weak under PS^-.

Proof.

- Consider v-table T and set $\mathcal{I} = \mathrm{Mod}(T)$
- Set $q(R) = \sigma_{(A=a_1 \wedge B=b) \vee (A=a_2 \wedge B \neq b)}(R)$
- c-table $T_{q,c}$ represents $\mathcal{I}_q = q(\mathrm{Mod}(T))$.
- Suppose v-table T_q PS^--represents $\mathcal{I}_q$
- Consider $q'(R) = \pi_C(\sigma_{A=a_1 \vee A=a_2}(R))$
    1. $(\exists t \in T_q)\, t_1.A = a_i$, then $a_i \in T_q^{\pi_A} \rightarrow \lightning$
    2. $(\forall t \in T_q)\, t.A \in \mathrm{Var}(T)$, then $T_q^{q'} = \emptyset \rightarrow \lightning$

T

A	B	C
a_1	x	c
a_2	x	c

$\mathcal{I}_q^{q'}$

C
c

$T_{q,c}$

A	B	C	con
a_1	x	c	$x = b$
a_2	x	c	$x \neq b$

$\square$

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Algebraic Completion

Definition

Let $(\mathscr{T}, \mathrm{Mod})$ be a representation system and $\mathscr{L}$ be a query language. The representation system obtained by *closing* $\mathscr{T}$ *under* $\mathscr{L}$ is the set of tables $\{\, (T, q) \mid T \in \mathscr{T}, q \in \mathscr{L} \,\}$ and function $\mathrm{Mod}(T, q) = q(\mathrm{Mod}(T))$.

Example

No Codd table for $\mathcal{I}$, but closure of f.d. Codd tables under JR suffices.

$$\mathcal{I} = \left\{ \begin{array}{|c|c|}\hline A & B \\ \hline a_1 & a_1 \\ \hline \end{array}, \begin{array}{|c|c|}\hline A & B \\ \hline a_2 & a_2 \\ \hline \end{array} \right\}, \quad T = \begin{array}{|c|}\hline A \\ \hline a_1 \| a_2 \\ \hline \end{array}, \quad q(R) = R \bowtie \rho_{A \to B}(R)$$

- Think of q as a *view* over T
- View result need not be represented directly

Algebraic completion extends the power of a representation system with the power of a query language.

$\mathcal{RA}$-completion for Codd tables

Theorem

The closure of Codd tables under SPJRU is $\mathcal{RA}$-complete.

Proof.

- c-tables are $\mathcal{RA}$-complete
- Every c-table T can be $\mathcal{RA}$-defined by an SPJRU-query q on $\mathcal{Z}_V$ (see slide 46)
- $\mathcal{Z}_V$ can be represented as a Codd table T'

$T' =$

B_1	B_2	...	B_k
@	@	...	@

- $\mathrm{Mod}(T', q) = q(\mathrm{Mod}(T')) = q(\mathcal{Z}_V) = \mathrm{Mod}(T)$

□

Relational databases with views can represent *any* $\mathcal{RA}$-definable database!

$\mathcal{RA}$-completion for v-tables

Theorem

The closure of v-tables under S^+P is $\mathcal{RA}$-complete.

Proof.

Let $T = \{ t_1, \ldots, t_m \}$ be a c-table on $A_1 \ldots A_n$ and let $\mathrm{Var}(T) = \{ x_1, \ldots, x_k \}$. Express T in terms of v-table T' and query q:

$$T' = \begin{array}{|c|c|c|c|c|c|c|}
\hline
A_1 & \ldots & A_n & B_1 & \ldots & B_k & C \\
\hline
t_1.A_1 & \ldots & t_1.A_n & x_1 & \ldots & x_k & 1 \\
t_2.A_1 & \ldots & t_2.A_n & x_1 & \ldots & x_k & 2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
t_m.A_1 & \ldots & t_m.A_n & x_1 & \ldots & x_k & m \\
\hline
\end{array}$$

$$q(R) = \pi_{A_1 \ldots A_n}(\sigma_{\bigvee_{i=1}^{m}(\psi_i \wedge C=i)}(R))$$

where ψ_i is obtained from $t_i.con$ by replacing all variables x_j by the corresponding attribute B_j. □

Finite completion results

Theorem

The following closures are finitely complete:

1. *or-set-tables under PJ,*
2. *finite v-tables under PJ or S^+P,*
3. *?-tables under $\mathcal{RA}$.*

Proof.

Try it yourself. Hints: Don't start with a c-table, but an incomplete database $\mathcal{I}$. You need two tables for cases 1 and 2; case 3 is quite tricky. ☐

Outline

1. Refresher: Relational Algebra
2. Incomplete Databases
3. Strong representation systems
4. Completeness
5. Weak Representation Systems
6. Completion
7. Summary

Lessons learned

- Incomplete databases are sets of possible databases
- Representation systems are concise descriptions of incomplete databases
- Queries can be analyzed in terms of
  1. Possible answer sets (strong representation)
  2. Certain answer tuples (weak representation)
  3. Possible answer tuples (finite i-databases only)
- Codd tables add null values; weak under PS
  → Be careful with null values in SQL
- v-tables add variables; weak under positive $\mathcal{RA}$
- c-tables add variables and conditions; strong under $\mathcal{RA}$ and $\mathcal{RA}$-complete
- $\mathcal{RA}$-views on Codd tables are $\mathcal{RA}$-complete → key property!

Suggested reading

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 2)
  Springer, 2009

- Dan Suciu, Dan Olteanu, Christopher Ré, Christoph Koch
  Probabilistic Databases (Chapter 2)
  Morgan & Claypool, 2011

- Serge Abiteboul, Richard Hull, Victor Vianu
  Foundations of Databases: The Logical Level (Chapter 19)
  Addison Wesley, 1994

- Tomasz Imieliński, Witold Lipski, Jr.
  Incomplete Infomation in Relational Databases
  Journal of the ACM, 31(4), Oct. 1984

Scalable Uncertainty Management

03 – Provenance

Rainer Gemulla

May 18, 2012

Overview

In this lecture

- Introduction to datalog
- What is provenance?
- Which types of provenance do exist?
    - Lineage
    - Why-provenance
    - How-provenance
- How to compute provenance?
- How do the types of provenance relate to each other?
- How to derive provenance information for datalog?

Not in this lecture

- Uncertainty
- Where-provenance

Outline

1. Datalog
2. Introduction to Provenance
    - Lineage
    - Why-provenance
    - How-provenance
3. Provenance Semirings
4. How-Provenance for nr-datalog
5. Summary

Datalog

- Datalog is a declarative language
- Datalog program is collection of if-then rules
- Supports recursion (in contrast to relational algebra)
- *Datalog* is a logic for relations ("database logic")
- Datalog is based on Prolog
  - No function symbols + safety condition
  - Unique and finite minimum model
  - Unique and finite minimum fixpoint
  - Expressive power in PTIME

Example

$$\text{ancestor}(x, z) \leftarrow \text{parent}(x, z)$$

$$\text{ancestor}(x, z) \leftarrow \text{ancestor}(x, y),\ \text{parent}(y, z)$$

Straightforward translation to first-order logic:

$$\begin{array}{ll} (\forall x)(\forall z) & \text{parent}(x, z) \rightarrow \text{ancestor}(x, z) \\ (\forall x)(\forall y)(\forall z) & \text{ancestor}(x, y) \wedge \text{parent}(y, z) \rightarrow \text{ancestor}(x, z) \end{array}$$

Predicates and atoms

- Relations are represented by *predicates* of same arity
  - For relation name R, we use predicate name R
  - Order of predicate arguments = natural order of relation attributes
- Predicate with arguments is called a *relational atom*
  - $R(a_1, \ldots, a_k)$ returns $\texttt{TRUE}$ if $(a_1, \ldots, a_k) \in I(R)$
  - $\texttt{FALSE}$ otherwise (*closed word assumption*)
- Predicate can take *constants* and *variables* as arguments
  - Atom with variables = function that takes values for variables and returns $\texttt{TRUE}/\texttt{FALSE}$

Example

For simplicity, we denote both predicate and its interpretation by R.

- $R(a_1, b_1) = \texttt{TRUE}$
- $R(a_2, b_2) = \texttt{TRUE}$
- $R(a_3, b_3) = \texttt{FALSE}$
- $R(x, b_1) = f(x) = \begin{cases} \texttt{TRUE} & \text{if } x = a_1 \\ \texttt{FALSE} & \text{otherwise} \end{cases}$

R

A	B
a_1	b_1
a_2	b_2

Extended datalog: arithmetic atoms

- Comparison between two arithmetic expressions
  - Arithmetic predicates: $=, <, >, \leq, \geq, \ldots$
  - Arithmetic expressions: constants, variables, $+, -, \times, /, \ldots$
- Arithmetic predicates are like infinite relations
  - Database relations are finite and may change
  - Arithmetic relations are infinite and unchanging

Example

- $x < y$
- $x + 1 \geq y + 4 \times z$
- $x < 5 = f(x) = \begin{cases} \texttt{TRUE} & \text{if } x < 5 \\ \texttt{FALSE} & \text{otherwise} \end{cases}$
- $\text{“}<\text{”} = \{\,(1, 2),\ (-1.5, 65.4),\ \ldots\,\}$

Datalog rules

- Operations are described by *datalog rules*
  1. A relational atom called *head*
  2. The symbol $\leftarrow$ (read as "if")
  3. A *body* consisting of one or more atoms, called *subgoals* (connected by $\wedge$; in datalog$^\neg$: optionally preceded by $\neg$)

Example

A movie schema:

Movies(Title, Year, Length, Genre, StudioName, Producer).

A $\mathcal{RA}$ expression:

$$\text{LongMovie} := \pi_{\text{Title},\text{Year}}(\sigma_{\text{Length}\geq 100}(\text{Movies})).$$

Corresponding datalog rule:

$$\underbrace{\text{LongMovie}(t,y)}_{\text{head}} \leftarrow \underbrace{\overbrace{\text{Movies}(t,y,l,g,s,p)}^{\text{subgoal 1}},\ \overbrace{l \geq 100}^{\text{subgoal 2}}}_{\text{body}}.$$

Semantics of rules

1. Possible assignments
   - Let the variables in the rule range over all possible values
   - When all subgoals are `TRUE`, insert tuple into the head's relation
2. Nonnegated relational subgoals
   - Consider sets of tuples for each nonnegated relational subgoal
   - Check whether assignment is *consistent* (same variable, same value)
   - If so, check negated subgoals and arithmetic subgoals
   - If all checks successful, insert tuple into the head's relation

Example

$$P(x,z) \leftarrow Q(x,y),\ R(y,z),\ \neg Q(x,z)$$

Q

1	2
1	3

R

2	3
3	1

	$Q(x,y)$	$R(y,z)$	Consistent?	$\neg Q(x,z)$?	Result
1)	$(1,2)$	$(2,3)$	Yes	No	—
2)	$(1,2)$	$(3,1)$	No; $y = 2, 3$	Irrelevant	—
3)	$(1,3)$	$(2,3)$	No; $y = 3, 2$	Irrelevant	—
4)	$(1,3)$	$(3,1)$	Yes	Yes	$P(1,1)$

CWA

Safe rules

Not all rules give a meaningful (i.e., finite) result → safety condition.

Example

- Safe:

  $\text{LongMovie}(t, y) \leftarrow \text{Movies}(t, y, l, g, s, p),\ l \geq 100$

- In safe rules, abbreviation _ for variables that occur only once

  $\text{LongMovie}(t, y) \leftarrow \text{Movies}(t, y, l, \_, \_, \_),\ l \geq 100$

- Unsafe: $P(x) \leftarrow Q(y)$
- Unsafe: $P(x) \leftarrow \neg Q(x)$
- Unsafe: $P(x, y) \leftarrow Q(y),\ x > y$

Definition

A rule is *safe* if every variable that appears anywhere in the rule also appears in some nonnegated, relational subgoal of the body. This condition is called the *safety condition*.

Extensional and intensional predicates

Definition

- *Extensional predicates* (EDB) are predicates whose relations are stored in a database. They can only occur in the bodies of datalog rules.
- *Intensional predicates* (IDB) are predicates whose relations is computed by applying datalog rules. They can occur in heads and bodies of datalog rules.

- "Extension" is another name for "instance of a relation"
- "Intensional" relations are defined by the programmer's "intent"

Example

$\text{LongMovie}(t, y) \leftarrow \text{Movies}(t, y, l, \_, \_, \_), \; l \geq 100$

- Movies is an EDB predicate (or relation)
- LongMovie is an IDB predicate (or relation)

Datalog queries

A *datalog query* is a collection of one or more rules (often with a designated output relation).

Example

Schema (EDB):

- Hotel(HotelNo, Name, City)
- Room(RoomNo, HotelNo, Type, Price)

$\mathcal{RA}$ query:

$$\pi_{\text{HotelNo,Name,City}}(\text{Hotel} \bowtie \sigma_{\text{Price}>500 \vee \text{Type}=\text{'suite'}}(\text{Room}))$$

Datalog query:

$$\begin{aligned}
\text{ExpensiveRoom}(r,h,t,p) &\leftarrow \text{Room}(r,h,t,p),\ p > 500\\
\text{ExpensiveRoom}(r,h,t,p) &\leftarrow \text{Room}(r,h,t,p),\ t = \text{'suite'}\\
\text{ExpensiveHotelRoom}(h,n,c,r,t,p) &\leftarrow \text{Hotel}(h,n,c),\ \text{ExpensiveRoom}(r,h,t,p)\\
\text{ExpensiveHotel}(h,n,c) &\leftarrow \text{ExpensiveHotelRoom}(h,n,c,\_,\_,\_)
\end{aligned}$$

Datalog and relational algebra

Example (Recursive query)

$$\text{ancestor}(x, z) \leftarrow \text{parent}(x, z)$$

$$\text{ancestor}(x, z) \leftarrow \text{ancestor}(x, y), \text{parent}(y, z)$$

- *Nonrecursive* if the rules can be ordered such that the head predicate of each rule does not occur in a body of the current or a previous rule
- *nr-datalog*: nonrecursive, no negation
- *nr-datalog*$^\neg$: nonrecursive, with negation

Theorem

- *nr-datalog and SPJRU queries have equivalent expressive power.*
- *nr-datalog*$^\neg$ *and relational algebra have equivalent expressive power.*

We will switch between datalog and (subsets of) $\mathcal{RA}$ as convenient.

Outline

1. Datalog

2. Introduction to Provenance
    - Lineage
    - Why-provenance
    - How-provenance

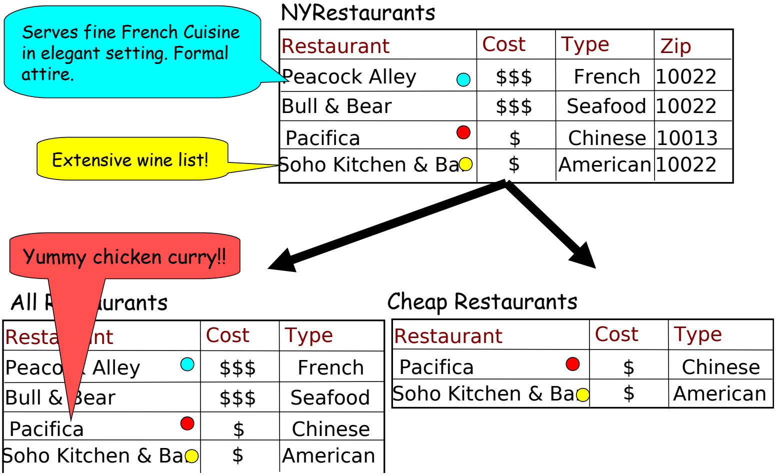
3. Provenance Semirings

4. How-Provenance for nr-datalog

5. Summary

Provenance and annotation management

- *Provenance* describes origins and history of data
- *Annotations* describe auxiliary information associated with the data



Serves fine French Cuisine in elegant setting. Formal attire.

Extensive wine list!

Yummy chicken curry!!

NYRestaurants

Restaurant	Cost	Type	Zip
Peacock Alley 🔵	$$$	French	10022
Bull & Bear	$$$	Seafood	10022
Pacifica 🔴	$	Chinese	10013
Soho Kitchen & Ba 🟡	$	American	10022

All R urants

Resta nt	Cost	Type
Peaco Alley 🔵	$$$	French
Bull & ear	$$$	Seafood
Pacifica 🔴	$	Chinese
Soho Kitchen & Ba 🟡	$	American

Cheap Restaurants

Restaurant	Cost	Type
Pacifica 🔴	$	Chinese
Soho Kitchen & Ba 🟡	$	American

Chiticariu, VLDB, 2004.

Outline

1. Datalog
2. Introduction to Provenance
   - Lineage
   - Why-provenance
   - How-provenance
3. Provenance Semirings
4. How-Provenance for nr-datalog
5. Summary

Tuple location

Definition

A tuple t tagged with a relation name R is called a *tuple location* and denoted (R, t) or simply $R(t)$. We can view a database instance $I(\mathbf{R})$ on $\mathbf{R}$ as a set $\{\, (R, t) \mid R \in \mathbf{R},\ t \in I(R) \,\}$.

Example

Agencies (A)

	Name	BasedIn	Phone
t_1	BayTours	SFO	415-1200
t_2	HarborCruz	SC	831-3000

ExternalTours (E)

	Name	Dest.	Type	Price
t_3	BayTours	SFO	Cable	\$50
t_4	BayTours	SC	Bus	\$100
t_5	BayTours	SC	Boat	\$250
t_6	BayTours	MRY	Boat	\$400
t_7	HarborCruz	MRY	Boat	\$200
t_8	HarborCruz	Carmel	Train	\$90

- Tuple locations: $A(t_1), A(t_2), A(\langle \text{FunTravel}, \text{SJ}, \text{415-2400} \rangle), \ldots$
- Database instance: $\{\, A(t_1), A(t_2), E(t_3), E(t_4), \ldots, E(t_8) \,\}$

Lineage

Definition (informal)

The *lineage* of a tuple t (w.r.t. a query) consists of all tuples of the input data that “contributed to” or “helped produce” t.

Example

Agencies (A)

	Name	BasedIn	Phone
t_1	BayTours	SFO	415-1200
t_2	HarborCruz	SC	831-3000

ExternalTours (E)

	Name	Dest.	Type	Price
t_3	BayTours	SFO	Cable	\$50
t_4	BayTours	SC	Bus	\$100
t_5	BayTours	SC	Boat	\$250
t_6	BayTours	MRY	Boat	\$400
t_7	HarborCruz	MRY	Boat	\$200
t_8	HarborCruz	Carmel	Train	\$90

$$\begin{aligned} &\text{BoatAgencies}(n, p) \leftarrow \\ &\qquad \text{Agencies}(n, \_, p), \\ &\qquad \text{ExternalTours}(n, \_, \text{'Boat'}, \_). \end{aligned}$$

BoatAgencies

Name	Phone	*Lineage*
BayTours	415-1200	$\{ A(t_1), E(t_5), E(t_6) \}$
HarborCruz	831-3000	$\{ A(t_2), E(t_7) \}$

Lineage & query rewriting

Example

Two equivalent queries:

$$
\begin{aligned}
q(x,y) &\leftarrow R(x,y)\\
q'(x,y) &\leftarrow R(x,y), R(x,z).
\end{aligned}
$$

R

	A	B
t_1	1	2
t_2	1	3
t_3	4	2

$q(R)$

A	B	*Lineage*
1	2	$\{\, R(t_1) \,\}$
1	3	$\{\, R(t_2) \,\}$
4	2	$\{\, R(t_3) \,\}$

$q'(R)$

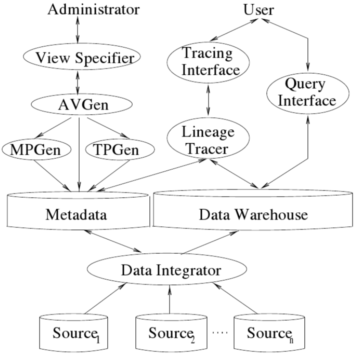
A	B	*Lineage*
1	2	$\{\, R(t_1), R(t_2) \,\}$
1	3	$\{\, R(t_1), R(t_2) \,\}$
4	2	$\{\, R(t_3) \,\}$

Theorem

Lineage is sensitive to query rewriting.

Application: Lineage tracing in data warehouses

- Data warehouses integrates data from multiple sources
- Warehouse directly used for coarse-grained analysis
- In-depth analysis requires access to source data
  → *view data lineage* problem



Lineage tracing in the WHIPS data warehouse system

Cui et al., TODS 25(2), 2000.

Outline

1. Datalog

2. Introduction to Provenance
    - Lineage
    - Why-provenance
    - How-provenance

3. Provenance Semirings

4. How-Provenance for nr-datalog

5. Summary

Witness

Definition

Let $\mathbf{I}$ be a database instance over $\mathbf{R}$, q a query over $\mathbf{R}$, and $t \in q(\mathbf{I})$. An instance $\mathbf{J} \subseteq \mathbf{I}$ is a *witness for t with respect to q* if $t \in q(\mathbf{J})$. The set of all witnesses is given by $\text{Wit}(q, \mathbf{I}, t) = \{ \mathbf{J} \subseteq \mathbf{I} \mid t \in q(\mathbf{J}) \}$.

Example

Agencies (A)

	Name	BasedIn	Phone
t_1	BayTours	SFO	415-1200
t_2	HarborCruz	SC	831-3000

ExternalTours (E)

	Name	Dest.	Type	Price
t_3	BayTours	SFO	Cable	\$50
t_4	BayTours	SC	Bus	\$100
t_5	BayTours	SC	Boat	\$250
t_6	BayTours	MRY	Boat	\$400
t_7	HarborCruz	MRY	Boat	\$200
t_8	HarborCruz	Carmel	Train	\$90

BoatAgencies

	Name	Phone	*Lineage*
t_9	BayTours	415-1200	$\{ A(t_1), E(t_5), E(t_6) \}$
t_{10}	HarborCruz	831-3000	$\{ A(t_2), E(t_7) \}$

Witnesses for

- t_9: $\{ A(t_1), E(t_5) \}$, $\{ A(t_1), E(t_6) \}$, $\{ A(t_1), E(t_5), E(t_6) \}$, ...
- t_{10}: $\{ A(t_2), E(t_7) \}$, $\{ A(t_1), A(t_2), E(t_7) \}$, ...
- $\mathbf{I}$ is a witness for both t_9 and t_{10}

Minimal why-provenance

Definition

A *minimal witness* is a minimal element of $\mathrm{Wit}(q, \mathbf{I}, t)$. The set of minimal witnesses is called *minimal why-provenance* and is given by $\mathrm{MWhy}(q, \mathbf{I}, t) = \{\, \mathbf{J} \in \mathrm{Wit}(q, \mathbf{I}, t) \mid (\forall \mathbf{J}' \in \mathrm{Wit}(q, \mathbf{I}, t))\ \mathbf{J}' = \mathbf{J} \vee \mathbf{J}' \not\subset \mathbf{J} \,\}$.

Example

Agencies (A)

	Name	BasedIn	Phone
t_1	BayTours	SFO	415-1200
t_2	HarborCruz	SC	831-3000

ExternalTours (E)

	Name	Dest.	Type	Price
t_3	BayTours	SFO	Cable	$50
t_4	BayTours	SC	Bus	$100
t_5	BayTours	SC	Boat	$250
t_6	BayTours	MRY	Boat	$400
t_7	HarborCruz	MRY	Boat	$200
t_8	HarborCruz	Carmel	Train	$90

BoatAgencies

	Name	Phone	*Minimal why-provenance*
t_9	BayTours	415-1200	$\{\,\{\,A(t_1), E(t_5)\,\}, \{\,A(t_1), E(t_6)\,\}\,\}$
t_{10}	HarborCruz	831-3000	$\{\,\{\,A(t_2), E(t_7)\,\}\,\}$

Minimal why-provenance & query rewriting

Example

Two equivalent queries:

$$\begin{aligned} q(x,y) &\leftarrow R(x,y) \\ q'(x,y) &\leftarrow R(x,y), R(x,z). \end{aligned}$$

R

	A	B
t_1	1	2
t_2	1	3
t_3	4	2

$q(R)$

A	B	*Min. why*
1	2	$\{\{R(t_1)\}\}$
1	3	$\{\{R(t_2)\}\}$
4	2	$\{\{R(t_3)\}\}$

$q'(R)$

A	B	*Min. why*
1	2	$\{\{R(t_1)\}\}$
1	3	$\{\{R(t_2)\}\}$
4	2	$\{\{R(t_3)\}\}$

Theorem

Minimal why-provenance is insensitive to query rewriting.

Application: View deletion problem

- Let **I** be a database instance and consider view $V = q(\mathbf{I})$
- *View deletion problem*: Find the set of tuples $\Delta\mathbf{I}$ to remove from **I** so that a tuple t is removed from V
- Intuitively, all minimal witnesses must be destroyed; many ways, e.g.,
  1. Source side-effect problem: Minimize changes to the source ($|\Delta\mathbf{I}|$)
  2. View side-effect problem: Minimize changes to the view ($|\Delta V|$)
- Both NP-hard for PJ and JU queries!

Example

BayTours does not offer boat tours anymore $\rightarrow$ delete t_9.

BoatAgencies

	Name	Phone	*Min. why*
t_9	BayTours	415-1200	$\{\{A(t_1), E(t_5)\}, \{A(t_1), E(t_6)\}\}$
t_{10}	HarborCruz	831-3000	$\{\{A(t_2), E(t_7)\}\}$

Examples:

- delete $A(t_1)$: optimum for both problems
- delete $E(t_5)$ and $E(t_6)$: optimum for (1) when $A \bowtie E$ is taken as source

Buneman et al., PODS, 2002.

Outline

1. Datalog
2. Introduction to Provenance
    - Lineage
    - Why-provenance
    - How-provenance
3. Provenance Semirings
4. How-Provenance for nr-datalog
5. Summary

How-provenance

Definition (informal)

The *how-provenance* of a tuple t describes how t is derived according to the query. It makes use of two "operations": combine ($\cdot$) and merge ($+$).

Example

Agencies (A)

	Name	BasedIn	Phone
t_1	BayTours	SFO	415-1200
t_2	HarborCruz	SC	831-3000

ExternalTours (E)

	Name	Dest.	Type	Price
t_3	BayTours	SFO	Cable	\$50
t_4	BayTours	SC	Bus	\$100
t_5	BayTours	SC	Boat	\$250
t_6	BayTours	MRY	Boat	\$400
t_7	HarborCruz	MRY	Boat	\$200
t_8	HarborCruz	Carmel	Train	\$90

BoatAgencies

Name	Phone	*How-provenance*
BayTours	415-1200	$A(t_1) \cdot E(t_5) + A(t_1) \cdot E(t_6)$
HarborCruz	831-3000	$A(t_2) \cdot E(t_7)$

How-provenance & query rewriting

Example

Two equivalent queries:

$$q(x, y) \leftarrow R(x, y)$$
$$q'(x, y) \leftarrow R(x, y), R(x, z).$$

R

	A	B
t_1	1	2
t_2	1	3
t_3	4	2

$q(R)$

A	B	*How*
1	2	$R(t_1)$
1	3	$R(t_2)$
4	2	$R(t_3)$

$q'(R)$

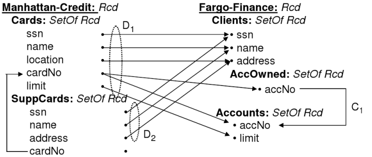
A	B	*How*
1	2	$R(t_1)^2 + R(t_1) \cdot R(t_2)$
1	3	$R(t_2)^2 + R(t_1) \cdot R(t_2)$
4	2	$R(t_3)^2$

Theorem

How-provenance is sensitive to query rewriting.

Application: Debugging of schema mappings

- Data exchange between two applications (source and target)
- Schema mapping relates data from source application to data from target application
- Schema debuggers help in developing such a mapping



Manhattan-Credit: *Rcd*
Cards: *SetOf Rcd*
ssn
name
location
cardNo
limit
SuppCards: *SetOf Rcd*
ssn
name
address
cardNo

Fargo-Finance: *Rcd*
Clients: *SetOf Rcd*
ssn
name
address
AccOwned: *SetOf Rcd*
accNo
Accounts: *SetOf Rcd*
accNo
limit

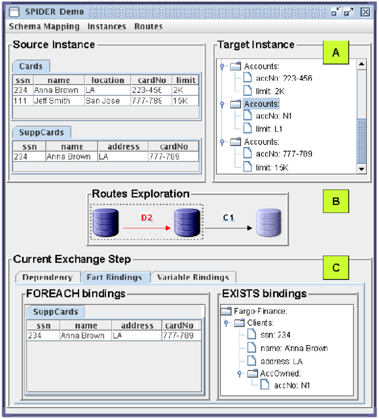
Source-to-target dependencies:

D_1: foreach x_1 in Manhattan-Credit.Cards
exists x_2 in Fargo-Finance.Clients, x_3 in x_2.AccOwned,
x_4 in Fargo-Finance.Accounts
where x_3.accNo=x_4.accNo
with x_1.ssn= x_2.ssn and x_1.name= x_2.name and x_1.location= x_2.address
and x_1.cardNo= x_3.accNo and x_1.limit= x_4.limit

D_2: foreach x_1 in SuppCards
exists x_2 in Fargo-Finance.Clients
with x_1.ssn= x_2.ssn and x_1.name= x_2.name and x_1.address= x_2.address

Target dependency:

C_1: foreach x_1 in Fargo-Finance.Clients, x_2 in x_1.AccOwned
exists x_3 in Fargo-Finance.Accounts with x_2.accNo=x_3.accNo



Source Instance

Cards

ssn	name	location	cardNo	limit
234	Anna Brown	LA	223-456	2K
111	Jeff Smith	San Jose	777-789	15K

SuppCards

ssn	name	address	cardNo
234	Anna Brown	LA	777-789

Target Instance

Accounts:
accNo: 223-456
limit: 2K
Accounts:
accNo: N1
limit: L1
Accounts:
accNo: 777-789
limit: 15K

Routes Exploration

Current Exchange Step

FOREACH bindings

SuppCards

ssn	name	address	cardNo
234	Anna Brown	LA	777-789

EXISTS bindings

Fargo-Finance:
Clients:
ssn: 234
name: Anna Brown
address: LA
AccOwned:
accNo: N1

Alexe et al., VLDB, 2006.

Outline

1. Datalog

2. Introduction to Provenance
    - Lineage
    - Why-provenance
    - How-provenance

3. Provenance Semirings

4. How-Provenance for nr-datalog

5. Summary

Provenance through annotations

Example

Agencies

Name	BasedIn	Phone	
BayTours	SFO	415-1200	t_1
HarborCruz	SC	831-3000	t_2

ExternalTours

Name	Dest.	Type	
BayTours	SFO	Cable	t_3
BayTours	SC	Bus	t_4
BayTours	SC	Boat	t_5
BayTours	MRY	Boat	t_6

$$\pi_{\text{Dest,Phone}}(\text{Agencies} \bowtie [\pi_{\text{Name,Dest}}(\rho_{\text{BasedIn}\rightarrow\text{Dest}}(\text{Agencies})) \cup \pi_{\text{Name,Dest}}(\text{ExternalTours})])$$

Dest	Phone	
SFO	415-1200	$t_1 \cdot (t_1 + t_3)$
SC	831-3000	t_2^2
SC	415-1200	$t_1 \cdot (t_4 + t_5)$
MTY	415-1200	$t_1 \cdot t_6$

We need a way to annotate relations and propagate these annotations.

K-relation

Definition

A *K-relation* is a function R that maps each tuple in the relation to nonzero elements of K, and each tuple not in the relation to a special element $0 \in K$. R has finite support $\text{supp}(R) = \{ t \mid R(t) \neq 0 \}$.

Intuitively, each tuple t is *annotated* with an element of K.

Example

1. $\mathbb{B}$-relations correspond to ordinary relations (zero element: `FALSE`)
2. $\mathbb{N}$-relations correspond to multisets or bags (zero element: 0)
3. $\mathscr{C}$-relations correspond to boolean c-tables (zero element: `FALSE`)
4. TupleLoc-relations (zero element: $\perp$)

A (1)

Name	
BayTours	`TRUE`
HarborCruz	`TRUE`

A (2)

Name	
BayTours	2
HarborCruz	5

A (3)

Name	
BayTours	x
HarborCruz	$\neg x$

A (4)

Name	
BayTours	$A(t_1)$
HarborCruz	$A(t_2)$

Positive K-relational algebra

Definition

Let $(K, 0, 1, +, \cdot)$ be an algebraic structure with two binary operators $+$ (*merge*) and $\cdot$ (*combine*) and two distinguished elements 0 (*not in relation*) and 1 (*in relation*). Let $q^K(I)t$ be the annotation of t in $q(I)$. The operations of the positive K-relational algebra are defined as follows:

Value	$(\{\,\langle A : a\rangle\,\})^K(I)t = \begin{cases} 1 & \text{if } t = \langle A : a\rangle \\ 0 & \text{otherwise} \end{cases}$	1
Relation	$R^K(I)t = I(R)t$	Copy
Selection	$(\sigma_\theta(q))^K(I)t = \begin{cases} q^K(I)t & \text{if } \theta(t) \\ 0 & \text{otherwise} \end{cases}$	Copy
Projection	$(\pi_U(q))^K(I)t = \sum_{t' \in \mathrm{supp}(q^K(I)),\, t'[U]=t} q^K(I)t'$	Merge
Union	$(q_1 \cup q_2)^K(I)t = q_1^K(I)t + q_2^K(I)t$	Merge
Join	$(q_1 \bowtie q_2)^K(I)t = q_1^K(I)t[U_1] \cdot q_2^K(I)t[U_2]$	Combine

Commutative semiring

Relational algebra over bags has the following properties:

- Union $(+)$ is associative and commutative, and has identity $\emptyset$
- Join $(\cdot)$ is associative, commutative, and distributes over union
- Projection and selection commute with each other as well as with union and join

Goal: Retain these properties with positive K-relational algebra.

Definition

$(K, 0, 1, +, \cdot)$ is a *commutative semiring* if:

- $(K, +, 0)$ is a commutative monoid (associative, commutative, identity 0),
- $(K, \cdot, 1)$ is a commutative monoid (associative, commutative, identity 1),
- $\cdot$ distributes over $+$,
- $0 \cdot a = a \cdot 0 = 0$ for all $a \in K$.

Common semirings

- How-provenance: $(\mathbb{N}[\mathsf{TupleLoc}], 0, 1, +, \cdot)$
  - TupleLoc denotes set of all tuple locations
  - $\mathbb{N}[K]$ = set of polynomials with coefficients in $\mathbb{N}$ and variables from K
  - $+$ and $\cdot$ have usual definitions
  - Start with $R^K(I)t = (R, t)$ if $t \in I(R)$, else 0

  Called *positive algebra provenance semiring*.
- Bag semantics: $(\mathbb{N}, 0, 1, +, \cdot)$
  - $+$ and $\cdot$ have usual definitions
  - Start with $R^K(I)t$ = multiplicity of t in $R(I)$
- Lineage: $(\mathscr{P}(\mathsf{TupleLoc}) \cup \{\perp\}, \perp, \emptyset, \cup_{\mathsf{L}}, \cup_{\mathsf{S}})$
  - lazy union $\cup_{\mathsf{L}}$: $\perp \cup X = X \cup \perp = X$ Merge
  - strict union $\cup_{\mathsf{S}}$: $\perp \cup X = X \cup \perp = \perp$ Combine
  - Start with $R^K(I)t = \{(R, t)\}$ if $t \in I(R)$, else $\perp$
- Minimal why-provenance: $(\mathscr{P}(\mathscr{P}(\mathsf{TupleLoc})), \emptyset, \{\emptyset\}, \cup_{\mathsf{Min}}, \uplus_{\mathsf{Min}})$
  - Min operator computes minimal elements (e.g., $\mathsf{Min}\{\{1\},\{1,2\}\} = \{\{1\}\}$)
  - pairwise union: $X \uplus_{\mathsf{Min}} Y = \mathsf{Min}\{x \cup y \mid x \in X, y \in Y\}$ Combine
  - Start with $R^K(I)t = \{\{(R, t)\}\}$ if $t \in I(R)$, else $\perp$

Common semirings (examples)

Example

Query:

$$q(x, y) \leftarrow R(x, y), R(x, z)$$

$$q(R) = \pi_{A,B}(R \bowtie \rho_{B \to C}(R))$$

How-provenance

R

A	B	
1	2	t_1
1	3	t_2
4	2	t_3

$q(R)$

A	B	
1	2	$t_1^2 + t_1 \cdot t_2$
1	3	$t_2^2 + t_1 \cdot t_2$
4	2	t_3^2

Bags

R

A	B	
1	2	2
1	3	3
4	2	1

$q(R)$

A	B	
1	2	10
1	3	15
4	2	1

Lineage

R

A	B	
1	2	$\{ t_1 \}$
1	3	$\{ t_2 \}$
4	2	$\{ t_3 \}$

$q(R)$

A	B	
1	2	$\{ t_1, t_2 \}$
1	3	$\{ t_1, t_2 \}$
4	2	$\{ t_3 \}$

Min. why-provenance

R

A	B	
1	2	$\{ \{ t_1 \} \}$
1	3	$\{ \{ t_2 \} \}$
4	2	$\{ \{ t_3 \} \}$

$q(R)$

A	B	
1	2	$\{ \{ t_1 \} \}$
1	3	$\{ \{ t_2 \} \}$
4	2	$\{ \{ t_3 \} \}$

Outline

1. Datalog

2. Introduction to Provenance
   - Lineage
   - Why-provenance
   - How-provenance

3. Provenance Semirings

4. How-Provenance for nr-datalog

5. Summary

Proof tree

Proof-theoretic semantics of datalog: A fact is in the result if there exists a proof for it using the rules and the database facts.

Definition

A proof tree of a fact A is a labeled tree where:

- Each vertex of the tree is labeled by a fact.
- Each leaf is labeled by an EDB fact from the base data.
- The root is labeled by A.
- For each internal vertex, there exists an instantiation $A_1 \leftarrow A_2, \ldots, A_n$ of a rule r such that the vertex is labeled A_1, its children are respectively labeled $A_2, \ldots, A_n$ and the edges are labeled r.

Proof tree (example)

Example

$$\begin{array}{lrl} r_1: & \text{ExpensiveRoom}(r,h) \leftarrow & \text{Room}(r,h,\_,p),\ p > \$500 \\ r_2: & \text{ExpensiveRoom}(r,h) \leftarrow & \text{Room}(r,h,t,\_),\ t = \text{'suite'} \\ r_3: & \text{ExpensiveHotelRoom}(h,r) \leftarrow & \text{Hotel}(h,\_,\_),\ \text{ExpensiveRoom}(r,h) \\ r_4: & \text{ExpensiveHotel}(h) \leftarrow & \text{ExpensiveHotelRoom}(h,\_) \end{array}$$

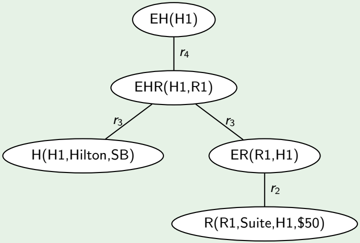
Room (R)

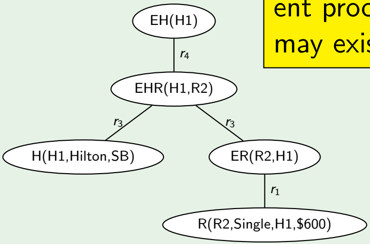
RoomNo	Type	HotelNo	Price
R1	Suite	H1	\$50
R2	Single	H1	\$600
R3	Double	H1	\$80

Hotel (H)

HotelNo	Name	City
H1	Hilton	SB

Multiple different proof trees may exist!





Lineage tree

Goal: Capture all ways of deriving an output fact.

Definition

A *lineage tree* of an nr-datalog query is computed with respect to the semiring $(\text{PosBool}(\mathscr{V}), \texttt{FALSE}, \texttt{TRUE}, \vee, \wedge)$, where

- $\mathscr{V}$ is a countable set of boolean variables,
- $\text{PosBool}(\mathscr{V})$ is the set of sets of equivalent boolean expressions involving $\texttt{TRUE}$, $\texttt{FALSE}$, variables from $\mathscr{V}$, $\vee$, and $\wedge$,
- Each fact is tagged with a representative from its class in $\text{PosBool}(\mathscr{V})$,
- Each EDB fact is tagged with a distinct variable from $\mathscr{V}$.

Example

$$\begin{aligned}\text{PosBool}(\{\, t_1, t_2 \,\}) = \{\, & \{\,\texttt{FALSE}\,\}, \{\,\texttt{TRUE}\,\} \\ & \{\, t_1,\ t_1 \vee t_1,\ t_1 \wedge \texttt{TRUE}, \ldots \}, \\ & \{\, t_2, \ldots \}, \{\, t_1 \vee t_2, \ldots \}, \{\, t_1 \wedge t_2, \ldots \} \}\end{aligned}$$

Lineage tree (example)

Example

$$\pi_{\text{HotelNo}}(\pi_{\text{HotelNo,RoomNo}}(\text{Hotel} \bowtie \pi_{\text{RoomNo,HotelNo}}(\sigma_{\text{price}>500 \lor \text{type='suite'}}(\text{Room}))))$$

Room (R)

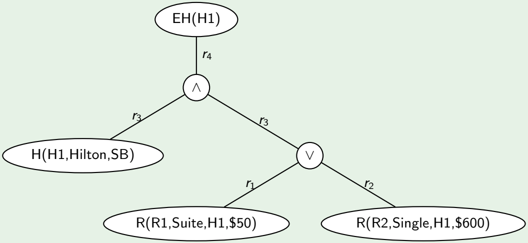
RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	t_1
R2	Single	H1	\$600	t_2
R3	Double	H1	\$80	t_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	t_4

ExpensiveHotels

HotelNo	
H1	$t_4 \land (t_1 \lor t_2)$



Not unique. There are many *different* trees, but all of them belong to the same PosBool equivalence class.

Outline

1. Datalog

2. Introduction to Provenance
   - Lineage
   - Why-provenance
   - How-provenance

3. Provenance Semirings

4. How-Provenance for nr-datalog

5. Summary

Lessons learned

- *Datalog* is a declarative language for relations
  - Based on Prolog
  - Collection of if-then rules
  - Closely related to relational algebra
- *Provenance* describes origins and history of data; *Annotation management* allows and propagates data annotations
  - Data warehousing, curated databases, annotated databases, update languages, *uncertain databases*, . . .
- Different types of provenance provide different amount of detail
  1. Lineage: *what* contributed to the output (tuples)
  2. Why-provenance: *why* an output tuple was produced (db instances)
  3. How-provenance: *how* an output tuple was produced (polynomial)
- Semirings are a natural way to study provenance
- Positive K-relational algebra can compute many forms of provenance
- Lineage trees are the preferred form of how-provenance for datalog (boolean formula)

Suggested reading

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom
  Database Systems: The Complete Book, 2nd ed. (ch. 5.3 & 5.4)
  Pearson Prentice Hall, 2009

- Serge Abiteboul, Richard Hull, Victor Vianu
  Foundations of Databases: The Logical Level (ch. 12)
  Addison Wesley, 1994

- James Cheney, Laura Chiticariu, Wang-Chiew Tan
  Provenance in Databases: Why, How, and Where
  Foundations and Trends in Databases, 1(4), 2007

Scalable Uncertainty Management

04 – Probabilistic Databases

Rainer Gemulla

Jun 1, 2012

Overview

In this lecture

- Refresher: Finite probability (not presented)
- What is a probabilistic database?
- How can probabilistic information be represented?
- How expressive are these representations?
- How to query probabilistic databases?

Not in this lecture

- Complexity
- Efficiency
- Algorithms

Outline

1. Refresher: Finite Probability

2. Probabilistic Databases

3. Probabilistic Representation Systems
   - pc-tables
   - Tuple-independent databases
   - Other common representation systems

4. Summary

Sample space

Definition

The *sample space* Ω of an *experiment* is the set of all possible *outcomes*. We henceforth assume that Ω is finite.

Example

- Toss a coin: $\Omega = \{ \text{Head}, \text{Tail} \}$
- Throw a dice: $\Omega = \{ 1, 2, 3, 4, 5, 6 \}$

In general, we cannot predict with certainty the outcome of an experiment in advance.

Event

Definition

An *event* $A \subseteq \Omega$ is a subset of the sample space. $\emptyset$ is called the *empty event*, Ω the *trivial event*. Two events A and B are *disjoint* if $A \cap B = \emptyset$.

Example

Coin:

- Outcome is a head: $A = \{\, \text{Head} \,\}$
- Outcome is head or tail: $A = \{\, \text{Head}, \text{Tail} \,\} = \{\, \text{Head} \,\} \cup \{\, \text{Tail} \,\}$
- Outcome is both head and tail: $A = \emptyset = \{\, \text{Head} \,\} \cap \{\, \text{Tail} \,\}$
- Outcome is not head: $A = \{\, \text{Tail} \,\} = \{\, \text{Head} \,\}^{\mathrm{c}}$

Die:

- Outcome is an even number: $A = \{\, 2, 4, 6 \,\} = \{\, 2 \,\} \cup \{\, 4 \,\} \cup \{\, 6 \,\}$
- Outcome is even and ≤ 3: $A = \{\, 2 \,\} = \{\, 2, 4, 6 \,\} \cap \{\, 1, 2, 3 \,\}$

When $A, B \subseteq \Omega$ are events, so are $A \cup B$, $A \cap B$, and A^{c}, representing 'A or B', 'A and B', and 'not A', respectively.

Probability space

Definition

A *probability measure* $(2^\Omega, \mathbb{P})$ is a function $\mathbb{P} : 2^\Omega \to [0,1]$ satisfying

a) $\mathbb{P}(\emptyset) = 0$, and $\mathbb{P}(\Omega) = 1$,

b) If $A_1, \ldots, A_n$ are pairwise disjoint, $\mathbb{P}(\bigcup_{i=1}^n A_n) = \sum_{i=1}^n \mathbb{P}(A_n)$.

The triple $(\Omega, 2^\Omega, \mathbb{P})$ is called a *probability space*.

Example

For $\omega \in \Omega$, we write $\mathbb{P}(\omega)$ for $\mathbb{P}(\{\omega\})$; $\{\omega\}$ called *elementary event*.

- Coin: $2^\Omega = \{\emptyset, \{\text{Head}\}, \{\text{Tail}\}, \{\text{Head}, \text{Tail}\}\}$
- Fair coin: $\mathbb{P}(\text{Head}) = \mathbb{P}(\text{Tail}) = \frac{1}{2}$
  Implied: $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\{\text{Head}, \text{Tail}\}) = 1$
- Fair dice: $\mathbb{P}(1) = \cdots = \mathbb{P}(6) = \frac{1}{6}$ (rest implied)
- Outcome is even: $\mathbb{P}(\{2,4,6\}) = \mathbb{P}(2) + \mathbb{P}(4) + \mathbb{P}(6) = \frac{1}{2}$
- Outcome is ≤ 3: $\mathbb{P}(\{1,2,3\}) = \mathbb{P}(1) + \mathbb{P}(2) + \mathbb{P}(3) = \frac{1}{2}$

Conditional probability

Definition

If $\mathbb{P}(B) > 0$, then the conditional probability that A occurs given that B occurs is defined to be

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Example

Two dice; prob. that total exceeds 6 given that first shows 3?

- $\Omega = \{1, \ldots, 6\}^2$
- Total exceeds 6: $A = \{(a, b) : a + b > 6\}$
- First shows 3: $B = \{(3, b) : 1 \leq b \leq 6\}$
- $A \cap B = \{(3, 4), (3, 5), (3, 6)\}$
- $\mathbb{P}(A \mid B) = \mathbb{P}(A \cap B) / \mathbb{P}(B) = \frac{3}{36} / \frac{6}{36} = \frac{1}{2}$

Independence

Definition

Two events A and B are called *independent* if $\mathbb{P}(A \cap B) = \mathbb{P}(A)\mathbb{P}(B)$.

If $\mathbb{P}(B) > 0$, implies that $\mathbb{P}(A \mid B) = \mathbb{P}(A)$.

Example

Two independent events:

- Die shows an even number: $A = \{2, 4, 6\}$
- Die shows at most 4: $B = \{1, 2, 3, 4\}$:
- $\mathbb{P}(A \cap B) = \mathbb{P}(\{2, 4\}) = \frac{1}{3} = \frac{1}{2} \cdot \frac{2}{3} = \mathbb{P}(A)\mathbb{P}(B)$

Not independent:

- Die shows an odd number: $C = \{1, 3, 5\}$
- $\mathbb{P}(A \cap C) = \mathbb{P}(\emptyset) = 0 \neq \frac{1}{2} \cdot \frac{1}{2} = \mathbb{P}(A)\mathbb{P}(C)$

Disjointness $\neq$ independence.

Conditional independence

Definition

Let A, B, C be events with $\mathbb{P}(\,C\,) > 0$. A and B are *conditionally independent given* C if $\mathbb{P}(\,A \cap B \mid C\,) = \mathbb{P}(\,A \mid C\,)\mathbb{P}(\,B \mid C\,)$.

Example

- Die shows an even number: $A = \{\,2, 4, 6\,\}$
- Die shows at most 3: $B = \{\,1, 2, 3\,\}$
- $\mathbb{P}(\,A \cap B\,) = \frac{1}{6} \neq \frac{1}{2} \cdot \frac{1}{2} = \mathbb{P}(\,A\,)\mathbb{P}(\,B\,)$
  $\rightarrow$ A and B are not independent
- Die does not show multiple of 3: $C = \{\,1, 2, 4, 5\,\}$
- $\mathbb{P}(\,A \cap B \mid C\,) = \frac{1}{4} = \frac{1}{2} \cdot \frac{1}{2} = \mathbb{P}(\,A \mid C\,)\mathbb{P}(\,B \mid C\,)$
  $\rightarrow$ A and B are conditionally independent given C

Product space

Definition

Let $(\Omega_1, 2^{\Omega_1}, \mathbb{P}_1)$ and $(\Omega_2, 2^{\Omega_2}, \mathbb{P}_2)$ be two probability spaces. Their *product space* is given by $(\Omega_{12}, 2^{\Omega_{12}}, \mathbb{P}_{12})$ with $\Omega_{12} = \Omega_1 \times \Omega_2$ and

$$\mathbb{P}_{12}\left(A_1 \times A_2 \right) = \mathbb{P}_1\left(A_1 \right) \mathbb{P}_2\left(A_2 \right).$$

Example

Toss two fair dice.

- $\Omega_1 = \Omega_2 = \{ 1, 2, 3, 4, 5, 6 \}$
- $\Omega_{12} = \{ (1, 1), \ldots, (6, 6) \}$
- First die: $A_1 = \{ 1, 2, 3 \} \subseteq \Omega_1$
- Second die: $A_2 = \{ 2, 3, 4 \} \subseteq \Omega_2$
- $\mathbb{P}_{12}\left(A_1 \times A_2 \right) = \mathbb{P}_1\left(A_1 \right) \mathbb{P}_2\left(A_2 \right) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

Product spaces combine the outcomes of several *independent* experiments into one space.

Random variable

Definition

A *random variable* is a function $X : \Omega \to \mathbb{R}$. We will write $\{ X = x \}$ or $\{ X \leq x \}$ for the events $\{ \omega : X(\omega) = x \}$ and $\{ \omega : X(\omega) \leq x \}$, respectively. The *probability mass function* of X is the function $f_X : \mathbb{R} \to [0, 1]$ given by $f_X(x) = \mathbb{P}(X = x)$; its *distribution function* is given by $F_X(x) = \mathbb{P}(X \leq x)$.

Example

Toss two dice:

- Sum of outcomes: $X((a, b)) = a + b$
- $f_X(6) = \mathbb{P}(X = 6) = \mathbb{P}(\{ (1, 5), (2, 4), (3, 3), (4, 2), (5, 1) \}) = \frac{5}{36}$
- $F_X(3) = \mathbb{P}(X \leq 3) = \mathbb{P}(\{ (1, 1), (1, 2), (2, 1) \}) = \frac{1}{12}$

The notions of conditional probability, independence (consider events $\{ X = x \}$ and $\{ Y = y \}$ for all x and y), and conditional independence also apply to random variables.

Expectation

Definition

The *expected value* of a random variable X is given by

$$\mathbb{E}\left[X \right] = \sum_{x} x\, f_X(x).$$

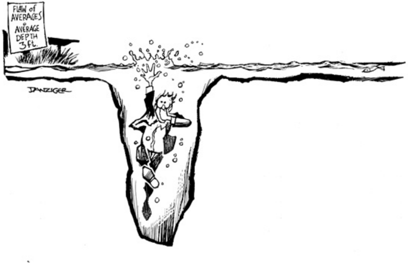
If $g : \mathbb{R} \to \mathbb{R}$, then

$$\mathbb{E}\left[g(X) \right] = \sum_{x} g(x) f_X(x).$$

Example

- Fair die (with X being identity)
- $\mathbb{E}\left[X \right] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + \cdots + 6 \cdot \frac{1}{6} = 3.5$
- Consider $g(x) = \lfloor x/2 \rfloor$
- $\mathbb{E}\left[g(x) \right] = 0 \cdot \frac{1}{6} + 1 \cdot \frac{1}{6} + \cdots + 3 \cdot \frac{1}{6} = 1.5$
- But: $g(\mathbb{E}\left[X \right]) = 1$!

Flaw of averages



Mean correct, variance ignored.

$$\mathbb{E}[g(X)] \neq g(\mathbb{E}[X])$$

Be careful with expected values!

Savage, 2009.

Conditional expectation

Definition

Let X, Y be random variables. The *conditional expection* of Y given X is the random variable $\psi(X)$ where

$$\psi(x) = \mathbb{E}[\, Y \mid X = x\,] = \sum_{y} y\, f_{Y|X}(y \mid x),$$

where $f_{Y|X}(y \mid x) = \mathbb{P}(\, Y = y \mid X = x\,)$.

Example

- *Indicator variable*: $I_A(\omega) = \begin{cases} 1 & \text{if } \omega \in A \\ 0 & \text{otherwise} \end{cases}$
- Fair die; set $X = I_{\text{even}} = I_{\{\,2,4,6\,\}}$; Y is identity
- $\mathbb{E}[\, Y \mid X = 1\,] = 1 \cdot 0 + 2 \cdot \frac{1}{3} + 3 \cdot 0 + 4 \cdot \frac{1}{3} + 5 \cdot 0 + 6 \cdot \frac{1}{3} = 4$
- $\mathbb{E}[\, Y \mid X = 0\,] = 1 \cdot \frac{1}{3} + 2 \cdot 0 + 3 \cdot \frac{1}{3} + 4 \cdot 0 + 5 \cdot \frac{1}{3} + 6 \cdot 0 = 3$
- $\mathbb{E}[\, Y \mid X\,](\omega) = \begin{cases} 4 & \text{if } X(\omega) = 1 \\ 3 & \text{if } X(\omega) = 0 \end{cases}$

Important properties

We use shortcut notation $\mathbb{P}(X)$ for $\mathbb{P}(X = x)$.

Theorem

$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B)$$

$$\mathbb{P}(A^{\mathrm{c}}) = 1 - \mathbb{P}(A)$$

$$\textit{If } B \supseteq A, \quad \mathbb{P}(B) = \mathbb{P}(A) + \mathbb{P}(B \setminus A) \geq \mathbb{P}(A)$$

$$\mathbb{P}(X) = \sum_{y} \mathbb{P}(X, Y = y) \qquad \textit{(sum rule)}$$

$$\mathbb{P}(X, Y) = \mathbb{P}(Y \mid X)\mathbb{P}(X) \qquad \textit{(product rule)}$$

$$\mathbb{P}(A \mid B) = \frac{\mathbb{P}(B \mid A)\mathbb{P}(A)}{\mathbb{P}(B)} \qquad \textit{(Bayes theorem)}$$

$$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b \qquad \textit{(linearity of expectation)}$$

$$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

$$\mathbb{E}[\mathbb{E}[X \mid Y]] = \mathbb{E}[X] \qquad \textit{(law of total expectation)}$$

Outline

1. Refresher: Finite Probability

2. Probabilistic Databases

3. Probabilistic Representation Systems
    - pc-tables
    - Tuple-independent databases
    - Other common representation systems

4. Summary

Amateur bird watching

- Bird watcher's observations

Sightings

Name	Bird	Species	
Mary	Bird-1	Finch: 0.8 ‖ Toucan: 0.2	t_1
Susan	Bird-2	Nightingale: 0.65 ‖ Toucan: 0.35	t_2
Paul	Bird-3	Humming bird: 0.55 ‖ Toucan: 0.45	t_3

- Which species may have been sighted? → CWA, possible tuples

ObservedSpecies

Species		
Finch	0.80	$(t_1, 1)$
Toucan	0.71	$(t_1, 2) \vee (t_2, 2) \vee (t_3, 2)$
Nightingale	0.65	$(t_2, 1)$
Humming bird	0.55	$(t_3, 1)$

Probabilistic databases quantify uncertainty.

What do probabilities mean?

- Multiple interpretations of probability
- Frequentist interpretation
  - Probability of an event = relative frequency when repeated often
  - Coin, n trials, n_{H} observed heads

$$\lim_{n \to \infty} \frac{n_{\mathrm{H}}}{n} = \frac{1}{2} \implies \mathbb{P}(\mathrm{H}) = \frac{1}{2}$$

- Bayesian interpretation
  - Probability of an event = degree of belief that event holds
  - Reasoning with "background knowledge" and "data"
  - Prior belief + model + data → posterior belief
    - Model parameter: θ = true "probability" of heads
    - Prior belief: $\mathbb{P}(\theta)$
    - Likelihood (model): $\mathbb{P}(n_{\mathrm{H}}, n \mid \theta)$
    - Bayes theorem: $\mathbb{P}(\theta \mid n_{\mathrm{H}}, n) \propto \mathbb{P}(n_{\mathrm{H}}, n \mid \theta)\,\mathbb{P}(\theta)$
    - Posterior belief: $\mathbb{P}(\theta \mid n_{\mathrm{H}}, n)$

But… what do probabilities really mean? And where do they come from?

- Answers differ from application to application, e.g.,
  - Information extraction → from probabilistic models
  - Data integration → from background knowledge & expert feedback
  - Moving objects → from particle filters
  - Predictive analytics → from statistical models
  - Scientific data → from measurement uncertainty
  - Fill in missing data → from data mining
  - Online applications → from user feedback
- Semantics sometimes precise, sometimes less so
- Often: Convert model scores to $[0, 1]$
  - Larger value → higher confidence
  - Carries over to queries: higher probability of an answer → more credible
  - Ranking often more informative than precise probabilities

Many applications can benefit from a platform that manages probabilistic data.

Probabilistic database

Example

Sightings

Name	Bird	Species
Mary	Bird-1	Finch: 0.8 ‖ Toucan: 0.2
Susan	Bird-2	Nightingale: 0.65 ‖ Toucan: 0.35
Paul	Bird-3	Humming bird: 0.55 ‖ Toucan: 0.45

Possible worlds:

N	B	S
M	1	F
S	2	N
P	3	H

0.286

N	B	S
M	1	F
S	2	N
P	3	T

0.234

N	B	S
M	1	F
S	2	T
P	3	H

0.154

N	B	S
M	1	F
S	2	T
P	3	T

0.126

N	B	S
M	1	T
S	2	N
P	3	H

0.0715

N	B	S
M	1	T
S	2	N
P	3	T

0.0585

N	B	S
M	1	T
S	2	T
P	3	H

0.0385

N	B	S
M	1	T
S	2	T
P	3	T

0.0315

Definition

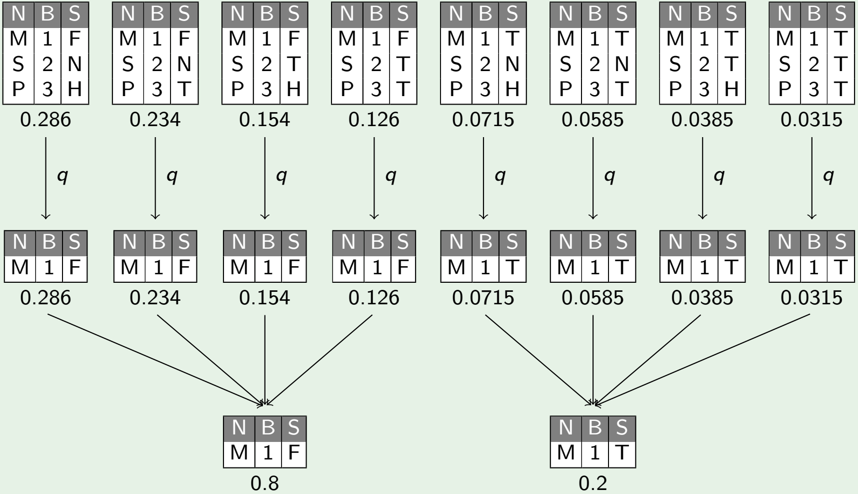
A (finite) *probabilistic database* (*p-database*, *PDB*) is a probability space $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ over a (finite) incomplete database $\mathcal{I}$ in which w.l.o.g. $\mathbb{P}(I) > 0$ for all $I \in \mathcal{I}$.

A PDB associates a nonzero probability to each *possible world* $I \in \mathcal{I}$.

Possible answer set semantics (example)

Example

What did Mary see? $\rightarrow$ $q(R) = \sigma_{\text{Name='Mary'}}(R)$



N	B	S
M	1	F
S	2	N
P	3	H

0.286

N	B	S
M	1	F
S	2	N
P	3	T

0.234

N	B	S
M	1	F
S	2	T
P	3	H

0.154

N	B	S
M	1	F
S	2	T
P	3	T

0.126

N	B	S
M	1	T
S	2	N
P	3	H

0.0715

N	B	S
M	1	T
S	2	N
P	3	T

0.0585

N	B	S
M	1	T
S	2	T
P	3	H

0.0385

N	B	S
M	1	T
S	2	T
P	3	T

0.0315

N	B	S
M	1	F

0.286

N	B	S
M	1	F

0.234

N	B	S
M	1	F

0.154

N	B	S
M	1	F

0.126

N	B	S
M	1	T

0.0715

N	B	S
M	1	T

0.0585

N	B	S
M	1	T

0.0385

N	B	S
M	1	T

0.0315

N	B	S
M	1	F

0.8

N	B	S
M	1	T

0.2

Possible answer set semantics

Definition

The *possible answer set* to a query q on a probabilistic database $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ is the probability space $\mathcal{D}_q = (q(\mathcal{I}), \mathbb{P}_q)$, where $q(\mathcal{I})$ is the possible answer set to q on $\mathcal{I}$, and

$$\mathbb{P}_q\,(\,J\,) = \mathbb{P}\,(\,q(I) = J\,) = \mathbb{P}\,(\,\{\,I \in \mathcal{I} : q(I) = J\,\}\,) = \sum_{I \in \mathcal{I} : q(I) = J} \mathbb{P}\,(\,I\,)\,.$$

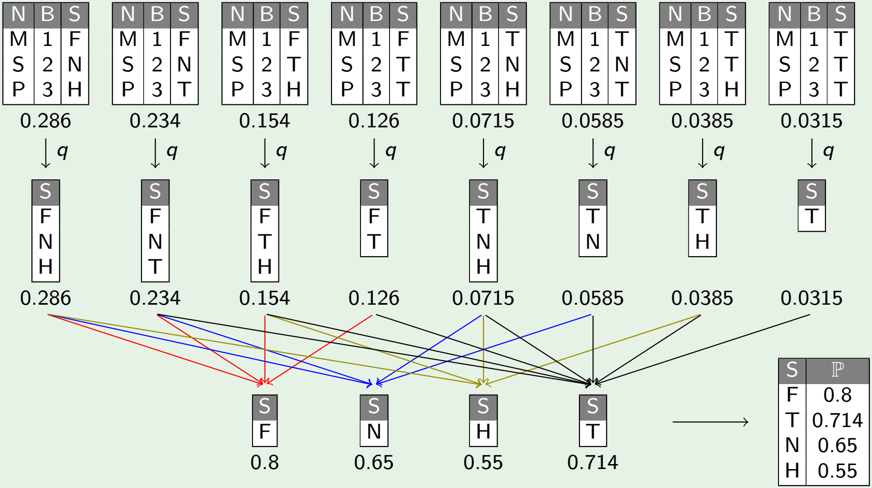
We refer to $\mathcal{D}_q$ as the *image* of $\mathcal{D}$ under q.

- Cf. definition for incomplete databases
- $|q(\mathcal{I})| \leq |\mathcal{I}|$ since each instance of $\mathcal{I}$ gives precisely one result $q(I)$

Possible tuple semantics (example)

Example

Which species have been sighted? $\rightarrow q(R) = \pi_{\text{Species}}(R)$



N	B	S
M	1	F
S	2	N
P	3	H

0.286

N	B	S
M	1	F
S	2	N
P	3	T

0.234

N	B	S
M	1	F
S	2	T
P	3	H

0.154

N	B	S
M	1	F
S	2	T
P	3	T

0.126

N	B	S
M	1	T
S	2	N
P	3	H

0.0715

N	B	S
M	1	T
S	2	N
P	3	T

0.0585

N	B	S
M	1	T
S	2	T
P	3	H

0.0385

N	B	S
M	1	T
S	2	T
P	3	T

0.0315

$\downarrow q$

S
F
N
H

0.286

S
F
N
T

0.234

S
F
T
H

0.154

S
F
T

0.126

S
T
N
H

0.0715

S
T
N

0.0585

S
T
H

0.0385

S
T

0.0315

S
F

0.8

S
N

0.65

S
H

0.55

S
T

0.714

$\longrightarrow$

S	$\mathbb{P}$
F	0.8
T	0.714
N	0.65
H	0.55

Possible tuple semantics

Definition

Let $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ be a probabilistic database. A tuple t is a *possible answer* to a query q if there exists a possible world $I \in \mathcal{I}$ such that $t \in q(I)$. The *marginal probability* of t is given by

$$\mathbb{P}(\, t \in q(I) \,) = \sum_{I \in \mathcal{I} : t \in q(I)} \mathbb{P}(\, I \,).$$

- A tuple t is a *certain answer* if $\mathbb{P}(\, t \in q(I) \,) = 1$;
  equivalently, $(\forall I \in \mathcal{I})\, t \in q(I)$
  $\rightarrow$ Certain answer tuple semantics as before (q-information).
  $\rightarrow$ Weak representation results carry over.
- Possible tuple semantics is the main focus of probabilistic databases

Outline

1. Refresher: Finite Probability

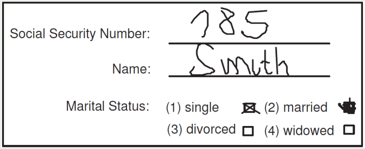
2. Probabilistic Databases

3. Probabilistic Representation Systems
   - pc-tables
   - Tuple-independent databases
   - Other common representation systems

4. Summary

Motivating example

Example



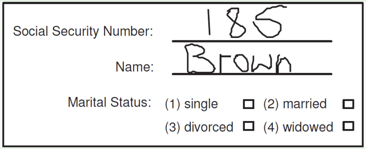
Social Security Number: 185

Name: Smith

Marital Status: (1) single ☒ (2) married ■

(3) divorced ☐ (4) widowed ☐

Form 1



Social Security Number: 185

Name: Brown

Marital Status: (1) single ☐ (2) married ☐

(3) divorced ☐ (4) widowed ☐

Form 2

Ambiguity:

- Is Smith single or married?
- What is the martial status of Brown?
- What is Smith's social security number: 185 or 785?
- What is Brown's social security number: 185 or 186?

Probabilistic database:

- Here: $2 \cdot 4 \cdot 2 \cdot 2 = 32$ possible readings → can easily store all of them
- 200M people, 50 questions, 1 in 10000 ambiguous (2 options) → 2^{10^6} possible readings
- Each reading is a table with 50 columns and 200M rows!

Probabilistic representation system

Finiteness assumption: Throughout our entire treatment of PDBs.

Definition

A *probabilistic representation system* consists of a set $\mathscr{T}$ of tables and a function Mod that associates to each table $T \in \mathscr{T}$ a probabilistic database $\mathrm{Mod}(T)$.

Definition

A probabilistic representation system is *complete* if it can represent any probabilistic database.

Definition

Let $(\mathscr{T}, \mathrm{Mod})$ be a probabilistic representation system and $\mathscr{L}$ be a query language. The probabilistic representation system obtained by *closing $\mathscr{T}$ under $\mathscr{L}$* is the set of tables $\{\, (T, q) \mid T \in \mathscr{T}, q \in \mathscr{L} \,\}$ and function $\mathrm{Mod}(T, q) = q(\mathrm{Mod}(T))$.

Outline

1. Refresher: Finite Probability

2. Probabilistic Databases

3. Probabilistic Representation Systems
    - pc-tables
    - Tuple-independent databases
    - Other common representation systems

4. Summary

pc-table (example)

Example

FID	SSN	Name	
1	185	Smith	$X = 1$
1	785	Smith	$X \neq 1$
2	185	Brown	$Y = 1 \land X \neq 1$
2	186	Brown	$Y \neq 1 \lor X = 1$

$\underline{V}$	D	$\mathbb{P}$
X	1	0.2
X	2	0.8
Y	1	0.3
Y	2	0.7

FID	SSN	Name
1	185	Smith
2	186	Brown

$\{X \mapsto 1, Y \mapsto 1\}$

$\{X \mapsto 1, Y \mapsto 2\}$

$0.2 \cdot 0.3 + 0.2 \cdot 0.7$

$= 0.2$

FID	SSN	Name
1	785	Smith
2	185	Brown

$\{X \mapsto 2, Y \mapsto 1\}$

$0.8 \cdot 0.3$

$= 0.24$

FID	SSN	Name
1	785	Smith
2	186	Brown

$\{X \mapsto 2, Y \mapsto 2\}$

$0.8 \cdot 0.7$

$= 0.56$

pc-tables

Definition

A *probabilistic c-table* (pc-table) is pair $(T, \mathbb{P})$, where T ia a c-table and $\mathbb{P}$ a probability distribution over the set of assignments Θ of $\mathrm{Var}(T)$ such that all variables are independent.

$$\mathrm{Mod}(T) = \{\, \theta(T) : \theta \in \Theta \,\}$$

$$\mathbb{P}(\, I \,) = \sum_{\theta \in \Theta : \theta(T) = I} \mathbb{P}(\, \theta \,)$$

- Variables are independent
  $\rightarrow$ need only specify probabilities of form $\mathbb{P}(\, X = a \,)$
- $\mathbb{P}$ can be stored in a standard relation storing (variable, value, probability)-triples

Completeness of pc-tables

Theorem

pc-tables are a complete representation system.

Proof.

Let $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ be a probabilistic database with $\mathcal{I} = \{ I^1, \ldots, I^n \}$ and $I^k = \{ t_{k1}, \ldots, t_{kn_k} \}$. Let X be a random variable with domain $\{ 1, \ldots, n \}$. Set $\mathbb{P}(X = k) = \mathbb{P}\left(\mathcal{I}^k \right)$ and use the c-table:

$\alpha(\mathcal{I})$	
t_{11}	$X = 1$
$\vdots$	
t_{1n_1}	$X = 1$
t_{21}	$X = 2$
$\vdots$	
t_{2n_2}	$X = 2$
t_{31}	$X = 3$
$\vdots$	

.

Completeness of pc-tables (example)

Example

I^1

FID	SSN	Name
1	185	Smith
2	186	Brown

0.2

I^2

FID	SSN	Name
1	785	Smith
2	185	Brown

0.24

I^3

FID	SSN	Name
1	785	Smith
2	186	Brown

0.56

FID	SSN	Name	
1	185	Smith	$X = 1$
2	186	Brown	$X = 1$
1	785	Smith	$X = 2$
2	185	Brown	$X = 2$
1	785	Smith	$X = 3$
2	186	Brown	$X = 3$

$\underline{V}$	D	$\mathbb{P}$
X	1	0.2
X	2	0.24
X	3	0.56

pc-tables are strong

Theorem

pc-tables are strong under $\mathcal{RA}$.

Proof.

Given a pc-table $(T, \mathbb{P})$ and a query q, the resulting pc-table is given by $(\bar{q}(T), \mathbb{P})$, where $\bar{q}$ is the c-table algebra query corresponding to q. ☐

Example

R

FID	SSN	Name	
1	185	Smith	$X = 1$
1	785	Smith	$X \neq 1$
2	185	Brown	$Y = 1 \wedge X \neq 1$
2	186	Brown	$Y \neq 1 \vee X = 1$

$\underline{V}$	D	$\mathbb{P}$
X	1	0.2
X	2	0.8
Y	1	0.3
Y	2	0.7

$\pi_{\text{SSN}}(R)$

SSN	
185	$X = 1 \vee (Y = 1 \wedge X \neq 1)$
785	$X \neq 1$
186	$Y \neq 1 \vee X = 1$

Outline

1. Refresher: Finite Probability
2. Probabilistic Databases
3. Probabilistic Representation Systems
   - pc-tables
   - Tuple-independent databases
   - Other common representation systems
4. Summary

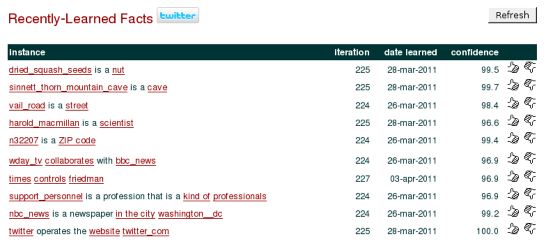
Tuple-independent databases (p?-tables)

Definition

In a *tuple-independent probabilistic database* T, each tuple $t \in T$ is marked with a probability $p_t > 0$. We have $\mathrm{Mod}(T) = (\mathcal{I}, \mathbb{P})$ where $\mathcal{I} = \{\, I \subseteq T : \mathbb{P}(\, I\,) > 0 \,\}$ and

$$\mathbb{P}(\, I\,) = \left(\prod_{t \in I} p_t \right) \left(\prod_{t \notin I} (1 - p_t) \right).$$

Example (Nell)



Recently-Learned Facts

instance	iteration	date learned	confidence
dried_squash_seeds is a nut	225	28-mar-2011	99.5
sinnett_thorn_mountain_cave is a cave	225	28-mar-2011	99.7
vail_road is a street	224	26-mar-2011	98.4
harold_macmillan is a scientist	225	28-mar-2011	96.6
n32207 is a ZIP code	224	26-mar-2011	99.4
wday_tv collaborates with bbc_news	224	26-mar-2011	96.9
times controls friedman	227	03-apr-2011	96.9
support_personnel is a profession that is a kind of professionals	224	26-mar-2011	96.9
nbc_news is a newspaper in the city washington__dc	224	26-mar-2011	99.2
twitter operates the website twitter_com	225	28-mar-2011	100.0

Completeness

Theorem

Tuple-independent databases are not complete.

Proof.

They can only represent databases in which all tuples are independent events. E.g., they cannot represent

$$\left\{ \underset{0.5}{\boxed{a}}\,,\ \underset{0.5}{\boxed{b}} \right\} \qquad \text{or} \qquad \left\{ \underset{0.1}{\boxed{\begin{array}{c}\\ \end{array}}}\,,\ \underset{0.1}{\boxed{\begin{array}{c}a\\ \end{array}}}\,,\ \underset{0.1}{\boxed{\begin{array}{c}b\\ \end{array}}}\,,\ \underset{0.7}{\boxed{\begin{array}{c}a\\ b\end{array}}} \right\}.$$

$\square$

Theorem

The closure of tuple-independent databases under positive $\mathcal{RA}$ is not complete.

Closure under $\mathcal{RA}$

Theorem

The closure of tuple-independent databases under $\mathcal{RA}$ is complete.

Proof.

Let $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ be a probabilistic database with $\mathcal{I} = \{ I^1, \dots, I^n \}$. To obtain a tuple-independent database, use n certain EDB predicates $R^1, \dots, R^n$ with $I(R^k) = I^k$ and one tuple-independent table W that contains tuples $\{ 1, \dots, n \}$ with $p_k = \mathbb{P}\left(I^k \mid \{ I_1, \dots, I_{k-1} \}^{\mathrm{c}} \right)$. Write a query that selects relation R^k iff $\operatorname{argmin}_{t:W(t)} = k$:

$$\begin{array}{ll}
R(\mathbf{x}) \leftarrow W(1), R^1(\mathbf{x}) & p_1 = \mathbb{P}\left(I^1 \right) \\
R(\mathbf{x}) \leftarrow \neg W(1), W(2), R^2(\mathbf{x}) & p_2 = \mathbb{P}\left(I^2 \mid \{ I^1 \}^{\mathrm{c}} \right) \\
R(\mathbf{x}) \leftarrow \neg W(1), \neg W(2), W(3), R^3(\mathbf{x}) & p_3 = \mathbb{P}\left(I^3 \mid \{ I^1, I^2 \}^{\mathrm{c}} \right) \\
\vdots & \vdots \\
R(\mathbf{x}) \leftarrow \neg W(1), \dots, \neg W(n-1), W(n), R^n(\mathbf{x}) & p_n = 1
\end{array}$$

□

Closure under $\mathcal{RA}$ (example)

Example

$I^1 = I(R^1)$

FID	SSN	Name
1	185	Smith
2	186	Brown

0.2

$I^2 = I(R^2)$

FID	SSN	Name
1	785	Smith
2	185	Brown

0.24

$I^3 = I(R^3)$

FID	SSN	Name
1	785	Smith
2	186	Brown

0.56

$$\begin{aligned} R(f,s,n) &\leftarrow W(1), R^1(f,s,n) & p_1 &= 0.2 \\ R(f,s,n) &\leftarrow \neg W(1), W(2), R^2(f,s,n) & p_2 &= 0.24/(1-0.2) \\ R(f,s,n) &\leftarrow \neg W(1), \neg W(2), W(3), R^3(f,s,n) & p_3 &= 0.56/(1-0.2-0.24) \end{aligned}$$

W

World	$\mathbb{P}$
1	0.2
2	0.3
3	1

$\mathbb{P}(\operatorname{argmin}_{t:W(t)} = 1) = 0.2$

$\mathbb{P}(\operatorname{argmin}_{t:W(t)} = 2) = 0.3 \cdot (1-0.2) = 0.24$

$\mathbb{P}(\operatorname{argmin}_{t:W(t)} = 3) = 1 \cdot (1-0.2) \cdot (1-0.3) = 0.56$

Probabilistic database design

- Database normalization → Minimize redundancy/correlations
- Tuple-independent databases are good building blocks
  - No correlations between tuples
  - No constraints
  - Database normalization can be applied
- Decompose complex databases into tuple-independent databases

Example (Nell)

- nellExtraction: extracted relations
  (tuple probability = belief that extracted tuple is correct)
- nellSource: source of extraction
  (tuple probability = belief that source is correct)
- Correlation via views

ProducesProduct(x, y) ← nellExtraction(x, 'ProducesProduct', y, s), nellSource(s)

Tuple-independent databases can be stored in standard relations.

Outline

1. Refresher: Finite Probability

2. Probabilistic Databases

3. Probabilistic Representation Systems
    - pc-tables
    - Tuple-independent databases
    - Other common representation systems

4. Summary

BID tables

- Relations are partitioned into blocks
- Events within a block a disjoint; events across blocks are independent
  → Block-independent-disjoint database
- Blocks are identified by key attributes

Example

FID	SSN	Name	
1	185	Smith	$X = 1$
1	785	Smith	$X = 2$
2	175	Brown	$Y = 1$
2	186	Brown	$Y = 2$

V	D	$\mathbb{P}$
X	1	0.8
X	2	0.2
Y	1	0.5
Y	2	0.5

→

FID	SSN	Name	$\mathbb{P}$
1	185	Smith	0.8
1	785	Smith	0.2
2	175	Brown	0.5
2	186	Brown	0.5

Theorem

BID-tables extended with PJR queries are a complete representation system.

U-tables (MayBMS)

- Goal: completeness + natural representation in RDBMS
- Restrict pc-table conditions to forms $X_1 = a_1 \wedge \ldots \wedge X_k = a_k$
- Conditions → *U*-tables (usually: one per set of correlated attributes)
- Distribution over assignments → BID-table (*world table*)

Example

R

FID	SSN	Name	
1	185	Smith	$X = 1$
1	785	Smith	$X = 2$
2	185	Brown	$Y = 1 \wedge X = 2$
2	186	Brown	$Y = 2$
2	186	Brown	$X = 1$

W

$\underline{V}$	D	$\mathbb{P}$
X	1	0.2
X	2	0.8
Y	1	0.3
Y	2	0.7

T

V_1	D_1	V_2	D_2	FID	SSN	Name
X	1	X	1	1	185	Smith
X	2	X	2	1	785	Smith
Y	1	X	2	2	185	Brown
Y	2	Y	2	2	186	Brown
X	1	X	1	2	186	Brown

Reconstruction via joins: $R(f, s, n) \leftarrow T(v_1, d_1, v_2, d_2, f, s, n), W(v_1, d_1), W(v_2, d_2)$

Theorem

U-databases are complete. They can compute/represent results of nr-datalog queries conveniently (i.e., in polynomial time and space).

Or-set tables

Example

Probabilistic or-set tables (= probabilistic finite-domain Codd tables):

Sightings

Name	Bird	Species
Mary	Bird-1	Finch: 0.8 ‖ Toucan: 0.2
Susan	Bird-2	Nightingale: 0.65 ‖ Toucan: 0.35
Paul	Bird-3	Humming bird: 0.55 ‖ Toucan: 0.45

Probabilistic ?-or-set tables (Trio):

Sightings

Name	Bird	Species	
Mary	Bird-1	Finch: 0.8 ‖ Toucan: 0.2	
Susan	Bird-2	Nightingale: 0.65 ‖ Toucan: 0.10	?
Paul	Bird-3	Humming bird	0.55

Outline

1. Refresher: Finite Probability
2. Probabilistic Databases
3. Probabilistic Representation Systems
    - pc-tables
    - Tuple-independent databases
    - Other common representation systems
4. Summary

Lessons learned

- Probabilistic databases quantify uncertainty
- Probabilistic database $=$ incomplete database $+$ probability distribution
- Many notions and results from incomplete databases carry over
- Queries can be analyzed in terms of
  1. Possible answer sets
  2. Certain answer tuples (same as incomplete databases)
  3. Possible answer tuples (main focus of PDBs)
- pc-tables $\rightarrow$ complete, strong under $\mathcal{RA}$
- Tuple-independent tables $\rightarrow$ complete when closed under $\mathcal{RA}$ (Good probabilistic database design)
- BID-tables $\rightarrow$ complete when closed under PJR queries
- U-databases $\rightarrow$ complete, handle positive $\mathcal{RA}$ well, easy to represent in an RDBMS

Suggested reading

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 2)
  Springer, 2009

- Dan Sucio, Dan Olteanu, Christopher Ré, Christoph Koch
  Probabilistic Databases (Chapter 2)
  Not yet published (But you'll get copies!)

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 5 → Trio)
  Springer, 2009

- Charu C. Aggarwal (Ed.)
  Managing and Mining Uncertain Data (Chapter 6 → MayBMS)
  Springer, 2009

Scalable Uncertainty Management

05 – Query Evaluation in Probabilistic Databases

Rainer Gemulla

Jun 1, 2012

Overview

In this lecture

- Primer: relational calculus
- Understand complexity of query evaluation
- How to determine whether a query is "easy" or "hard"
- How to efficently evaluate easy queries
  → extensional query evaluation
- How to evaluate hard queries
  → intensional query evaluation
- How to approximately evaluate queries

Not in this lecture

- Possible answer set semantics
- Most representation systems but tuple-independent databases

Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary

Relational calculus ($\mathcal{RC}$)

- Similar to nr-datalog$^\neg$, but uses a *single query expression*
- Suitable to reason over query expressions as a whole
- Queries are built from logical connectives

  $$q ::= u = v \mid R(\mathbf{x}) \mid \exists x.q_1 \mid q_1 \land q_2 \mid q_1 \lor q_2 \mid \neg q_1,$$

  where u, v are either variables of constants
- Extended $\mathcal{RC}$: adds arithmetic expressions
- Free variables in q are called *head variables*

Example

$\mathcal{RA}$ query:

$$\pi_{\text{HotelNo},\text{Name},\text{City}}(\text{Hotel} \bowtie \sigma_{\text{Price}>500 \lor \text{Type}=\text{'suite'}}(\text{Room}))$$

$\mathcal{RC}$ query and its abbreviation:

$$q(h, n, c) \leftarrow \exists r.\exists t.\exists p.\text{Hotel}(h, n, c) \land \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})$$

$$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \land \text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})$$

Alternative $\mathcal{RC}$ query:

$$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \land \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})$$

Boolean query

Definition

A *Boolean query* is an $\mathcal{RC}$ query with no head variables.

- Asks whether the query result is empty
- Can be obtained from $\mathcal{RC}$-query by
  1. Adding existential quantifiers for the head variables
  2. Replacing head variables by constants (potential results)

Example

$\mathcal{RC}$-query:

$q(h, n, c) \leftarrow \text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$

Boolean $\mathcal{RC}$-query ("Is there an answer?"):

$q \leftarrow \exists h.\exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$

Another Boolean $\mathcal{RC}$-query ("Is (H1,Hilton,Paris) an answer?"):

$q \leftarrow \text{Hotel}(\text{'H1'}, \text{'Hilton'}, \text{'Paris'}) \wedge \exists r.\exists t.\exists p.\text{Room}(r, \text{'H1'}, t, p) \wedge (p > 500 \vee t = \text{'suite'})$

Query semantics

- *Active domain*: set of all constants occurring in the database
- *Active domain semantics*
  1. Every quantifier $\exists x$ ranges over active domain
  2. Query answers are restricted to active domain
- *Domain-independent query*: query result independent of domain (cf. safe queries for datalog)
- Domain-independent queries and query evaluation under active domain semantics are equally expressive

Example

- Active domain of R: $\{1, 2\}$
- Domain-independent query

$$q(x) \leftarrow \exists y . R(x, y)$$

- Domain-dependent queries

$$q(x) \leftarrow \exists y . \exists z . R(y, z)$$

$$q(x) \leftarrow \exists y . \neg R(x, y)$$

R

1	1
1	2

Relationships between query languages

Theorem

Each row of languages in the following table is equally expressive (we consider only safe rules with a single output relation for nr-datalog$^\neg$ and domain-independent rules for $\mathcal{RC}$).

Relational algebra	nr-datalog$^\neg$	Relational calculus
SPJR	No repeated head predicates, no negation	$\exists, \wedge$ *(conjunctive queries: $\mathcal{CQ}$)*
SPJRU *(positive $\mathcal{RA}$)*	No negation *(nr-datalog)*	$\exists, \wedge, \vee$ *(unions of $\mathcal{CQ}$: $\mathcal{UCQ}$)*
SPJRUD *($\mathcal{RA}$)*	– *(nr-datalog$^\neg$)*	$\exists, \wedge, \vee, \neg$ *($\mathcal{RC}$)*

Outline

The query evaluation problem

- Database systems are expected to *scale* to large datasets and *parallelize* to a large number of processors
  → Same behavior is expected from probabilistic databases
- We consider the possible tuple semantics, i.e., a query answer is an ordered set of answer-probability pairs

$$\{\,(t_1, p_1), (t_2, p_2), \dots\,\} \quad \text{with } p_1 \geq p_2 \geq \dots$$

Definition (Query evaluation problem)

Fix a query q. Given a (representation of a) probabilistic database $\mathcal{D}$ and a possible answer tuple t, compute its marginal probability $\mathbb{P}\,(\,t \in q(\mathcal{D})\,)$.

Questions of interest

- Characterize which queries are hard
  → Understand what makes query evaluation hard
- Given a query, determine whether it is hard
  → Guide query processing
- Given an easy query, solve the QEP
  → Be efficient whenever possible
- Given a hard query, solve the QEP (exactly or approximately)
  → Don't give up on hard queries

Query evaluation on deterministic databases

Definition

The *data complexity* of a query q is the complexity of evaluating it as a function of the size of the input database. A query is *tractable* if its data complexity is in polynomial time; otherwise, it is *intractable*.

Example

- Fix a relation schema R and consider an instance I with n tuples
- $q(R) = R \rightarrow O(n)$
- $q(R) = \sigma_E(R) \rightarrow O(n)$
- $q(R) = \pi_U(R) \rightarrow O(n^2)$; can be tightened

Theorem

On deterministic databases, the data complexity of every $\mathcal{RA}$ *query is in polynomial time. Thus query evaluation is always tractable.*

Query evaluation on probabilistic databases

Corollary

Query evaluation over probabilistic databases is tractable.

Proof.

Fix query q. Given a probabilistic database $\mathcal{D} = (\mathcal{I}, \mathbb{P})$ with $\mathcal{I} = \{ I^1, \dots, I^n \}$, perform the following steps:

1. Compute $q(I^k)$ for $1 \leq k \leq n \rightarrow$ polynomial time
2. For each tuple $t \in q(I^k)$ for some k, compute

$$\mathbb{P}(\, t \in q(\mathcal{D})\,) = \sum_{k : t \in q(I^k)} \mathbb{P}(\, I^k \,)$$

$\rightarrow$ polynomially many tuples, polynomial time per tuple □

This result is treacherous: It talks about probabilistic databases but not about *probabilistic representation systems*!

Lineage trees and the query evaluation problem

Example

$$q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \land \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \land (p > 500 \lor t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	X_1
R2	Single	H1	\$600	X_2
R3	Double	H1	\$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	$X_4 \land (X_1 \lor X_2)$

Theorem

Fix a $\mathcal{RA}$ query q. Given a boolean pc-table $(T, \mathbb{P})$, we can compute the lineage Φ_t of each possible output tuple t in polynomial time, where Φ_t is a propositional formula. We have

$$\mathbb{P}(\, t \in q(T)\,) = \mathbb{P}(\, \Phi_t\,)\,.$$

How can we compute Φ_t?

A naive approach

Let $\omega(\Phi)$ be the set of assignments over $\mathrm{Var}(T)$ that make Φ true. Then apply $\mathbb{P}(\Phi) = \sum_{\theta \in \omega(\Phi)} \mathbb{P}(\theta)$.

Exponential time: n variables $\rightarrow 2^n$ assignments to check!

Definition (Model counting problem)

Given a propositional formula Φ, count the number of satisfying assignments $\#\Phi = |\omega(\Phi)|$.

Definition (Probability computation problem)

Given a propositional formula Φ and a probability $\mathbb{P}(X) \in [0, 1]$ for each variable X, compute the probability $\mathbb{P}(\Phi) = \sum_{\theta \in \omega(\Phi)} \mathbb{P}(\theta)$.

Model counting is a special case of probability computation

- Suppose we have an algorithm to compute $\mathbb{P}(\Phi)$
- We can use the algorithm to compute $\#\Phi$
- Define $\mathbb{P}(X) = \frac{1}{2}$ for every variable X
- $\mathbb{P}(\theta) = 1/2^n$ for every assignment (n = number of variables)
- $\#\Phi = \mathbb{P}(\Phi) \cdot 2^n$

Example

- $\Phi = (X_1 \vee X_2) \wedge X_4$; $n = 3$
- $\#\Phi = 3$
- $\mathbb{P}(\Phi) = \frac{3}{8} = \frac{\#\Phi}{2^n}$

X_1	X_2	X_4	Φ_θ	$\mathbb{P}(\theta)$
0	0	0	FALSE	1/8
0	0	1	FALSE	1/8
0	1	0	FALSE	1/8
0	1	1	TRUE	1/8
1	0	0	FALSE	1/8
1	0	1	TRUE	1/8
1	1	0	FALSE	1/8
1	1	1	TRUE	1/8

The complexity class #P

Definition

The complexity class *#P* consists of all function problems of the following type: Given a polynomial-time, non-deterministic Turing machine, compute the number of accepting computations.

Theorem (Valiant, 1979)

Model counting (#SAT) is complete for #P.

- NP asks whether there exists at least one accepting computation
- #P counts the number of accepting computations
- SAT is NP-complete
- #SAT is #P-complete

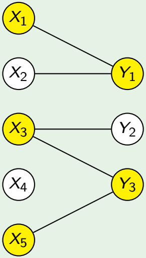
Directly implies that probability computation is hard for #P!

A graph problem

Definition (Bipartite vertex cover)

Given a bipartite graph (V, E), compute $|\{ S \subseteq V : (u, w) \in E \rightarrow u \in S \vee w \in S \}|$.

Example



80 possible ways

Theorem (Provan and Ball, 1983)

Bipartite vertex cover is #P-complete.

#PP2DNF and #PP2CNF

Definition

Let $X_1, X_2, \ldots$ and $Y_1, Y_2, \ldots$ be two disjoint sets of Boolean variables.

- A positive, partitioned 2-CNF propositional formula (PP2CNF) has form $\Psi = \bigwedge_{(i,j)\in E}(X_i \vee Y_j)$.
- A positive, partitioned 2-DNF propositional formula (PP2DNF) has form $\Phi = \bigvee_{(i,j)\in E} X_i Y_j$.

Theorem

#PP2CNF and #PP2DNF are #P-complete.

Proof.

#PP2CNF reduces to bipartite vertex cover. For any given E, we have $\#\Phi = 2^n - \#\Psi$, where n is the total number of variables. ☐

Note: 2-CNF is in P.

A hard query

Theorem

The query evaluation problem of the $\mathcal{CQ}$ query H_0 given by

$$H_0 \leftarrow R(x) \wedge S(x, y) \wedge T(y)$$

on tuple-independent databases is hard for $\#P$.

Proof.

Given a PP2DNF formula $\Phi = \bigvee_{(i,j)\in E} X_i Y_j$, where $E = \{ (X_{e_1}, Y_{e_1}), (X_{e_2}, Y_{e_2}), \dots \}$, construct the tuple-independent DB:

R

X	
X_1	$1/2$
X_2	$1/2$
$\vdots$	

S

X	Y	
X_{e_1}	Y_{e_1}	1
X_{e_2}	Y_{e_2}	1
$\vdots$	$\vdots$	$\vdots$

T

Y	
Y_1	$1/2$
Y_2	$1/2$
$\vdots$	

Then $\#\Phi = 2^n \, \mathbb{P}(H_0)$, where n is the total number of variables. □

More hard queries

Theorem

All of the following $\mathcal{RC}$ queries on tuple-independent databases are #P-hard:

$$
\begin{aligned}
H_0 &\leftarrow R(x) \wedge S(x,y) \wedge T(y) \\
H_1 &\leftarrow [R(x_0) \wedge S(x_0,y_0)] \vee [S(x_1,y_1) \wedge T(y_1)] \\
H_2 &\leftarrow [R(x_0) \wedge S_1(x_0,y_0)] \vee [S_1(x_1,y_1) \wedge S_2(x_1,y_1)] \\
&\qquad \vee [S_2(x_2,y_2) \wedge T(y_2)] \\
&\vdots
\end{aligned}
$$

Queries can be tractable even if they have intractable subqueries!

- $q(x,y) \leftarrow R(x) \wedge S(x,y) \wedge T(y)$ is tractable
- $q \leftarrow H_0 \vee T(y)$ is tractable

Extensional and intensional query evaluation

- We'll say more about data complexity as we go
- Extensional query evaluation
  - Evaluation process guided by query expression q
  - Not always possible
  - When possible, data complexity is in polynomial time
- Extensional plans
  - Extensional query evaluation in the database
  - Only minor modifications to RDBMS necessary
  - Scalability, parallelizability retained
- Intensional query evaluation
  - Evaluation process guided by query lineage
  - Reduces query evaluation to the problem of computing the probability of a propositional formula
  - Works for every query

Outline

Problem statement

- Tuple-independent database
  - Each tuple t annotated with a unique boolean variable X_t
  - We write $\mathbb{P}(\, t \,) = \mathbb{P}(\, X_t \,)$
- Boolean query Q
  - With lineage Φ_Q
  - We write $\mathbb{P}(\, Q \,) = \mathbb{P}(\, \Phi_Q \,)$
- Goal: compute $\mathbb{P}(\, Q \,)$ when Q is tractable
  - Evaluation process guided by query expression q
  - I.e., without first computing lineage!

Example

Birds

Species	$\mathbb{P}$	
Finch	0.80	X_1
Toucan	0.71	X_2
Nightingale	0.65	X_3
Humming bird	0.55	X_4

- $\mathbb{P}(\, \text{Finch} \,) = \mathbb{P}(\, X_1 \,) = 0.8$
- Is there a finch? $Q \leftarrow \text{Birds(Finch)}$
  - $\Phi_Q = X_1$
  - $\mathbb{P}(\, Q \,) = 0.8$
- Is there some bird? $Q \leftarrow \text{Birds}(s)$?
  - $\Phi_Q = X_1 \vee X_2 \vee X_3 \vee X_4$
  - $\mathbb{P}(\, Q \,) \approx 99.1\%$

Overview of extensional query evaluation

- Break the query into “simpler” subqueries
- By applying one of the rules
  1. Independent-join
  2. Independent-union
  3. Independent-project
  4. Negation
  5. Inclusion-exclusion (or Möbius inversion formula)
  6. Attribute ranking
- Each rule application is polynomial in size of database
- Main results for $\mathcal{UCQ}$ queries
  - Completeness: Rules succeed iff query is tractable
  - Dichotomy: Query is #P-hard if rules don’t succeed

Outline

Unifiable atoms

Definition

Two relational atoms L_1 and L_2 are said to be *unifiable* (or *to unify*) if they have a common image. I.e., there exists substitutions such that $L_1[\mathbf{a}_1/\mathbf{x}_1] = L_2[\mathbf{a}_2/\mathbf{x}_2]$, where $\mathbf{x}_1$ are the variables in L_1 and $\mathbf{x}_2$ are the variables in L_2.

Example

Unifiable:

- $R(a)$, $R(a)$ via $[]$, $[]$
- $R(x)$, $R(y)$ via $[a/x]$, $[a/y]$
- $R(a, y)$, $R(x, y)$ via $[b/y]$, $[(a, b)/(x, y)]$
- $R(a, b)$, $R(x, y)$ via $[]$, $[(a, b)/(x, y)]$
- $R(a, y)$, $R(x, b)$ via $[b/y]$, $[a/x]$

Not unifiable:

- $R(a)$, $R(b)$
- $R(a, y)$, $R(b, y)$
- $R(x)$, $S(x)$

Unifiable atoms must use the same relation symbol.

Syntactic independence

Definition

Two queries Q_1 and Q_2 are called *syntactically independent* if no two atoms from Q_1 and Q_2 unify.

Example

Syntactically independent:

- $R(a),\ R(b)$
- $R(a, y),\ R(b, y)$
- $R(x),\ S(x)$
- $R(a, x) \lor S(x),\ R(b, x) \land T(x)$

Not syntactically independent:

- $R(a),\ R(x)$
- $R(x),\ R(y)$
- $R(x),\ S(x) \land \neg R(x)$

Checking for syntactic independence can be done in polynomial time in the size of the queries.

Syntactic independence and probabilistic independence

Proposition

Let $Q_1, Q_2, \ldots, Q_k$ be pairwise syntactically independent. Then $Q_1, \ldots, Q_k$ are independent probabilistic events.

Proof.

The sets $\mathrm{Var}(\Phi_{Q_1}), \ldots, \mathrm{Var}(\Phi_{Q_k})$ are pairwise disjoint, i.e., the lineage formulas do not share any variables. Since all variables are independent (because we have a tuple-independent database), the proposition follows. $\square$

Example

Syntactically independent:

- $R(a)$, $R(b)$
- $R(a, y)$, $R(b, y)$
- $R(x)$, $S(x)$
- $R(a, x) \vee S(x)$, $R(b, x) \wedge T(x)$

Not syntactically independent:

- $R(a)$, $R(x)$
- $R(x)$, $R(y)$
- $R(x)$, $S(x) \wedge \neg R(x)$

Probabilistic independence and syntactic independence

Proposition

Probabilistic independence does not necessarily imply syntactic independence.

Example

- Consider

$$\begin{aligned} Q_1 &\leftarrow R(x,y) \wedge R(x,x) \\ Q_2 &\leftarrow R(a,b) \end{aligned}$$

- If Φ_{Q_1} does not contain $X_{R(a,b)}$, Q_1 and Q_2 are independent
- Otherwise, Φ_{Q_1} contains $X_{R(a,b)}$ and therefore $X_{R(a,b)} \wedge X_{R(a,a)}$
- Then, Φ_{Q_1} also contains $X_{R(a,a)} \wedge X_{R(a,a)} = X_{R(a,a)}$
- Thus, by the absorption law,

$$(X_{R(a,b)} \wedge X_{R(a,a)}) \vee X_{R(a,a)} = X_{R(a,a)}$$

- $X_{R(a,b)}$ can be eliminated from Φ_{Q_1} so that Q_1 and Q_2 are independent

Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
   - Syntactic Independence
   - Six Simple Rules
   - Tractability and Completeness
   - Extensional Plans
4. Intensional Query Evaluation
   - Syntactic independence
   - 5 Simple Rules
   - Query Compilation
   - Approximation Techniques
5. Summary

Base case: Atoms

Definition

If Q is an atom, i.e., of form $Q = R(\mathbf{a})$, simply lookup its probability in the database.

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did Mary see a toucan?
- $Q = \text{Sightings}(\text{Mary}, \text{Toucan})$
- $\mathbb{P}(Q) = 0.3$

Rule 1: Independent-join

Definition

If Q_1 and Q_2 are syntactically independent, then

$$\mathbb{P}(\,Q_1 \wedge Q_2\,) = \mathbb{P}(\,Q_1\,) \cdot \mathbb{P}(\,Q_2\,)\,. \qquad (\textit{independent-join})$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did both Mary and Susan see a toucan?
- $Q = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \wedge \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan})$
- $Q_1 = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \qquad \mathbb{P}(\,Q_1\,) = 0.3$
- $Q_2 = \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan}) \qquad \mathbb{P}(\,Q_2\,) = 0.5$
- $\mathbb{P}(\,Q\,) = \mathbb{P}(\,Q_1\,) \cdot \mathbb{P}(\,Q_2\,) = 0.15$

Rule 2: Independent-union

Definition

If Q_1 and Q_2 are syntactically independent, then

$$\mathbb{P}(Q_1 \vee Q_2) = 1 - (1 - \mathbb{P}(Q_1))(1 - \mathbb{P}(Q_2)). \quad (\textit{independent-union})$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did Mary or Susan see a toucan?
- $Q = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \vee \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan})$
- $Q_1 = \mathrm{S}(\mathrm{Mary}, \mathrm{Toucan}) \qquad \mathbb{P}(Q_1) = 0.3$
- $Q_2 = \mathrm{S}(\mathrm{Susan}, \mathrm{Toucan}) \qquad \mathbb{P}(Q_2) = 0.5$
- $\mathbb{P}(Q) =$
  $1 - (1 - \mathbb{P}(Q_1))(1 - \mathbb{P}(Q_2)) = 0.65$

Root variables and separator variables

Definition

Consider atom L and query Q. Denote by $\mathrm{Pos}(L, x)$ the set of positions where x occurs in Q (maybe empty). If Q is of form $Q = \exists x.Q'$:

- Variable x is a *root variable* if it occurs in all atoms, i.e., $\mathrm{Pos}(L, x) \neq \emptyset$ for every atom L that occurs in Q'.
- A root variable x is a *separator variable* if for any two atoms that unify, x occurs on a common position, i.e., $\mathrm{Pos}(L_1, x) \cap \mathrm{Pos}(L_2, x) \neq \emptyset$.

Example

$Q_1 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Likes}(x, a)$

- $\mathrm{Pos}(\mathrm{Likes}(a, x), x) = \{ 2 \}$
- $\mathrm{Pos}(\mathrm{Likes}(x, a), x) = \{ 1 \}$
- x is root variable
- x is no separator variable

$Q_2 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Likes}(x, x)$

- x is root variable
- x is a separator variable

$Q_3 \leftarrow \exists x.\mathrm{Likes}(a, x) \wedge \mathrm{Popular(a)}$

- x is no root variable
- x is no separator variable

Separator variables and syntactic independence

Lemma

Let x be a separator variable in $Q = \exists x.Q'$. Then for any two distinct constants a, b, the queries $Q'[a/x]$, $Q'[b/x]$ are syntactically independent.

Proof.

Any two atoms L_1, L_2 that unify in Q' do not unify in $Q'[a/x]$ and $Q'[b/x]$. Since x is a separator variable, there is a position at which both L_1 and L_2 have x; at this position, $L_1[a/x]$ has a and $L_2[b/x]$ has b. □

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Has anybody seen a toucan?
- $Q = \exists x.\text{Sightings}(x, \text{Toucan})$
- $Q'(x) = \text{Sightings}(x, \text{Toucan})$
- $Q'[\text{Mary}/x] = \text{Sightings}(\text{Mary}, \text{Toucan})$
- $Q'[\text{Susan}/x] = \text{Sightings}(\text{Susan}, \text{Toucan})$

Rule 3: Independent-project

Definition

If Q is of form $Q = \exists x . Q'$ and x is a separator variable, then

$$\mathbb{P}\left(Q \right) = 1 - \prod_{a \in \mathrm{ADom}} \left(1 - \mathbb{P}\left(Q'[a/x] \right)\right), \qquad (\textit{independent-project})$$

where ADom is the active domain of the database.

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Has anybody seen a toucan?
- $Q = \exists x . \mathrm{S}(x, \mathrm{Toucan})$
- $Q' = \mathrm{S}(x, \mathrm{Toucan})$
- $$\begin{aligned} \mathbb{P}\left(Q \right) &= 1 - \prod_{x \in \{ \mathrm{M,S,F,\ldots} \}} \left(1 - \mathbb{P}\left(\mathrm{S}(x, \mathrm{T}) \right)\right) \\ &= 1 - (1 - 0.3)(1 - 0.5)1 \cdots 1 \\ &= 0.65 \end{aligned}$$

Rule 4: Negation

Definition

If the query is $\neg Q$, then

$$\mathbb{P}(\neg Q) = 1 - \mathbb{P}(Q) \qquad (\textit{negation})$$

Example

Sightings

Name	Species	$\mathbb{P}$	
Mary	Finch	0.8	X_1
Mary	Toucan	0.3	X_2
Susan	Finch	0.2	X_3
Susan	Toucan	0.5	X_4
Susan	Nightingale	0.6	X_5

- Did nobody see a toucan?
- $Q = \neg[\exists x.S(x, \text{Toucan})]$
- $\mathbb{P}(Q) = 1 - \mathbb{P}(\exists x.S(x, \text{Toucan})) = 0.35$

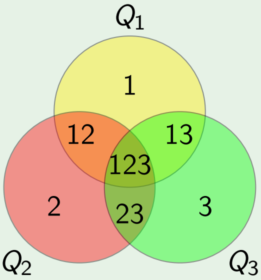
Rule 5: Inclusion-exclusion

Definition

Suppose $Q = Q_1 \wedge Q_2 \wedge \ldots Q_k$. Then,

$$\mathbb{P}(\,Q\,) = -\sum_{\emptyset \neq S \subseteq \{\,1,\ldots,k\,\}} (-1)^{|S|}\,\mathbb{P}(\,\bigvee_{i \in S} Q_i\,) \qquad (\textit{inclusion-exclusion})$$

Example



1	2	3	12	13	23	123	$\mathbb{P}(\,Q_1 \wedge Q_2 \wedge Q_3\,) =$
1	0	0	1	1	0	1	$+\,\mathbb{P}(\,Q_1\,)$
1	1	0	2	1	1	2	$+\,\mathbb{P}(\,Q_2\,)$
1	1	1	2	2	2	3	$+\,\mathbb{P}(\,Q_3\,)$
0	0	1	1	1	1	2	$-\,\mathbb{P}(\,Q_1 \vee Q_2\,)$
-1	0	0	0	0	0	1	$-\,\mathbb{P}(\,Q_1 \vee Q_3\,)$
-1	-1	-1	-1	-1	-1	0	$-\,\mathbb{P}(\,Q_2 \vee Q_3\,)$
0	0	0	0	0	0	1	$+\,\mathbb{P}(\,Q_1 \vee Q_2 \vee Q_3\,)$

Inclusion-exclusion for independent-project

Goal of inclusion-exclusion is to apply the rewrite

$$(\exists x_1.Q_1) \vee (\exists x_2.Q_2) \equiv \exists x.(Q_1[x/x_1] \vee Q_2[x/x_2]).$$

Example

Sightings

Name	Species	$\mathbb{P}$
Mary	Finch	0.8
Mary	Toucan	0.3
Susan	Finch	0.2
Susan	Toucan	0.5
Susan	Nightingale	0.6

Has both Mary seen some bird and someone seen a finch?

$$\mathbb{P}\left((\exists x.\mathrm{S}(\mathrm{M},x)) \wedge (\exists y.\mathrm{S}(y,\mathrm{F})) \right) \qquad (ie)$$

$$= \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \right) + \mathbb{P}\left(\exists y.\mathrm{S}(y,\mathrm{F}) \right) - \mathbb{P}\left((\exists x.\mathrm{S}(\mathrm{M},x)) \vee (\exists y.\mathrm{S}(y,\mathrm{F})) \right) \quad (ip/ip/rewrite)$$

$$= 0.86 + 0.84 - \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \vee \mathrm{S}(x,\mathrm{F}) \right)$$

$$= 1.7 - \mathbb{P}\left(\exists x.\mathrm{S}(\mathrm{M},x) \vee \mathrm{S}(x,\mathrm{F}) \right)$$

Now we are stuck → Need another rule (attribute-constant ranking)!

Rule 6: Attribute ranking

Definition

Attribute-constant ranking. If Q is a query that contains a relation name R with attribute A, and there exists two unifiable atoms such that the first has constant a at position A and the second has a variable, substitute each occurence of form $R(\dots)$ by $R_1(\dots) \vee R_2(\dots)$, where

$$R_1 = \sigma_{A=a}(R), \qquad R_2 = \sigma_{A\neq a}(R).$$

Attribute-attribute ranking. If Q is a query that contains a relation name R with attributes A and B, substitute each occurence of form $R(\dots)$ by $R_1(\dots) \vee R_2(\dots) \vee R_3(\dots)$, where

$$R_1 = \sigma_{A<B}(R), \qquad R_2 = \sigma_{A=B}(R), \qquad R_3 = \sigma_{A>B}(R).$$

Syntactic rewrites. For selections of form $\sigma_{A=\cdot}$, decrease the arity of the resulting relation by 1 and add an equality predicate.

Attribute-constant ranking (continues prev. example)

Example

Has both Mary seen some bird and someone seen a finch?

$$
\begin{aligned}
&\mathbb{P}(\,(\exists x.\mathsf{S}(\mathsf{M},x)) \wedge (\exists y.\mathsf{S}(y,\mathsf{F}))\,)\\
&= 1.7 - \mathbb{P}(\,\exists x.\mathsf{S}(\mathsf{M},x) \vee \mathsf{S}(x,\mathsf{F})\,) && (\textit{rank}\ (\text{Name}{=}\text{Mary}))\\
&= 1.7 - \mathbb{P}(\,\exists x.\mathsf{S}_{\mathsf{M}}(x) \vee S_{\neg\mathsf{M}}(\mathsf{M},x) \vee [\mathsf{S}_{\mathsf{M}}(\mathsf{F}) \wedge x = \mathsf{M}] \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,) && (\textit{simplify})\\
&= 1.7 - \mathbb{P}(\,\exists x.\mathsf{S}_{\mathsf{M}}(x) \vee \mathsf{S}_{\mathsf{M}}(\mathsf{F}) \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,) && (\textit{rank}\ (\text{Species}{=}\text{Finch}))\\
&= 1.7 - \mathbb{P}(\,\exists x.[\mathsf{S}_{\mathsf{MF}}() \wedge x = \mathsf{F}] \vee \mathsf{S}_{\mathsf{M}\neg\mathsf{F}}(x) \vee \mathsf{S}_{\mathsf{MF}}() \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,) && (\textit{push}\ \exists x)\\
&= 1.7 - \mathbb{P}(\,\mathsf{S}_{\mathsf{MF}}() \vee \exists x.\mathsf{S}_{\mathsf{M}\neg\mathsf{F}}(x) \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,) && (\textit{iu})\\
&= 1.7 - 1 + (1 - \mathbb{P}(\,\mathsf{S}_{\mathsf{MF}}()\,))(1 - \mathbb{P}(\,\exists x.\mathsf{S}_{\mathsf{M}\neg\mathsf{F}}(x) \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,) && (\textit{base}/\textit{ip})\\
&= 0.7 + (1 - 0.8)\Big[\textstyle\prod_{x\in\{\,\mathsf{M,S,F,T,N}\,\}}(1 - \mathbb{P}(\,\mathsf{S}_{\mathsf{M}\neg\mathsf{F}}(x) \vee \mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,)\Big] && (\textit{iu})\\
&= 0.7 + 0.2\Big[\textstyle\prod_{x\in\{\,\mathsf{M,S,F,T,N}\,\}}(1 - \mathbb{P}(\,\mathsf{S}_{\mathsf{M}\neg\mathsf{F}}(x)\,))(1 - \mathbb{P}(\,\mathsf{S}_{\neg\mathsf{M}}(x,\mathsf{F})\,))\Big] && (\textit{product})\\
&= 0.7 + 0.2[11 \cdot 1(1 - 0.2) \cdot 11 \cdot (1 - 0.3)1 \cdot 11]\\
&= 0.812
\end{aligned}
$$

S

N	S	$\mathbb{P}$
M	F	0.8
M	T	0.3
S	F	0.2
S	T	0.5
S	N	0.6

S_M

S	$\mathbb{P}$
F	0.8
T	0.3

$S_{\neg M}$

N	S	$\mathbb{P}$
S	F	0.2
S	T	0.5
S	N	0.6

S_{MF}

$\mathbb{P}$
0.8

$S_{M\neg F}$

S	$\mathbb{P}$
T	0.3

Attribute-attribute ranking (example)

The goal of attribute ranking is to establish syntactic independence and new separators by exploiting disjointness.

L

P_1	P_2	$\mathbb{P}$
A	B	0.8
B	A	0.7
C	A	0.2
C	C	0.9

$L_<$

P_1	P_2	$\mathbb{P}$
A	B	0.8

$L_=$

P_{12}	$\mathbb{P}$
C	0.9

$L_>$

P_1	P_2	$\mathbb{P}$
B	A	0.7
C	A	0.2

Example

Are there two people who like each other?

$$\begin{aligned}
&\mathbb{P}(\exists x.\exists y.\mathrm{Likes}(x,y) \wedge \mathrm{Likes}(y,x)) && (\mathit{rank})\\
&= \mathbb{P}(\exists x.\exists y.\\
&\qquad (\mathrm{Likes}_<(x,y) \vee (\mathrm{Likes}_=(x) \wedge x = y) \vee \mathrm{Likes}_>(x,y)) \wedge\\
&\qquad (\mathrm{Likes}_<(y,x) \vee (\mathrm{Likes}_=(x) \wedge x = y) \vee \mathrm{Likes}_>(y,x))) && (\mathit{expand}, \mathit{disjoint})\\
&= \mathbb{P}(\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x) \vee (\mathrm{L}_=(x) \wedge x = y) \vee \mathrm{L}_>(x,y)\mathrm{L}_<(y,x)) && (\mathit{push}\ \exists)\\
&= \mathbb{P}((\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x))\\
&\qquad \vee (\exists x.\mathrm{L}_=(x))\\
&\qquad \vee (\exists x.\exists y.\mathrm{L}_>(x,y)\mathrm{L}_<(y,x))) && (1\mathit{st} \equiv 3\mathit{rd})\\
&= \mathbb{P}((\exists x.\exists y.\mathrm{L}_<(x,y)\mathrm{L}_>(y,x))\\
&\qquad \vee (\exists x.\mathrm{L}_=(x)))
\end{aligned}$$

Now we can apply independent-union, then independent-project, then independent-join.

Outline

Inclusion-exclusion and cancellation

Consider the query

$$Q \leftarrow (Q_1 \vee Q_3) \wedge (Q_1 \vee Q_4) \wedge (Q_2 \vee Q_4)$$

Apply inclusion exclusion to get

$$\begin{aligned}\mathbb{P}(Q) = {} & \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) \\ & - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4) \\ & + \mathbb{P}(Q_1 \vee Q_2 \vee Q_3 \vee Q_4) \\ = {} & \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) \\ & - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4)\end{aligned}$$

One can construct cases in which $Q_1 \vee Q_2 \vee Q_3 \vee Q_4$ is hard, but any subset is not (e.g., consider H_3 on slide 20).

The inclusion-exclusion formula needs to be replaced by the Möbius inversion formula.

Möbius inversion formula (example)

Given a query expression of form $Q_1 \wedge \ldots \wedge Q_k$:

1. Put the formulas $Q_S = \bigvee_{i \in S} Q_i$, $\emptyset \neq S \subseteq \{1, \ldots, j\}$, in a lattice (plus special element $\hat{1}$)
2. Eliminate duplicates (equivalent formulas)
3. Use the partial order $Q_{S_1} \geq Q_{S_2}$ iff $Q_{S_1} \Leftarrow Q_{S_2}$
4. Label each node by its Möbius value

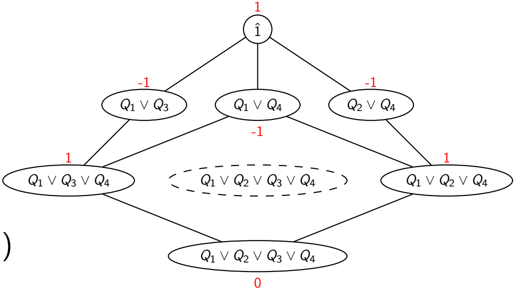
$$\mu(\hat{1}) = 1$$

$$\mu(u) = -\sum_{u < w \leq \hat{1}} \mu(w)$$

5. Use the inversion formula

$$\mathbb{P}(Q_1 \wedge \ldots \wedge Q_k) = -\sum_{u < \hat{1} : \mu(u) \neq 0} \mu(u)\,\mathbb{P}(Q_u)$$

$Q \leftarrow (Q_1 \vee Q_3) \wedge (Q_1 \vee Q_4) \wedge (Q_2 \vee Q_4)$



$$\mathbb{P}(Q) = \mathbb{P}(Q_1 \vee Q_3) + \mathbb{P}(Q_1 \vee Q_4) + \mathbb{P}(Q_2 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_3 \vee Q_4) - \mathbb{P}(Q_1 \vee Q_2 \vee Q_4)$$

An nondeterministic algorithm

Consider the algorithm:

1. As long as possible, apply one of the rules R1–R6
2. If all formulas are atoms, SUCCESS
3. If there is a formula that is not an atom, FAILURE

Definition

A rule is $\mathbf{R}_6$-*safe* if the above algorithm succeeds.

- Order of rule application does not affect SUCCESS
- Algorithm is polynomial in size of database
  - Easy to see for independent-join, independent-union, negation, Möbius inversion formula, attribute ranking → do not depend on database
  - Independent-project increases number of queries by a factor of $|\text{ADom}|$ → applied at most k times, where k is the maximum arity of a relation

How the rules fail

Example

Consider the hard query

$$H_0 \leftarrow \exists x.\exists y.R(x) \wedge S(x,y) \wedge T(y)$$

- independent-join, independent-union, independent-project, negation, Möbius inversion formula all do not apply
- But we could rank S:

$$\begin{aligned} H_0 &\leftarrow H_{01} \vee H_{02} \vee H_{03} \\ H_{01} &\leftarrow \exists x.\exists y.R(x) \wedge S_<(x,y) \wedge T(y) \\ H_{02} &\leftarrow \exists x.R(x) \wedge S_=(x) \wedge T(x) \\ H_{03} &\leftarrow \exists x.\exists y.R(x) \wedge S_>(y,x) \wedge T(y) \end{aligned}$$

- Now we are stuck at H_{01} and H_{03}

Dichotomy theorem for $\mathcal{UCQ}$

- Safety is a syntactic property
- Tractability is a semantic property
- What is their relationship?

Theorem (Dalvi and Suciu, 2010)

For any $\mathcal{UCQ}$ query Q, one of the following holds:

- *Q is $\mathbf{R}_6$-safe, or*
- *the data complexity of Q is hard for $\#P$.*

- No queries of "intermediate" difficulty
- Can check for tractability in time polynomial in database size (can be done by assuming an active domain of size 1)
- Query complexity is unknown (Möbius inversion formula)
- For $\mathcal{RC}$, completeness/dichotomy unknown

We can handle all safe $\mathcal{UCQ}$ queries!

Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
    - Syntactic Independence
    - Six Simple Rules
    - Tractability and Completeness
    - Extensional Plans
4. Intensional Query Evaluation
    - Syntactic independence
    - 5 Simple Rules
    - Query Compilation
    - Approximation Techniques
5. Summary

Overview of extensional plans

Can we evaluate safe queries directly in an RDBMS?

- Extensional query evaluation
  - Based on the query expression
  - Uses rules to break query into simpler pieces
  - For $\mathcal{UCQ}$, detects whether queries are tractable or intractable
- Extensional operators
  - Extend relational operators by probability computation
  - Standard database algorithms can be used
- Extensional plans
  - Can be safe (correct) or unsafe (incorrect)
  - For tractable $\mathcal{UCQ}$ queries, we can always produce a safe plan
  - Plan construction based on $\mathbf{R}_6$ rules
  - Can be written in SQL (though not "best" approach)
  - **Enables scalable query processing on probabilistic databases**

Basic operators

Definition

Annotate each tuple by its probability. The operators

- *Independent join* ($\bowtie^{\mathrm{i}}$)
- *Independent project* (π^{i})
- *Independent union* ($\cup^{\mathrm{i}}$)
- Construction / selection / renaming

correspond to the positive K-relational algebra over $([0,1],0,1,\oplus,\cdot)$, where $p_1 \oplus p_2 = 1-(1-p_1)(1-p_2)$.

(Union needs to be replaced by outer join for non-matching schemas; see Sucio, Olteneau, Ré, Koch, 2011.)

$\boxed{([0,1],0,1,\oplus,\cdot) \text{ is not a semiring} \rightarrow \text{unsafe plans!}}$

Example plans

Incriminates

Witness	Suspect	
Mary	Paul	p_1
Mary	John	p_2
Susan	John	p_3

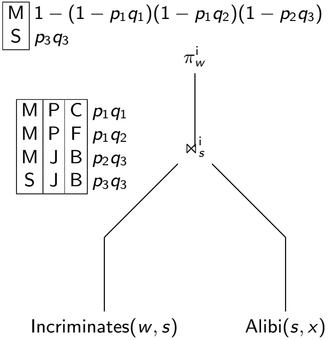
Alibi

Suspect	Claim	
Paul	Cinema	q_1
Paul	Friend	q_2
John	Bar	q_3

Who incriminates someone
who has an alibi?

$$Q_1(w) \leftarrow \exists s.\exists x.\mathrm{Incriminates}(w,s) \wedge \mathrm{Alibi}(s,x)$$

$$Q_2(w) \leftarrow \exists s.\mathrm{Incriminates}(w,s) \wedge \exists x.\mathrm{Alibi}(s,x)$$

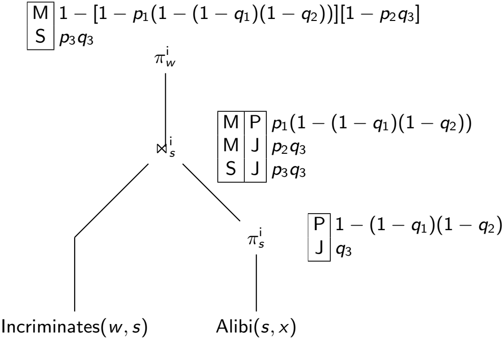


M	$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)$
S	p_3q_3

M	P	C	p_1q_1
M	P	F	p_1q_2
M	J	B	p_2q_3
S	J	B	p_3q_3

Plan 1

Incorrect (unsafe)



M	$1-[1-p_1(1-(1-q_1)(1-q_2))][1-p_2q_3]$
S	p_3q_3

M	P	$p_1(1-(1-q_1)(1-q_2))$
M	J	p_2q_3
S	J	p_3q_3

P	$1-(1-q_1)(1-q_2)$
J	q_3

Plan 2

Correct (safe)

Not all plans are safe!

Weighted sum

How to deal with the Möbius inversion formula?

Definition

The *weighted sum* of relations $R_1, \ldots, R_k$ with parameters $\mu_1, \ldots, \mu_k$ is given by:

$$\left(\sum_{U}^{\mu_1,\ldots,\mu_k} (R_1, \ldots, R_k)\right)[] = R_1 \bowtie \cdots \bowtie R_k$$

$$\left(\sum_{U}^{\mu_1,\ldots,\mu_k} (R_1, \ldots, R_k)\right)(t) = \mu_1(R_1(t)) + \cdots \mu_k(R_k(t))$$

Intuitively,

- Computes the natural join
- Sums up the weighted probabilities of joining tuples

Weighted sum (example)

Example

Consider relations/subqueries $V_1(A, B)$ and $V_2(A, C)$ and the query:

$$Q(x, y, z) \leftarrow V_1(x, y) \wedge V_2(x, z)$$

Suppose we apply the Möbius inversion formula to get:

- $Q_1(x, y) = V_1(x, y)$ with $\mu_1 = 1$
- $Q_2(x, z) = V_2(x, z)$ with $\mu_2 = 1$
- $Q_3(x, y, z) = V_1(x, y) \vee V_2(x, z)$ with $\mu_3 = -1$

We obtain:

$$\sum_{\{\,A,B,C\,\}}^{1,1,-1} (Q_1, Q_2, Q_3)[] = Q_1 \bowtie Q_2 \bowtie Q_3 = V_1 \bowtie V_2$$

$$\sum_{\{\,A,B,C\,\}}^{1,1,-1} (Q_1, Q_2, Q_3) = \{\, (t, p_{t_1} + p_{t_2} - p_{t_3}) : t[AB] = t_1 \in Q_1, t[AC] = t_2 \in Q_2, \\ t[ABC] = t_3 \in Q_3 \,\}$$

Complement

How to deal with negation?

Definition

The *complement* of a deterministic relation R of arity k is given by

$$C(R) = \left\{ (t, 1 - \mathbb{P}(\, t \in R \,)) : t \in \mathrm{ADom}^k \right\}.$$

In practice, every complement operation can be replaced by difference (since queries are domain-independent).

Example

- Query: $Q \leftarrow R(x) \land \neg S(x)$
- Result: $R -^{\mathrm{i}} S = \{\, (t, \mathbb{P}(\, t \in R \,)\,(1 - \mathbb{P}(\, t \in S \,))) : t \in R \,\}$

Computation of safe plans (1)

Definition

A query plan for Q is *safe* if it computes the correct probabilities for all input databases.

Theorem

There is an algorithm A that takes in a query Q and outputs either FAIL of a safe plan for Q. If Q is a $\mathcal{UCQ}$ query, A fails only if Q is intractable.

- Key idea: Apply rules R1–R6, but produce a query plan instead of computing probabilities
- Extension to non-Boolean queries: treat head variables as "constants"
- Ranking step produces "views" that are treated as base tables

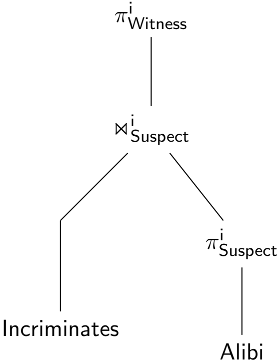
Computation of safe plans (2)

```
if $Q = Q_1 \wedge Q_2$ and $Q_1, Q_2$ are syntactically independent then
    return plan($Q_1$) $\bowtie^{\mathrm{i}}$ plan($Q_2$)
end if
if $Q = Q_1 \vee Q_2$ and $Q_1, Q_2$ are syntactically independent then
    return plan($Q_1$) $\cup^{\mathrm{i}}$ plan($Q_2$)
end if
if $Q(\mathbf{x}) = \exists z.Q_1(\mathbf{x}, z)$ and $z$ is a separator variable then
    return $\pi^{\mathrm{i}}_{\mathbf{x}}$(plan($Q_1(\mathbf{x}, z)$))
end if
if $Q = Q_1 \wedge \ldots \wedge Q_k,\ k \geq 2$ then
    Construct CNF lattice $Q'_1, \ldots, Q'_m$
    Compute Möbius coefficients $\mu_1, \ldots, \mu_m$
    return $\sum^{\mu_1,\ldots,\mu_m}$(plan($Q'_1$), ..., plan($Q'_m$))
end if
if $Q = \neg Q_1$ then
    return $C$(plan $Q_1$)
end if
if $Q(\mathbf{x}) = R(\mathbf{x})$ where $R$ is a base table (possibly ranked) then
    return $R(\mathbf{x})$
end if
otherwise FAIL
```

Computation of safe plans (example)

$Q(w) \leftarrow \exists s.\exists x.\text{Incriminates}(w,s) \wedge \text{Alibi}(s,x)$

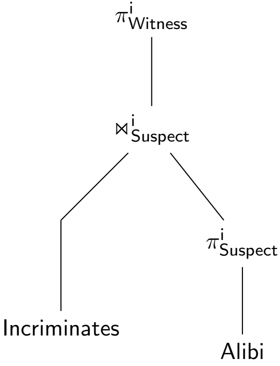
1. Apply independent-project to Q on s
    - $Q_1(w,s) \leftarrow \exists x.\text{Incriminates}(w,s) \wedge \text{Alibi}(s,x)$
2. x is not a root variable in Q_1 → push $\exists x$:
   $Q_2(w,s) \leftarrow \text{Incriminates}(w,s) \wedge \exists x.\text{Alibi}(s,x)$
3. Apply independent-join to Q_2
    - $Q_3(w,s) \leftarrow \text{Incriminates}(w,s)$
    - $Q_4(s) \leftarrow \exists x.\text{Alibi}(s,x)$
4. Q_3 is an atom
5. Apply independent-project to Q_4 on x
    - $Q_5(s,x) = \text{Alibi}(s,x)$
6. Q_5 is an atom



Safe plans with PostgreSQL (example)

$Q(w) \leftarrow \exists s.\exists x.\mathsf{Incriminates}(w,s) \wedge \mathsf{Alibi}(s,x)$

- $Q_4 \leftarrow \pi^{\mathsf{i}}_{\mathsf{Suspect}}(\mathsf{Alibi})$
- $Q_2 \leftarrow \mathsf{Incriminates} \bowtie^{\mathsf{i}}_{\mathsf{Suspect}} Q_4$
- $Q \leftarrow \pi^{\mathsf{i}}_{\mathsf{Witness}}(Q_2)$



```
SELECT Witness, 1-PRODUCT(1-P) AS P
FROM (
       SELECT Witness, Incriminates.Suspect,
              Incriminates.P * Q4.P as P
       FROM Incriminates,
            (
              SELECT Suspect, 1-PRODUCT(1-P) AS P
              FROM Alibi
              GROUP BY Suspect
            ) AS Q4
       WHERE Incriminates.Suspect = Q4.Suspect
     ) AS Q2
GROUP BY Witness
```

Deterministic tables

- Often: Mix of probabilistic and deterministic tables
- Naive approach: Assign probability 1 to tuples in a deterministic table
  → Suboptimal: Some tractable queries are missed!

Example

- If T is known to be deterministic, the query

  $$Q \leftarrow R(x), S(x,y), T(y)$$

  becomes tractable!
- Why? $S \bowtie T$ now is a tuple-independent table!
- We can use the safe plan
  $$\pi^{\mathrm{i}}_{\emptyset}\left[R(x) \bowtie^{\mathrm{i}}_{x} (S(x,y) \bowtie_{y} T(y))\right]$$

Additional information about the nature of the tables (e.g., deterministic, tuple-independent with keys, BID tables) can help extensional query processing.

Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
    - Syntactic Independence
    - Six Simple Rules
    - Tractability and Completeness
    - Extensional Plans
4. Intensional Query Evaluation
    - Syntactic independence
    - 5 Simple Rules
    - Query Compilation
    - Approximation Techniques
5. Summary

Overview

Given a query $Q(\mathbf{x})$, a TI database $\mathcal{D}$; for each output tuple t

1. Compute the lineage $\Phi = \Phi^{\mathcal{D}}_{Q(t)}$
   - $|\Phi| = O(|\text{ADom}|^m)$, where m is the number of variables in Φ
   - Data complexity is polynomial time
   - Difference to extensional query evaluation: $|\Phi|$ depends on input
     $\rightarrow$ rules exponential in $|\Phi|$ also exponential in the size of the input!
2. Compute the probability $\mathbb{P}(\ \Phi\ )$
   - Intensional query evaluation $\approx$ probability computation on propositional formulas
   - Studied in verification and AI communities
   - Different approaches: rule-based evaluation, formula compilation, approximation

Can deal with hard queries.

Example (tractable query)

Example

$$q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	X_1
R2	Single	H1	\$600	X_2
R3	Double	H1	\$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

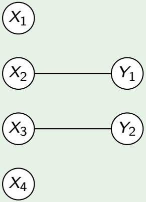
- $\Phi = X_4 \wedge (X_1 \vee X_2)$
- $\mathbb{P}(\Phi) = \mathbb{P}(X_4)[1 - (1 - \mathbb{P}(X_1))(1 - \mathbb{P}(X_2))]$
- E.g., $\mathbb{P}(X_i) = \frac{1}{2}$ for all $i \rightarrow \mathbb{P}(\Phi) = 0.375$

ExpensiveHotels

HotelNo	$\mathbb{P}$
H1	0.375

Example (intractable query)

Example



R

X_1	0.5
X_2	0.5
X_3	0.5
X_4	0.5

S

X_2	Y_1	1
X_3	Y_2	1

T

Y_1	0.5
Y_2	0.5

- $H_0 \leftarrow \exists x.\exists y.R(x), S(x,y), T(y)$
- $\Phi = X_2 Y_1 \vee X_3 Y_2$
- $\mathbb{P}(\Phi) = 1 - (1 - \mathbb{P}(X_2)\mathbb{P}(Y_1))(1 - \mathbb{P}(X_3)\mathbb{P}(Y_2)) = 0.4375$
- Model counting: $\#\Phi = 2^6\,\mathbb{P}(\Phi) = 28$
- Bipartite vertex cover: $\#\Psi = 2^6 - \#\Phi = 36 = 2 \cdot 3 \cdot 3 \cdot 2$

Outline

Overview of rule-based intensional query evaluation

- Break the lineage formula into "simpler" formulas
- By applying one of the rules
  1. Independent-and
  2. Independent-or
  3. Disjoint-or
  4. Negation
  5. Shannon expansion
- Rules work on lineage, not on query → data dependent
- Rules *always* succeed
- Rule 5 may lead to exponential blowup

Can be used on any query but data complexity can be exponential. However, depending on the database, even a hard query might be "easy" to evaluate.

Support

Definition

For a propositional formula Φ, denote by $V(\Phi)$ the set of variables that occur in Φ. Denote by $\mathrm{Var}(\Phi)$ the set of variables on which Φ depends; $\mathrm{Var}(\Phi)$ is called the *support* of Φ. $X \in \mathrm{Var}(\Phi)$ iff there exists an assignment θ to all variables but X and constants $a \neq b$ such that $\Phi[\theta \cup \{ X \mapsto a \}] \neq \Phi[\theta \cup \{ X \mapsto b \}]$.

Example

$\Phi = X \vee (Y \wedge Z)$

- $V(\Phi) = \{ X, Y, Z \}$
- $\mathrm{Var}(\Phi) = \{ X, Y, Z \}$

$\Phi = Y \vee (X \wedge Y) \equiv Y$

- $V(\Phi) = \{ X, Y \}$
- $\mathrm{Var}(\Phi) = \{ Y \}$

Syntactic independence

Definition

Φ_1 and Φ_2 are *syntactically independent* if they have disjoint support, i.e., $\mathrm{Var}(\Phi_1) \cap \mathrm{Var}(\Phi_2) = \emptyset$.

Example

$\Phi_1 = X \qquad \Phi_2 = Y \qquad \Phi_3 = \neg X \neg Y \vee XY$

- Φ_1 and Φ_2 are syntactically independent
- All other combinations are not

Checking for syntactic independence is co-NP-complete in general.

Practical approach:

Proposition

A sufficient condition for syntactic independence is $V(\Phi_1) \cap V(\Phi_2) = \emptyset$*.*

Probabilistic independence

Proposition

If $\Phi_1, \Phi_2, \ldots, \Phi_k$ are pairwise syntactically independent, then the probabilistic events $\Phi_1, \Phi_2, \ldots, \Phi_k$ are independent.

Note that pairwise *probabilistic* independence does not imply probabilistic independence!

Example

$\Phi_1 = X \qquad \Phi_2 = Y \qquad \Phi_3 = \neg X \neg Y \vee XY$

- Φ_1 and Φ_2 are probabilistically independent
- Φ_1, Φ_2, Φ_3 are not pairwise syntactically independent

Assume $\mathbb{P}(X) = \mathbb{P}(Y) = 1/2$

- Φ_1, Φ_2, Φ_3 are pairwise independent
- Φ_1, Φ_2, Φ_3 are not independent!

Outline

Rules 1 and 2: independent-and, independent-or

Definition

Let Φ_1 and Φ_2 be two syntactically independent propositional formulas:

$$\begin{aligned} \mathbb{P}(\,\Phi_1 \wedge \Phi_2\,) &= \mathbb{P}(\,\Phi_1\,) \cdot \mathbb{P}(\,\Phi_2\,) && (\textit{independent-and}) \\ \mathbb{P}(\,\Phi_1 \vee \Phi_2\,) &= 1 - (1 - \mathbb{P}(\,\Phi_1\,))(1 - \mathbb{P}(\,\Phi_2\,)) && (\textit{independent-or}) \end{aligned}$$

Independent-and, independent-or (example)

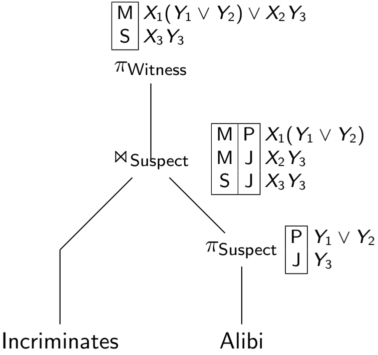
Incriminates

Witness	Suspect	
Mary	Paul	X_1 (p_1)
Mary	John	X_2 (p_2)
Susan	John	X_3 (p_3)

Alibi

Suspect	Claim	
Paul	Cinema	Y_1 (q_1)
Paul	Friend	Y_2 (q_2)
John	Bar	Y_3 (q_3)

$$Q(w) \leftarrow \exists s . \exists x . \text{Incriminates}(w, s) \wedge \text{Alibi}(s, x)$$



M	$X_1(Y_1 \vee Y_2) \vee X_2 Y_3$
S	$X_3 Y_3$

π_{Witness}

M	P	$X_1(Y_1 \vee Y_2)$
M	J	$X_2 Y_3$
S	J	$X_3 Y_3$

$\bowtie_{\text{Suspect}}$

P	$Y_1 \vee Y_2$
J	Y_3

π_{Suspect}

Incriminates

Alibi

- $\Phi_S = X_3 Y_3$
  1. Independent-and: $\mathbb{P}(\Phi_S) = p_3 q_3$
- $\Phi_M = X_1(Y_1 \vee Y_2) \vee X_2 Y_3$
  1. Independent-or:
     $\mathbb{P}(\Phi_M) = 1 - (1 - \mathbb{P}(X_1(Y_1 \vee Y_2)))(1 - \mathbb{P}(X_2 Y_3))$
  2. Independent-and: $\mathbb{P}(X_2 Y_3) = p_2 q_3$
  3. Independent-and:
     $\mathbb{P}(X_1(Y_1 \vee Y_2)) = p_1 \mathbb{P}(Y_1 \vee Y_2)$
  4. Independent-or:
     $\mathbb{P}(Y_1 \vee Y_2) = 1 - (1 - q_1)(1 - q_2)$
  5. $\mathbb{P}(\Phi_M) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2 q_3)$

Rule 3: Disjoint-or

Definition

Two propositional formulas Φ_1 and Φ_2 are *disjoint* if $\Phi_1 \wedge \Phi_2$ is not satisfiable.

Definition

If Φ_1 and Φ_2 are disjoint:

$$\mathbb{P}(\Phi_1 \vee \Phi_2) = \mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) \qquad (\textit{disjoint-or})$$

Example

- $\mathbb{P}(X) = 0.2; \quad \mathbb{P}(Y) = 0.7$
- $\Phi_1 = XY; \quad \mathbb{P}(XY) = \mathbb{P}(X)\mathbb{P}(Y) = 0.14$
- $\Phi_2 = \neg X; \quad \mathbb{P}(\neg X) = 0.8$
- $\mathbb{P}(\Phi_1 \vee \Phi_2) = \mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2) = 0.94$

Checking for disjointness is NP-complete in general. But disjoint-or will play a major role for Shannon expansion.

Rule 4: Negation

Definition

$$\mathbb{P}(\neg\Phi) = 1 - \mathbb{P}(\Phi) \qquad (\textit{negation})$$

Example

- $\mathbb{P}(X) = 0.2; \quad \mathbb{P}(Y) = 0.7$
- $\mathbb{P}(XY) = \mathbb{P}(X)\mathbb{P}(Y) = 0.14$
- $\mathbb{P}(\neg(XY)) = 1 - 0.14 = 0.86$

Shannon expansion

Definition

The *Shannon expansion* of a propositional formula Φ w.r.t. a variable X with domain $\{ a_1, \ldots, a_m \}$ is given by:

$$\Phi \equiv (\Phi[X \mapsto a_1] \wedge (X = a_1)) \vee \ldots \vee (\Phi[X \mapsto a_m] \wedge (X = a_m))$$

Example

- $\Phi = XY \vee XZ \vee YZ$
- $\Phi \equiv (\Phi[X \mapsto \texttt{TRUE}] \wedge X) \vee (\Phi[X \mapsto \texttt{FALSE}] \wedge \neg X)$
  $= (Y \vee Z)X \vee YZ\neg X$

In the Shannon expansion rule, every $\wedge$ is an independent-and; every $\vee$ is a disjoint-or.

Rule 5: Shannon expansion

Definition

Let Φ be a propositional formula and X be a variable:

$$\mathbb{P}(\Phi) = \sum_{a \in \text{dom}(X)} \mathbb{P}(\Phi[X \mapsto a])\,\mathbb{P}(X = a) \qquad (\textit{Shannon expansion})$$

Example

- $\Phi = XY \vee XZ \vee YZ$
- $\mathbb{P}(\Phi) = \mathbb{P}(Y \vee Z)\,\mathbb{P}(X) + \mathbb{P}(YZ)\,\mathbb{P}(\neg X)$

- Can always be applied
- Effectively eliminates X from the formula
- But may lead to exponential blowup!

Shannon expansion (example)

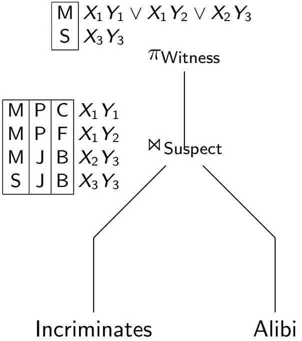
Incriminates

Witness	Suspect	
Mary	Paul	$X_1\ (p_1)$
Mary	John	$X_2\ (p_2)$
Susan	John	$X_3\ (p_3)$

Alibi

Suspect	Claim	
Paul	Cinema	$Y_1\ (q_1)$
Paul	Friend	$Y_2\ (q_2)$
John	Bar	$Y_3\ (q_3)$

$$Q(w) \leftarrow \exists s.\exists x.\text{Incriminates}(w, s) \land \text{Alibi}(s, x)$$



M	$X_1Y_1 \lor X_1Y_2 \lor X_2Y_3$
S	X_3Y_3

M	P	C	X_1Y_1
M	P	F	X_1Y_2
M	J	B	X_2Y_3
S	J	B	X_3Y_3

$$\Phi_{\text{M}} = X_1Y_1 \lor X_1Y_2 \lor X_2Y_3$$

① Independent-or:

$$\mathbb{P}(\Phi_{\text{M}}) = 1 - (1 - \mathbb{P}(X_1Y_1 \lor X_1Y_2))(1 - \mathbb{P}(X_2Y_3))$$

② Independent-and: $\mathbb{P}(X_2Y_3) = p_2q_3$

③ Shannon expansion: $\mathbb{P}((X_1Y_1 \lor X_1Y_2)) = \mathbb{P}(Y_1 \lor Y_2)\mathbb{P}(X_1) + \mathbb{P}(\texttt{FALSE})\mathbb{P}(\neg X_1)$

④ Independent-or:

$$\mathbb{P}(Y_1 \lor Y_2) = 1 - (1 - q_1)(1 - q_2)$$

⑤ $\mathbb{P}(\Phi_{\text{M}}) = 1 - [1 - p_1(1 - (1 - q_1)(1 - q_2))](1 - p_2q_3)$

The intensional rules work on all plans!

A non-deterministic algorithm

```
if $\Phi = \Phi_1 \wedge \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
    return $\mathbb{P}(\Phi_1) \cdot \mathbb{P}(\Phi_2)$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
    return $1 - (1 - \mathbb{P}(\Phi_1))(1 - \mathbb{P}(\Phi_2))$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and $\Phi_1, \Phi_2$ are disjoint then
    return $\mathbb{P}(\Phi_1) + \mathbb{P}(\Phi_2)$
end if
if $\Phi = \neg\Phi_1$ then
    return $1 - \mathbb{P}(\Phi_1)$
end if
Choose $X \in \mathrm{Var}(\Phi)$
return $\sum_{a \in \mathrm{dom}(X)} \mathbb{P}(\Phi[X \mapsto a])\, \mathbb{P}(X = a)$
```

Should be implemented with dynamic programming to avoid evaluating the same subformula multiple times.

Outline

Materialized views in TID databases (1)

- TID databases complete only with views
- How to deal with views in a PDBMS?
  1. Store just the view definition
  2. Store the view result and probabilities
  3. Store the view result and lineage
  4. Store the view results and "compiled lineage"
- Trade-off between precomputation and query cost (just as in DBMS)

Example (ExpensiveHotel view)

$$q(h) \leftarrow \exists n.\exists c.\text{Hotel}(h, n, c) \wedge \exists r.\exists t.\exists p.\text{Room}(r, h, t, p) \wedge (p > 500 \vee t = \text{'suite'})$$

Room (R)

RoomNo	Type	HotelNo	Price	
R1	Suite	H1	\$50	X_1
R2	Single	H1	\$600	X_2
R3	Double	H1	\$80	X_3

Hotel (H)

HotelNo	Name	City	
H1	Hilton	SB	X_4

ExpensiveHotels

HotelNo	
H1	0.375

(2)

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge (X_1 \vee X_2)$

(3)

ExpensiveHotels

HotelNo	
H1	$X_4 \wedge^{i} (X_1 \vee^{i} X_2)$

(4)

Materialized views in TID databases (2)

Example (Continued)

Hotel (H)

HotelNo	Name	City
H1	Hilton	SB

X_4

Consider the query

$$q(h) \leftarrow \exists c.\text{ExpensiveHotel}(h), \text{Hotel}(h, \text{'Hilton'}, c),$$

which asks for expensive Hilton hotels using a view. Can we answer this query when ExpensiveHotel is a precomputed materialized view?

ExpensiveHotels

HotelNo
H1

$X_4 \wedge (X_1 \vee X_2)$

Yes, combine lineages

ExpensiveHotels

HotelNo
H1

0.375

No, dependency between ExpensiveHotels and Hotels lost

ExpensiveHotels

HotelNo
H1

$X_4 \wedge^{i} (X_1 \vee^{i} X_2)$

Yes, combine "compiled lineages" → Need to be able to combine compiled lineages efficiently!

ExpensiveHiltons

HotelNo
H1

$[X_4 \wedge (X_1 \vee X_2)] \wedge X_4$

ExpensiveHiltons

HotelNo
H1

$X_4 \wedge^{i} (X_1 \vee X_2)$

Query compilation

- "Compile" Φ into a Boolean circuit with certain desirable properties
- $\mathbb{P}(\Phi)$ can be computed in linear time in the size of the circuit
  - Many other tasks can be solved in polynomial time
  - E.g., combining formulas $\Phi_1 \wedge \Phi_2$ (even when not independent!)
  - Key application in PDBMS: Compile materialized views
- Tractable compilation = circuit of size polynomial in database
  $\rightarrow$ Implies tractable computation of $\mathbb{P}(\Phi)$ (converse may not be true)
- Compilation targets
  1. RO (read-once formula)
  2. OBDD (ordered binary decision diagram)
  3. FBDD (free binary decision diagram)
  4. d-DNF (deterministic-decomposable normal form)

Goals: (1) Reusability. (2) Understand complexity of intensional QE.

Restricted Boolean circuit (RBC)

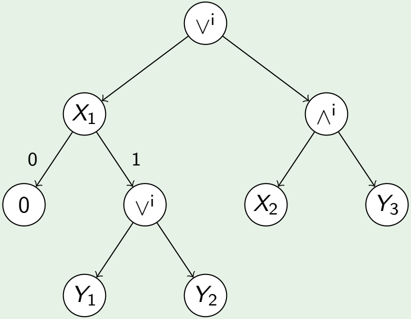
- Rooted, labeled DAG
- All variables are Boolean
- Each node (called *gate*) representents a propositional formula Ψ
- Let Ψ be represented by a gate with children representing $\Psi_1, \dots, \Psi_n$; we consider the following gates & restrictions:
  - *Independent-and* ($\wedge^{\mathrm{i}}$): $\Psi_1, \dots, \Psi_n$ are syntactically independent
  - *Independent-or* ($\vee^{\mathrm{i}}$): $\Psi_1, \dots, \Psi_n$ syntactically independent
  - *Disjoint-or* ($\vee^{\mathrm{d}}$): $\Psi_1, \dots, \Psi_n$ are disjoint
  - *Not* ($\neg$): single child, represents $\neg\Psi$
  - *Conditional gate* (X): two children representing $X \wedge \Psi_1$ and $\neg X \wedge \Psi_2$, where $X \notin \mathrm{Var}(\Psi_1)$ and $X \notin \mathrm{Var}(\Psi_2)$
  - *Leaf node* (0, 1, X): represents `FALSE`, `TRUE`, X

The different compilation targets restrict which and where gates may be used.

Restricted Boolean circuit (example)

Example

Who incriminates someone who has an alibi?
Lineage of unsafe plan: $\Phi_{\mathrm{M}} = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$



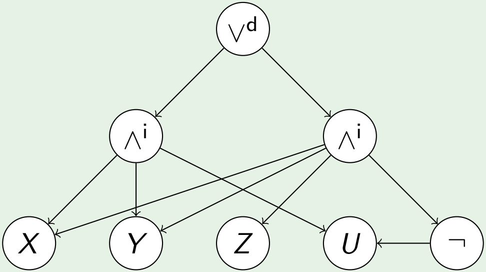
"Documents" the non-deterministic algorithm for intensional query evaluation.

Deterministic-decomposable normal form (d-DNF)

- Restricted to gates: $\wedge^{\mathsf{i}}$, $\vee^{\mathsf{d}}$, $\neg$
  - $\wedge^{\mathsf{i}}$-gates are called *decomposable* (D)
  - $\vee^{\mathsf{d}}$-gates are called *deterministic* (d)

Example

$\Phi = XYU \vee XYZ\neg U$



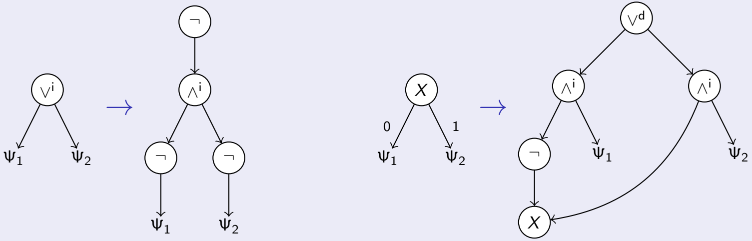
RBC and d-DNF

Theorem

Every RBC with n gates can be transformed into an equivalent d-DNF with at most $5n$ gates, a polynomial increase in size.

Proof.

We are not allowed to use $\vee^{\mathrm{i}}$ and conditional nodes. Apply the transformations:



A $\vee^{\mathrm{i}}$-node is replaced by 4 new nodes. A conditional node is replaced by (at most) 5 new nodes. □

Application: knowledge compilation

- Tries to deal with intractability of propositional reasoning
- Key idea
  1. Slow offline phase: Compilation into a target language
  2. Fast online phase: Answers in polynomial time

  → Offline cost amortizes over many online queries
- Key aspects
  - Succinctness of target language (d-DNF, FBDD, OBDD, ...)
  - Class of queries that can be answered efficiently once compiled (consistency, validity, entailment, implicants, equivalence, model counting, probability computation, ...)
  - Class of transformations that can be performed efficiently once compiled ($\wedge$, $\vee$, $\neg$, conditioning, forgetting, ...)
- How to pick a target language?
  1. Identify which queries/transformations are needed
  2. Pick the *most succinct* language

Which queries admit polynomial representation in which target language?

Darwiche and Marquis, 2002

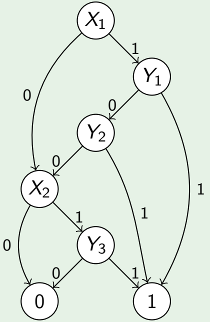
Free binary decision diagram (FBDD)

- Restricted to conditional gates
- *Binary decision diagram*: Each node decides on the value of a variable
- *Free*: Each variable occurs only on every root-leaf path

Example

Who incriminates someone who has an alibi?

Lineage of safe plan: $\Phi_{\mathrm{M}} = X_1(Y_1 \vee Y_2) \vee X_2 Y_3$



Ordered binary decision diagram (OBDD)

- An *ordered* FBDD, i.e.,
    - Same ordering of variables on each root-leaf path
    - Omissions are allowed

Example

The FBDD on slide 88 is an OBDD with ordering X_1, Y_1, Y_2, X_2, Y_3.

Theorem

Given two ODDBs Ψ_1 *and* Ψ_2 *with a common variable order, we can compute an ODDB for* $\Psi_1 \wedge \Psi_2$*,* $\Psi_1 \vee \Psi_2$*, or* $\neg\Psi_1$ *in polynomial time. Note that* Ψ_1 *and* Ψ_2 *do not need to be independent or disjoint.*

(Many other results of this kind exist. Many BDD software packages exist, e.g., BuDDy, JDD, CUDD, CAL).

Read-once formulas (RO)

Definition

A propositional formula Φ is read-once (or *repetition-free*) if there exists a formula Φ' such that $\Phi \equiv \Phi'$ and every variable occurs at most once in Φ'.

Example

- $\Phi = X_1 \vee X_2 \vee X_3$ → read-once
- $\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3 \vee X_2 Y_4 \vee X_2 Y_5$
  - $\Phi' = X_1(Y_1 \vee Y_2) \vee X_2(Y_3 \vee Y_4 \vee Y_5)$ → read-once
- $\Phi = XY \vee XU \vee YU$ → not read-once

Theorem

If Φ is given as a read-once formula, we can compute $\mathbb{P}(\Phi)$ in linear time.

Proof.

All $\wedge$'s and $\vee$'s are independent, and negation is easily handled. □

When is a formula read-once? (1)

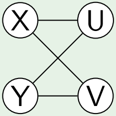
Definition

Let Φ be given in DNF such that no conjunct is a strict subset of some other conjunct. Φ is *unate* if every propositional variable X occurs either only positively or negatively. The *primal graph* $G(V, E)$ where V is the set of propositional variables in Φ and there is an edge $(X, Y) \in E$ if X and Y occur together in some conjunct.

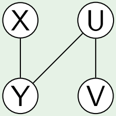
Example

- Unate: $XY \vee \neg ZX$
- Not unate: $XY \vee Z\neg X$

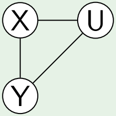
$XU \vee XV \vee YU \vee YV$



$XY \vee YU \vee UV$



$XY \vee XU \vee YU$



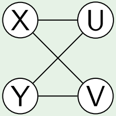
When is a formula read-once? (2)

Definition

A primal graph G for Φ is *P_4-free* if no induced subgraph is isomorphic to P_4 (○–○–○–○). G is *normal* if for every clique in G, there is a conjunct in Φ that contains all of the clique's variables.

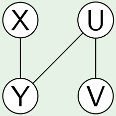
Example

$XU \vee XV \vee YU \vee YV$



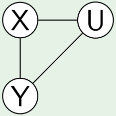
P_4-free
Normal
Read-once

$XY \vee YU \vee UV$



Not P_4-free
Normal
Not read-once

$XY \vee XU \vee YU$



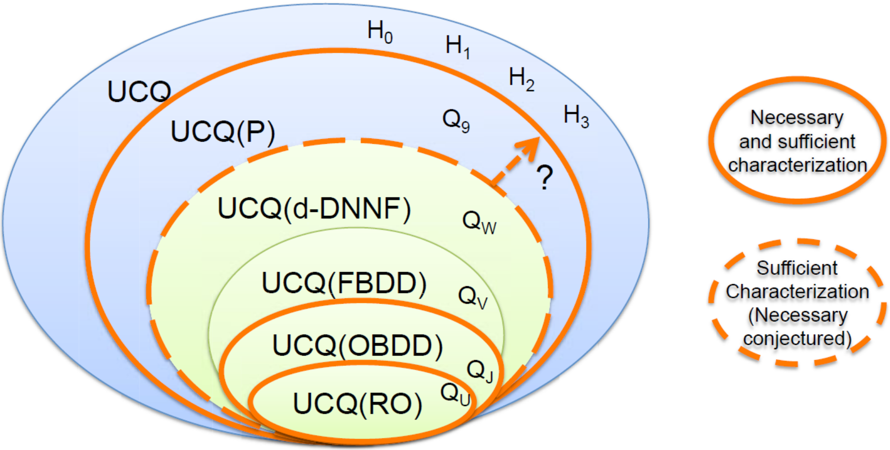
P_4-free
Not normal
Not read-once

Theorem

A unate formula is read-once iff it is P_4-free and normal.

Query compilation hierarchy

Denote by $\mathscr{L}(\mathscr{T})$ the class of queries from $\mathscr{L}$ that can be compiled efficiently to target $\mathscr{T}$. The following relationships hold for $\mathcal{UCQ}$-queries:



Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
    - Syntactic Independence
    - Six Simple Rules
    - Tractability and Completeness
    - Extensional Plans
4. Intensional Query Evaluation
    - Syntactic independence
    - 5 Simple Rules
    - Query Compilation
    - Approximation Techniques
5. Summary

Why approximation?

- Exact inference may require exponential time → expensive
- Often absolute probability values of little interest; ranking desired
  → Good approximations of $\mathbb{P}(\,\Phi\,)$ suffice
- Desiderata
  - (Provably) low approximation error
  - Efficient
  - Polynomial in database size
  - Anytime algorithm (gradual improvement)
- Approaches
  - Probability intervals
  - Monte-Carlo approximation

We will show: Approximation is tractable for all $\mathcal{RA}$-queries w.r.t. absolute error and for all $\mathcal{UCQ}$-queries w.r.t. relative error!

Probability bounds

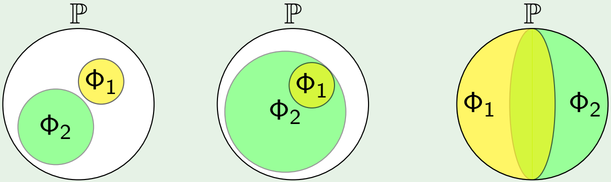
Theorem

Let Φ_1 and Φ_2 be propositional formulas. Then,

$$\begin{array}{rcccl} \max(\mathbb{P}(\Phi_1),\mathbb{P}(\Phi_2)) & \leq & \mathbb{P}(\Phi_1 \vee \Phi_2) & \leq & \overbrace{\min(\mathbb{P}(\Phi_1)+\mathbb{P}(\Phi_2),1)}^{\textit{Boole's inequality / union bound}} \\ \underbrace{\max(0,\mathbb{P}(\Phi_1)+\mathbb{P}(\Phi_2)-1)}_{\textit{via inclusion-exclusion}} & \leq & \mathbb{P}(\Phi_1 \wedge \Phi_2) & \leq & \min(\mathbb{P}(\Phi_1),\mathbb{P}(\Phi_2)). \end{array}$$

Example

Border cases:



$\mathbb{P}(\Phi_1 \vee \Phi_2)$	$\mathbb{P}(\Phi_1)+\mathbb{P}(\Phi_2)$	$\mathbb{P}(\Phi_2)$	1
$\mathbb{P}(\Phi_1 \wedge \Phi_2)$	0	$\mathbb{P}(\Phi_1)$	$\mathbb{P}(\Phi_1)+\mathbb{P}(\Phi_2)-1$

Computation of probability intervals

Theorem

Let Φ_1 and Φ_2 be propositional formulas with bounds $[L_1, U_1]$ and $[L_2, U_2]$, respectively. Then,

$$\begin{aligned}
\Phi_1 \lor \Phi_2: &\quad [L, U] = [\max(L_1, L_2), \min(U_1 + U_2, 1)] \\
\Phi_1 \land \Phi_2: &\quad [L, U] = [\max(0, L_1 + L_2 - 1), \min(U_1, U_2)] \\
\neg\Phi_1: &\quad [L, U] = [1 - U_1, 1 - L_1]
\end{aligned}$$

Example (Does Mary incriminate someone who has an alibi?)

$\Phi = X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3$

- $X_1 Y_1 : \; [0.75, 0.85]$
- $X_1 Y_2 : \; [0.65, 0.75]$
- $X_2 Y_3 : \; [0.45, 0.65]$
- $X_1 Y_1 \lor X_1 Y_2 \lor X_2 Y_3 : \; [0.75, 1]$

Incriminates

Witness	Suspect	$\mathbb{P}$	
Mary	Paul	0.9	X_1
Mary	John	0.8	X_2

Alibi

Suspect	Claim	$\mathbb{P}$	
Paul	Cinema	0.85	Y_1
Paul	Friend	0.75	Y_2
John	Bar	0.65	Y_3

Bounds can be computed in linear time in size of Φ.

Probability intervals and intensional query evaluation

```
if $\Phi = \Phi_1 \wedge \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
    return $[L, U] = [L_1 \cdot L_2, U_1 \cdot U_2]$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and $\Phi_1, \Phi_2$ are syntactically independent then
    return $[L, U] = [L_1 \oplus L_2, U_1 \oplus U_2]$
end if
if $\Phi = \Phi_1 \vee \Phi_2$ and $\Phi_1, \Phi_2$ are disjoint then
    return $[L, U] = [L_1 + L_2, \min(U_1 + U_2, 1)]$
end if
if $\Phi = \neg\Phi_1$ then
    return $[L, U] = [1 - U_1, 1 - L_1]$
end if
Choose $X \in \mathrm{Var}(\Phi)$
Shannon expansion to $\Phi = \bigvee_i \Phi_i \wedge (X = a_i)$
return $[L, U] = [\sum_i L_i \mathbb{P}(X = a_i), \min(\sum_i U_i \mathbb{P}(X = a_i), 1)]$
```

Independence and disjointness allow for tighter bounds.

Probability intervals and intensional query evaluation (2)

Example

Incriminates

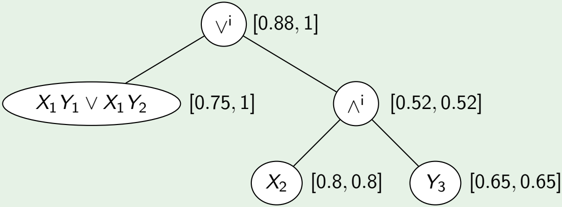
Witness	Suspect	$\mathbb{P}$	
Mary	Paul	0.9	X_1
Mary	John	0.8	X_2

Alibi

Suspect	Claim	$\mathbb{P}$	
Paul	Cinema	0.85	Y_1
Paul	Friend	0.75	Y_2
John	Bar	0.65	Y_3

$$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$$

- $X_1 Y_1 : \; [0.75, 0.85]$
- $X_1 Y_2 : \; [0.65, 0.75]$
- $X_2 Y_3 : \; [0.45, 0.65]$
- $\Phi : \; [0.75, 1]$



Discussion

- Incremental construction of RBC circuit
- If all leaf nodes are atomic, computes exact probability
- If some leaf nodes are not atomic, computes probability bounds
- Anytime algorithm (makes incremental progress)
- Can be stopped as soon as bounds become accurate enough
  - Absolute ϵ-approximation: $U - L \leq 2\epsilon \rightarrow$ choose $\hat{p} \in [U - \epsilon, L + \epsilon]$
  - Relative ϵ-approximation:
    $(1-\epsilon)U \leq (1+\epsilon)L \rightarrow$ choose $\hat{p} \in [(1-\epsilon)U, (1+\epsilon)L]$
- But: no apriori runtime bounds!

Definition

A value $\hat{p}$ is an *absolute ϵ-approximation* of $p = \mathbb{P}(\Phi)$ if

$$p - \epsilon \leq \hat{p} \leq p + \epsilon;$$

it is an *relative ϵ-approximation* of p if

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

Monte-Carlo approximation w/ naive estimator

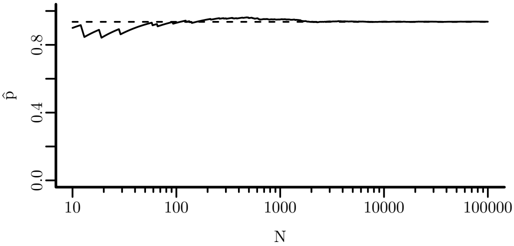
Let Φ be a propositional formula with $V(\Phi) = \{ X_1, \ldots, X_l \}$.

- Pick a value n and for $k \in \{ 1, 2, \ldots, n \}$, do
    1. Pick a random assignment θ_k by setting

$$X_i = \begin{cases} \texttt{TRUE} & \text{with probability } \mathbb{P}(X_i) \\ \texttt{FALSE} & \text{otherwise} \end{cases}$$

    2. Evaluate $Z_k = \Phi[\theta_k]$
- Return $\hat{p} = \frac{1}{n} \sum_k Z_k$

How good is this algorithm?



$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$

X_1	X_2	Y_1	Y_2	Y_3		Z_k	$\hat{p}$
1	1	1	1	1		1	1.00
1	0	1	1	0		1	1.00
1	0	0	0	1		0	0.66
1	0	1	1	1		1	0.75
1	1	1	0	1		1	0.80
1	1	1	1	1		1	0.83
1	1	1	0	0		1	0.85
1	1	1	1	1		1	0.88
1	1	1	1	1		1	0.89
1	1	1	1	1		1	0.90

Naive estimator: expected value

Theorem

The naive estimator $\hat{p}$ *is unbiased, i.e.,* $\mathbb{E}\,[\,\hat{p}\,] = \mathbb{P}\,(\,\Phi\,)$ *so that* $\hat{p}$ *is correct in expectation.*

Proof.

$$\begin{aligned}\mathbb{E}\,[\,\hat{p}\,] &= \mathbb{E}\,[\,\frac{1}{n}\sum_{k=1}^{n} Z_k\,] = \frac{1}{n}\sum_{k=1}^{n}\mathbb{E}\,[\,Z_k\,]\\ &= \mathbb{E}\,[\,Z_1\,]\\ &= \sum_{\theta}\Phi[\theta]\,\mathbb{P}\,(\,\theta\,)\\ &= \mathbb{P}\,(\,\Phi\,)\,.\end{aligned}$$

□

But: Is the actual estimate likely to be close to the expected value?

Chernoff bound (1)

Theorem (Two-sided Chernoff bound, simple form)

Let $Z_1, \ldots, Z_n$ be i.i.d. 0/1 random variables with $\mathbb{E}[Z_1] = p$ and set $\bar{Z} = \frac{1}{n}\sum_k Z_k$. Then,

$$\mathbb{P}\left(\left|\bar{Z} - p\right| \geq \gamma p\right) \leq 2\exp\left(-\frac{\gamma^2}{2+\gamma} pn\right)$$

In words:

- Take a coin with (unknown) probability of heads p (thus tail $1 - p$)
- Flip the coin n times: outcomes $Z_1, \ldots, Z_n$
- Compute the fraction $\bar{Z}$ of heads
- Estimate p using $\bar{Z}$
- Then: Probability that relative error larger than γ
  1. Decreases exponentially with increasing number of flips n
  2. Decreases with increasing error bound γ
  3. Decreases with increasing probability of heads p

Very important result with many applications!

Chernoff bound (2)

Theorem (Two-sided Chernoff bound, simple form)

Let $Z_1, \ldots, Z_n$ *be i.i.d. 0/1 random variables with* $\mathbb{E}[Z_1] = p$ *and set* $\bar{Z} = \frac{1}{n}\sum_k Z_k$. *Then,*

$$\mathbb{P}\left(\left|\bar{Z} - p\right| \geq \gamma p\right) \leq 2\exp\left(-\frac{\gamma^2}{2+\gamma}pn\right)$$

Proof (outline).

We give the first steps of the proof of the one-sided Chernoff bound. First,

$$\mathbb{P}(Z \geq q) = \mathbb{P}(e^{tZ} \geq e^{tq}).$$

for any $t > 0$. Use the Markov inequality $\mathbb{P}(|X| \geq a) \leq \mathbb{E}[|X|]/a$ to obtain

$$\begin{aligned}\mathbb{P}(Z \geq q) &\leq \mathbb{E}[e^{tZ}]/e^{tq}\\ &= \mathbb{E}[e^{tZ_1} \cdots e^{tZ_n}]/e^{tq} = \mathbb{E}[e^{tZ_1}] \cdots \mathbb{E}[e^{tZ_n}]/e^{tq} = \mathbb{E}[e^{tZ_1}]^n/e^{tq}\end{aligned}$$

Use definition of expected value and find the value of t that minimizes RHS to obtain the precise one-sided Chernoff bound. Relax the RHS to obtain the simple form. ☐

Naive estimator: absolute (ϵ,δ)-approximation (1)

Theorem (sampling theorem)

To obtain an absolute ϵ-approximation with probability at least $1 - \delta$, it suffices to run

$$n \geq \frac{2+\epsilon}{\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{\epsilon^2} \ln \frac{1}{\delta}\right)$$

sampling steps.

Proof.

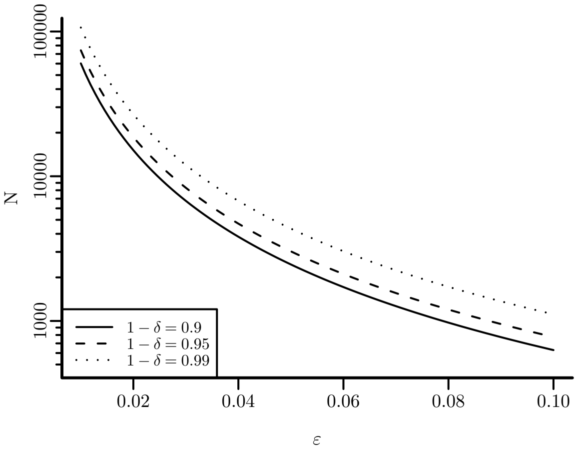
Take $\gamma = \epsilon/p$ and apply the Chernoff bound to obtain

$$\begin{aligned} \mathbb{P}\left(\left|\bar{Z} - p\right| \geq \epsilon \right) &\leq 2\exp\left(-\frac{\epsilon^2/p^2}{2+\epsilon/p} pn\right) = 2\exp\left(-\frac{\epsilon^2}{2p+\epsilon} n\right) \\ &\leq 2\exp\left(-\frac{\epsilon^2}{2+\epsilon} n\right) \end{aligned}$$

since $p \leq 1$. Now solve RHS $\leq \delta$ for n. ☐

Naive estimator: absolute (ϵ,δ)-approximation (2)

The number of sampling steps given by the sampling theorem is independent of Φ.



Naive estimator: relative (ϵ,δ)-approximation (1)

Theorem

To obtain a relative ϵ-approximation with probability at least $1 - \delta$, it suffices to run

$$n \geq \frac{2+\epsilon}{p\epsilon^2} \ln \frac{2}{\delta} = O\left(\frac{1}{p\epsilon^2} \ln \frac{1}{\delta}\right)$$

sampling steps.

Proof.

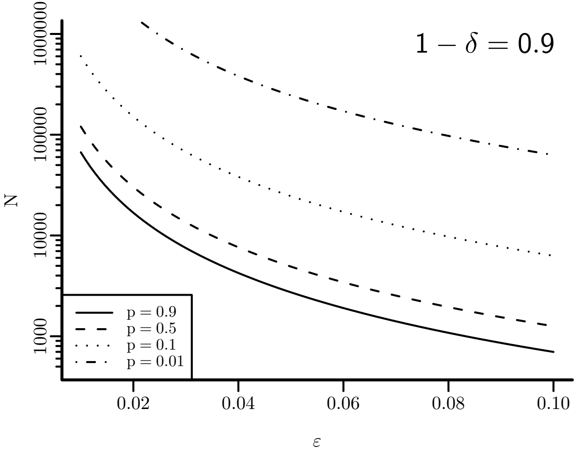
Take $\gamma = \epsilon$ and apply the Chernoff bound to obtain

$$\mathbb{P}\left(\left|\bar{Z} - p\right| \geq \epsilon p\right) \leq 2 \exp\left(-\frac{\epsilon^2}{2+\epsilon} pn\right)$$

Now solve RHS $\leq \delta$ for n. □

Naive estimator: relative (ϵ,δ)-approximation (2)

The number of sampling steps given by the sampling theorem now is dependent on Φ; we cannot compute the number of required steps in advance! Obtaining small relative error for small *p* (i.e., Φ is often false) requires a large number of sampling steps.



Why care about relative ϵ-approximation?

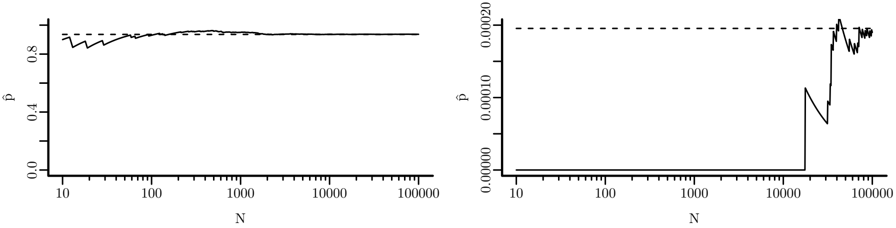
1. Absolute error ill-suited to compare estimates of small probabilities
    - $p_1 = 0.001$, $p_2 = 0.01$, $\epsilon = 0.1$
    - Absolute error: $I_1 = [0, 0.101]$, $I_2 = [0, 0.11]$
    - Relative error: $I_1 = [0.0009, 0.0011]$, $I_2 = [0.009, 0.011]$

    $\rightarrow$ Ranking of tuples more sensitive to absolute error
2. For $p \in [0, 1)$, relative error ϵ is *always tighter* than absolute error ϵ (esp. when probabilities are small)

Can we get a relative ϵ-approximation in which the minimum number of sampling steps does not depend on $\mathbb{P}(\Phi)$?

The problem with the naive estimator

$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$



Large probabilities

Small probabilities ($\times 10^{-2}$)

- When $\mathbb{P}(\Phi)$ is small, Φ not satisfied on most samples
  → Slow convergence

Idea: Change the sampling strategy so that Φ is satisfied on every sample.

Karp-Luby estimator (basic idea)

Let Φ be a propositional *DNF formula* with $V(\Phi) = \{ X_1, \ldots, X_l \}$, i.e.,

$$\Phi = C_1 \vee C_2 \vee \cdots \vee C_m.$$

Easy to find satisfying assignments!

Set $q_i = \mathbb{P}(C_i)$ and $Q = \sum_i q_i$. Note that $p \leq Q$ (union bound).

$$\begin{aligned}
\mathbb{P}(\Phi) &= \mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \wedge C_2) + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \wedge C_m) \\
&= \mathbb{P}(\mathtt{TRUE} \mid C_1) \mathbb{P}(C_1) + \mathbb{P}(\neg C_1 \mid C_2) \mathbb{P}(C_2) + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \mid C_m) \mathbb{P}(C_m) \\
&= Q \, [\mathbb{P}(\mathtt{TRUE} \mid C_1) q_1/Q + \mathbb{P}(\neg C_1 \mid C_2) q_2/Q + \cdots \\
&\qquad + \mathbb{P}(\neg(C_1 \vee \cdots \vee C_{m-1}) \mid C_m) q_m/Q]
\end{aligned}$$

Idea of Karp-Luby estimator:

1. q_i/Q is computed exactly (in linear time)
2. $\mathbb{P}(\neg(C_1 \vee \cdots \vee C_{i-1}) \mid C_i)$ are estimated
   - Impact of estimate proportional to $\mathbb{P}(C_i)$

     → Focus on clauses with highest probability

Karp-Luby estimator

- Pick a value n and for $k \in \{1, 2, \ldots, n\}$, do
  1. Pick a random clause C_i (with probability q_i/Q)
  2. Pick a random assignment θ_k
     - ★ For a variable $X \in V(C_i)$

$$X = \begin{cases} \texttt{TRUE} & \text{if } X \text{ is positive in } C_i \\ \texttt{FALSE} & \text{if } X \text{ is negative in } C_i \end{cases}$$

       → Clause C_i is satisfied (and thus Φ)
     - ★ For the other variables $X \notin V(C_i)$

$$X = \begin{cases} \texttt{TRUE} & \text{with probability } \mathbb{P}(X) \\ \texttt{FALSE} & \text{otherwise} \end{cases}$$

       → All other variables take random values
  3. Evaluate

$$Z_k = \begin{cases} 1 & \text{if } \neg(\bigvee_{1 \leq j < i} C_j[\theta]) \\ 0 & \text{otherwise} \end{cases}$$

- Return $\hat{p} = \dfrac{Q}{n} \displaystyle\sum_{k=1}^{n} Z_k$

Example of KL estimator

$\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$

- $m = 3$, probabilities of X_1 and Y_3 reduced to 1/10th
- $C_1 = X_1 Y_1, \quad q_1 = 0.09 \cdot 0.85 = 0.0765, \quad q_1/Q \approx 0.39$
- $C_2 = X_1 Y_2, \quad q_2 = 0.09 \cdot 0.75 = 0.0675, \quad q_2/Q \approx 0.34$
- $C_3 = X_2 Y_3, \quad q_3 = 0.8 \cdot 0.065 = 0.052, \quad q_3/Q \approx 0.27$
- $Q = 0.196$, $p \approx 0.134$

i		X_1	X_2	Y_1	Y_2	Y_3		C_1	C_2	C_3		Z_k	$\hat{p}$
1		1	1	1	1	0		1	1	0		1	0.196
3		0	1	1	1	1		0	0	1		1	0.196
2		1	1	1	1	0		1	1	0		0	0.131
1		1	1	1	1	0		1	1	0		1	0.147
1		1	1	1	1	0		1	1	0		1	0.157
2		1	0	1	1	0		1	1	0		0	0.131

KL estimator: expected value

Theorem

The KL estimator $\hat{p}$ is unbiased, i.e., $\mathbb{E}[\,\hat{p}\,] = \mathbb{P}(\,\Phi\,)$ so that $\hat{p}$ is correct in expectation.

Proof.

$$\begin{aligned}
\mathbb{E}[\,\hat{p}\,] &= \mathbb{E}[\,\frac{Q}{n}\sum_{k=1}^{n} Z_k\,] = Q\,\mathbb{E}[\,Z_1\,] = Q\,\mathbb{E}[\,\mathbb{E}[\,Z_1 \mid C_i \text{ picked}\,]\,] \\
&= Q\sum_{i=1}^{m}\frac{q_i}{Q}\,\mathbb{E}[\,Z_1 \mid C_i \text{ picked}\,] \\
&= \sum_{i=1}^{m}\mathbb{P}(\,C_i\,)\,\mathbb{P}(\neg \bigvee_{1\le j<i} C_j \mid C_i\,) \\
&= \mathbb{P}(\,\Phi\,)\,.
\end{aligned}$$

□

KL estimator: relative (ϵ,δ)-approximation

Theorem

To obtain a relative ϵ-approximation with probability at least $1-\delta$, it suffices to run

$$n \geq m\frac{2+\epsilon}{\epsilon^2}\ln\frac{2}{\delta} = O\left(\frac{m}{\epsilon^2}\ln\frac{1}{\delta}\right)$$

sampling steps of the KL estimator.

Proof.

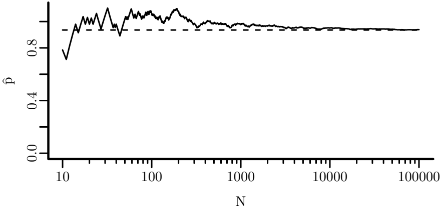
Use the Chernoff bound with $\gamma = \epsilon$ and $\mathbb{E}\left[\bar{Z}\right] = Q^{-1}p$.

$$\begin{aligned}
\mathbb{P}\left(\left|\bar{Z} - Q^{-1}p\right| \geq \epsilon Q^{-1}p\right) &\leq 2\exp\left(-\epsilon^2/(2+\epsilon)Q^{-1}pn\right)\\
\mathbb{P}\left(\left|Q^{-1}\hat{p} - Q^{-1}p\right| \geq \epsilon Q^{-1}p\right) &= \mathbb{P}\left(\left|\hat{p}-p\right| \geq \epsilon p\right)\\
&\leq 2\exp\left(-\epsilon^2/(2+\epsilon)m^{-1}n\right),
\end{aligned}$$

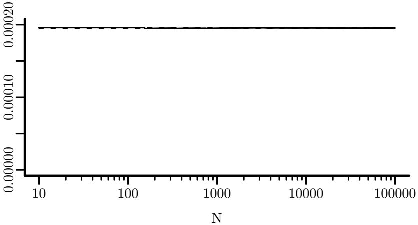
since $mp \geq Q$. Now solve RHS $\leq \delta$ for n. □

KL estimator: discussion

- KL estimator provides relative (ϵ,δ)-approximation in polynomial time in *size* of Φ and $\frac{1}{\epsilon}$
  →*fully polynomial-time randomized approximation scheme* (*FPTRAS*)
- Example: $\Phi = X_1 Y_1 \vee X_1 Y_2 \vee X_2 Y_3$



Large probabilities



Small probabilities ($\times 10^{-2}$)

- Requires DNF (=why-provenance; polynomial in DB size for $\mathcal{UCQ}$)

For ϵ, δ fixed and relative error, the naive estimator requires $O(p^{-1})$ sampling steps and the KL estimator requires $O(m)$ steps. In general, the naive estimator is preferable when the DNF is very large. The KL estimator preferable if probabilities are small.

Outline

1. Primer: Relational Calculus
2. The Query Evaluation Problem
3. Extensional Query Evaluation
    - Syntactic Independence
    - Six Simple Rules
    - Tractability and Completeness
    - Extensional Plans
4. Intensional Query Evaluation
    - Syntactic independence
    - 5 Simple Rules
    - Query Compilation
    - Approximation Techniques
5. Summary

Lessons learned

- Relational calculus is a great tool for query analysis & manipulation
- Query evaluation computes marginal probabilities $\mathbb{P}(\, t \in q(\mathcal{D})\,)$
- On tuple-independent DBs and $\mathcal{UCQ}$, data complexity either P or #P
- Extensional query evaluation
  - Detects and evaluates the subset of safe queries (P)
  - Leverages query structure to obtain polynomial-time algorithm
  - Uses $\mathbf{R}_6$-rules to create an extensional plan that can be executed in an (extended) RDBMS → highly scalable
  - Rules are sound and complete for $\mathcal{UCQ}$
- Intensional query evaluation
  - Applies to all queries, but focus is on hard (sub)queries
  - Ignores query structure, leverages data properties
  - Computes probabilities of propositional lineage formulas
  - Rule-based evaluation computes probabilities precisely, but potentially exponential blow-up → stop early to obtain probability bounds
  - Sampling techniques apply to all formulas; FPTRAS for $\mathcal{UCQ}$
- Hybrids of extensional and intensional query evaluation promising

Suggested reading

- Serge Abiteboul, Richard Hull, Victor Vianu
  Foundations of Databases: The Logical Level (ch. 12)
  Addison Wesley, 1994

- Dan Sucio, Dan Olteanu, Christopher Ré, Christoph Koch
  Probabilistic Databases (ch. 3–5)
  Morgan&Claypool, 2011

- Michael Mitzenmacher, Eli Upfal
  Probability and Computing: Randomized Algorithms and Probabilistic Analysis (ch. 10)
  Cambridge University Press, 2005

Scalable Uncertainty Management

06 – Markov Logic

Rainer Gemulla

July 13, 2012

Overview

In this lecture

- Statistical relational learning (SRL)
- Introduction to probabilistic graphical models (PGM)
- Basics of undirected models (called *Markov networks*)
- Markov logic as a template for undirected models
- Basics of inference in Markov logic networks

Not in this lecture

- Directed models (called *Bayesian networks*)
- Other SRL approaches (such as *probabilistic relational models*)
- High coverage and in-depth discussion of inference
- Learning Markov logic networks

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Correlations in probabilistic databases

- Simple probabilistic models
  - Tuple-independent databases
  - Block-disjoint independent databases
  - Key/foreign key constraints, ...
- Correlations (mainly) through $\mathcal{RA}$ queries/views
  - Any discrete probability distribution can be modeled
  - Queries describe *precisely* how result is derived

Example (Nell)

NellExtraction

Subject	Pattern	Object	Source	$\mathbb{P}$
Sony	produces	Walkman	1	0.96
IBM	produces	PC	1	0.96
IBM	produces	PC	2	1
Microsoft	produces	MacOS	2	0.9
AlbertEinstein	bornIn	Ulm	1	0.9

NellSource

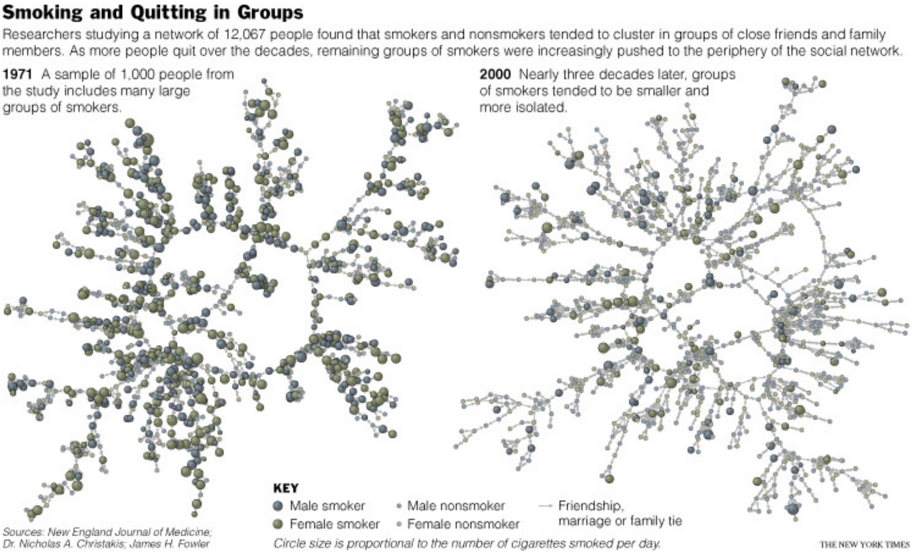
Source	$\mathbb{P}$
1	0.99
2	0.1

$$\text{Produces}(x, y) \leftarrow \text{NellExtraction}(x, \text{'produces'}, y, s), \text{NellSource}(s)$$

Produces

Subject	Object	$\mathbb{P}$
Sony	Walkman	0.9504
IBM	PC	0.95536
Microsoft	MacOS	0.09

Statistical relational learning (I)



Smoking and Quitting in Groups

Researchers studying a network of 12,067 people found that smokers and nonsmokers tended to cluster in groups of close friends and family members. As more people quit over the decades, remaining groups of smokers were increasingly pushed to the periphery of the social network.

1971 A sample of 1,000 people from the study includes many large groups of smokers.

2000 Nearly three decades later, groups of smokers tended to be smaller and more isolated.

Circle size is proportional to the number of cigarettes smoked per day.

Sources: New England Journal of Medicine; Dr. Nicholas A. Christakis; James H. Fowler

THE NEW YORK TIMES

Does John smoke?

Learn correlations from structured data, then apply to new data.

Statistical relational learning (II)

- Goal: Declarative modelling of correlations in structured data
- Idea: Use (subsets of) first-order logic
  - Very expressive formalism; lots of knowledge bases use it
  - Symmetry: $\forall x. \forall y. \text{Friends}(x, y) \iff \text{Friends}(y, x)$
  - Everybody has a friend: $\forall x. \exists y. \text{Friends}(x, y)$
  - Transitivity: $\forall x. \forall y. \forall z. \text{Friends}(x, y) \wedge \text{Friends}(y, z) \implies \text{Friends}(x, z)$
  - Smoking causes cancer: $\forall x. \text{Smokes}(x) \implies \text{Cancer}(x)$
  - Friends have similar smoking habits: $\forall x. \forall y. \text{Friends}(x, y) \implies (\text{Smokes}(x) \iff \text{Smokes}(y))$
- Problem: Real-world knowledge is incomplete, contradictory, complex $\rightarrow$ Above rules do *not* generally hold, but they are "likely" to hold!
- Approach: Combine first-order logic with probability theory
  - Expressiveness of first-order logic
  - Principled treatment of uncertainty using probability theory

There are many approaches of this kind. Our focus is on *Markov logic*, a recent and very successful language.

Markov logic networks

Definition

A *Markov logic network* is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and the weight w_i is a real number.

Example

$$1.5 \begin{cases} \text{Smoking causes cancer} \\ \forall x.\mathrm{Smokes}(x) \implies \mathrm{Cancer}(x) \end{cases}$$

$$1.1 \begin{cases} \text{Friends have similar smoking habits} \\ \forall x.\forall y.\mathrm{Friends}(x,y) \implies (\mathrm{Smokes}(x) \iff \mathrm{Smokes}(y)) \end{cases}$$

- Formulas may or may not hold
- Weights express confidence
  - High positive weight → confident that formula holds
  - High negative weight → confident that formula does not hold
  - But careful: weights actually express confidence of certain "groundings" of a formula and *not* the formula as a whole (more later)
- Formulas may introduce complex correlations

Simple MLN for entity resolution

Which citations refer to the same publication?

author	Richardson, Matt and Domingos, Pedro	M. Richardson and P. Domingos	Domingos, Pedro and Richardson, Matthew
title	Markov Logic Networks	Markov logic networks	Markov Logic: A Unifying Framework for Statistical Relational Learning
year	2006	2006	2007

```
// predicates
HasToken(token, field, citation)         // e.g., HasToken('Logic', 'title ', C1)
SameField(field, citation, citation)     // Semantic equality of values in a field
SameCitation(citation, citation)         // Semantic equality of citations

// formulas
HasToken(+t, +f, c1) ^ HasToken(+t, +f, c2) => SameField(+f, c1, c2)
SameField(+f, c1, c2) => SameCitation(c1, c2)
SameCitation(c1, c2) ^ SameCitation(c2, c3) => SameCitation(c1, c3)
```

Rule weights are usually learned from data. The same rule may have different weights for different constants (indicated by "+").

Alchemy

- Alchemy is well-known software package for Markov logic
- Developed at University of Washington
- Supports a wide range of tasks
  - Structure learning
  - Weight learning
  - Probabilistic inference
- Has been used for wide range of applications
  - Information extraction
  - Social network modeling
  - Entity resolution
  - Collective classification
  - Link prediction
- Check out `http://alchemy.cs.washington.edu/`
  - Code
  - Real-world datasets
  - Real-world Markov logic networks
  - Literature

From Markov logic to graphical models (example)

Friends

Name1	Name2	Value
Anna	Bob	Yes
Bob	Anna	Yes
Anna	Anna	Yes
Bob	Bob	Yes

Smokes

Name	Value
Anna	Yes

Cancer

Name	Value
Anna	No

$$1.5 \left\{ \begin{array}{l} \text{Smoking causes cancer} \\ \forall x. \text{Smokes}(x) \implies \text{Cancer}(x) \end{array} \right.$$

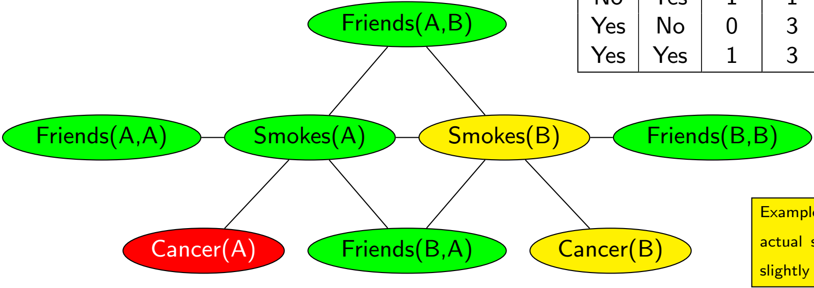
$$1.1 \left\{ \begin{array}{l} \text{Friends have similar smoking habits} \\ \forall x. \forall y. \text{Friends}(x, y) \implies (\text{Smokes}(x) \iff \text{Smokes}(y)) \end{array} \right.$$

Inference result

- $\mathbb{P}$ (Bob smokes) $= 84.6\%$
- $\mathbb{P}$ (Bob has cancer) $= 76.9\%$

Inference (conceptual)

S(B)	C(B)	#R1	#R2	$\sum w$	$\mathbb{P}$
No	No	1	1	2.6	7.7%
No	Yes	1	1	2.6	7.7%
Yes	No	0	3	3.3	15.4%
Yes	Yes	1	3	4.8	69.2%



Example is simplified; actual semantics are slightly different.

Probabilistic databases and graphical models

	Probabilistic databases	**Graphical models**
Probabilistic model	Simple (disjoint-independent tuples)	Complex (independencies given by graph)
Query	Complex (e.g., $\exists x.\exists y.R(x,y) \wedge S(x)$)	Simple (e.g., $\mathbb{P}(\, X_1, X_2 \mid Z_1, Z_2, Z_3 \,)$)
Network	Dynamic (database + query)	Static (Bayesian or Markov network)
Complexity measured in size of	Database	Network
Complexity parameter	Query	Treewidth
System	Extension to RDBMS	Stand-alone

Hybrid approaches have many potential applications and are under active research.

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Reasoning with uncertainty

- Goal: Automated reasoning system
  - Take all available information
    (e.g., patient information: symptoms, test results, personal data)
  - Reach conclusions
    (e.g., which diseases the patient has, which medication to give)
- Desiderata
  1. Separation of knowledge and reasoning
     - Declarative, model-based representation of knowledge
     - General suite of reasoning algorithms, applicable to many domains
  2. Principled treatment of uncertainty
     - Partially observed data
     - Noisy observations
     - Non-deterministic relationships
- Lots of applications
  - medical diagnosis, fault diagnosis, analysis of genetic and genomic data, communication and coding, analysis of marketing data, speech recognition, *natural language understanding*, segmenting and denoising images, social network analysis, . . .

Probabilistic models

- Multiple interrelated aspects may relate to the reasoning task
  - Possible diseases
  - Hundreds of symptoms and diagnostic tests
  - Personal characteristics

1. Characterize data by a set of random variables
   - Flu (yes / no)
   - Hayfever (yes / no)
   - Season (Spring / Sommer / Autumn / Winter)
   - Congestion (yes / no)
   - MusclePain (yes / no)

   → Variables and their domain are important design decision
2. Model dependencies by a *joint distribution*
   - Diseases, season, and symptoms are correlated
   - Probabilistic models construct joint probability space
     → $2 \cdot 2 \cdot 4 \cdot 2 \cdot 2$ outcomes (64 values, 63 non-redundant)
   - Given joint probability space, interesting questions can be answered

$$\mathbb{P}\,(\,\text{Flu} \mid \text{Season}{=}\text{Spring}, \text{Congestion}, \neg\text{MusclePain}\,)$$

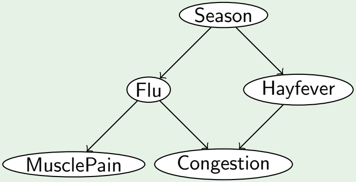
Specifying a joint distribution is infeasible in general!

Probabilistic graphical models

- A graph-based representation of *direct* probabilistic interactions
- A break-down of high-dimensional distributions into smaller *factors* (here: 63 vs. 17 non-redundant parameters)
- A compact representation of a set of (conditional) *independencies*

Example (directed graphical model)

Graph representation



Factorization

$$\mathbb{P}(\,S, F, H, M, C\,) \\ = \mathbb{P}(\,S\,)\,\mathbb{P}(\,F \mid S\,)\,\mathbb{P}(\,H \mid S\,)\,\mathbb{P}(\,C \mid F, H\,)\,\mathbb{P}(\,M \mid F\,)$$

Independencies

$$(F \perp H \mid S), (C \perp S \mid F, H), (M \perp H, C, S \mid F)$$

Main components

1. Representation
    - Tractability
        - ★ Variables tend to interact *directly* only with very few others
        - ★ Natural and compact encoding as graphical model
    - Transparency
        - ★ Models can be understood/evaluated by human experts
2. Inference
    - Answer queries using the distribution as model of the world
    - Work on graph structure
      → orders of magnitude faster than working on joint probability
3. Learning
    - Learn a model from data that captures past experience to a good approximation
    - Human experts may provide rough guidance
    - Details filled in by fitting the model to the data
      → Often better reflection of domain than hand-constructed models, sometimes surprising insights

Graphical models exploit locality structure that appears in many distributions that arise in practice.

Outline

Notation

Let $\mathbf{X}$ and $\mathbf{Y}$ be sets of random variables with domain $\mathrm{Dom}(\mathbf{X})$ and $\mathrm{Dom}(\mathbf{Y})$. Let $\mathbf{x} \in \mathrm{Dom}(\mathbf{X})$ and $\mathbf{y} \in \mathrm{Dom}(\mathbf{Y})$.

Expression	Shortcut notation
$\mathbb{P}(\mathbf{X} = \mathbf{x})$	$\mathbb{P}(\mathbf{x})$
$\mathbb{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y})$	$\mathbb{P}(\mathbf{x} \mid \mathbf{y})$
$\forall \mathbf{x}.\, \mathbb{P}(\mathbf{X} = \mathbf{x}) = f(\mathbf{x})$	$\mathbb{P}(\mathbf{X}) = f(\mathbf{X})$
$\forall \mathbf{x}. \forall \mathbf{y}.\, \mathbb{P}(\mathbf{X} = \mathbf{x} \mid \mathbf{Y} = \mathbf{y}) = f(\mathbf{x}, \mathbf{y})$	$\mathbb{P}(\mathbf{X} \mid \mathbf{Y}) = f(\mathbf{X}, \mathbf{Y})$

- $\mathbb{P}(\mathbf{X})$ and $\mathbb{P}(\mathbf{X} \mid \mathbf{Y})$ are entire probability distributions
- Can be thought of as functions from $\mathrm{Dom}(\mathbf{X}) \to [0, 1]$ or $(\mathrm{Dom}(\mathbf{X}), \mathrm{Dom}(\mathbf{Y})) \to [0, 1]$, respectively
- $f_{\mathbf{y}}(\mathbf{X}) = \mathbb{P}(\mathbf{X} \mid \mathbf{y})$ is often referred to as *conditional probability distribution* (CPD)
- For discrete variables, may be represented as a table (CPT)

Conditional independence

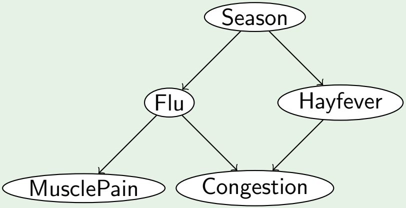
Definition

Let $\mathbf{X}, \mathbf{Y}$ and $\mathbf{Z}$ be sets of random variables. $\mathbf{X}$ and $\mathbf{Y}$ are said to be *conditionally independent* given $\mathbf{Z}$ if and only if

$$\mathbb{P}(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) = \mathbb{P}(\mathbf{X} \mid \mathbf{Z})\, \mathbb{P}(\mathbf{Y} \mid \mathbf{Z}).$$

We write $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$ for this conditional independence statement. If $\mathbf{Z} = \emptyset$, we write $(\mathbf{X} \perp \mathbf{Y})$ for *marginal independence*.

Example



$(F \perp H \mid S), (C \perp S \mid F, H)$
$(M \perp H, C, S \mid F)$

$$\mathbb{P}(S, F, H, M, C) \\ = \mathbb{P}(S) \cdot \mathbb{P}(F \mid S) \cdot \mathbb{P}(H \mid S) \\ \cdot \mathbb{P}(C \mid F, H) \cdot \mathbb{P}(M \mid F)$$

Properties of conditional independence

Theorem

In general, $(\mathbf{X} \perp \mathbf{Y})$ *does not imply nor is implied by* $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$

The following relationships hold:

$$\begin{aligned}
(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) &\iff (\mathbf{Y} \perp \mathbf{X} \mid \mathbf{Z}) && (\textit{symmetry})\\
(\mathbf{X} \perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) &\implies (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) && (\textit{decomposition})\\
(\mathbf{X} \perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) &\implies (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) && (\textit{weak union})\\
(\mathbf{X} \perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \wedge (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) &\implies (\mathbf{X} \perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) && (\textit{contraction})
\end{aligned}$$

For positive distributions and mutally disjoint sets $\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{W}$*:*

$$(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}, \mathbf{W}) \wedge (\mathbf{X} \perp \mathbf{W} \mid \mathbf{Z}, \mathbf{Y}) \implies (\mathbf{X} \perp \mathbf{Y}, \mathbf{W} \mid \mathbf{Z}) \quad (\textit{intersection})$$

Proof.

Discussed in exercise group. □

Querying a distribution (1)

Consider a joint distribution on a set of variables $\mathcal{X}$

- Let $\mathbf{E} \subseteq \mathcal{X}$ be a set of *evidence variables* that takes values $\mathbf{e}$
- Let $\mathbf{W} = \mathcal{X} - \mathbf{E}$ be the set of *latent variables*
- Let $\mathbf{Y} \subseteq \mathbf{W}$ be a set of *query variables*
- Let $\mathbf{Z} = \mathbf{W} - \mathbf{Y}$ be the set of *non-query variables*

Example

- $\mathcal{X} = \{\, \text{Season}, \text{Congestion}, \text{MusclePain}, \text{Flu}, \text{Hayfever} \,\}$
- $\mathbf{E} = \{\, \text{Season}, \text{Congestion}, \text{MusclePain} \,\}$
- $\mathbf{e} = \{\, \text{Spring}, \text{Yes}, \text{No} \,\}$
- $\mathbf{W} = \{\, \text{Flu}, \text{Hayfever} \,\}$
- $\mathbf{Y} = \{\, \text{Flu} \,\}$
- $\mathbf{Z} = \{\, \text{Hayfever} \,\}$

Querying a distribution (2)

1. Conditional probability query
   - Compute the *posterior distribution* of the query variables $\mathbb{P}(\mathbf{Y} \mid \mathbf{e})$
2. MAP query
   - Compute the most likely value of the latent variables $\mathrm{MAP}(\mathbf{W} \mid \mathbf{e}) = \mathrm{argmax}_{\mathbf{w}}\, \mathbb{P}(\mathbf{w} \mid \mathbf{e}) = \mathrm{argmax}_{\mathbf{w}}\, \mathbb{P}(\mathbf{w}, \mathbf{e})$
3. Marginal MAP query
   - Compute the most likely value of the query variables $\mathrm{MAP}(\mathbf{Y} \mid \mathbf{e}) = \mathrm{argmax}_{\mathbf{y}}\, \mathbb{P}(\mathbf{y} \mid \mathbf{e}) = \mathrm{argmax}_{\mathbf{y}} \sum_{\mathbf{z}} \mathbb{P}(\mathbf{y}, \mathbf{z}, \mathbf{e})$

Example

$\mathbb{P}(\mathbf{W} \mid \mathbf{e})$	Flu	¬Flu
Hayfever	5%	35%
¬Hayfever	40%	20%

1. $\mathbb{P}(\mathrm{Flu} \mid \mathrm{Spring}, \mathrm{Congestion}, \neg\mathrm{MusclePain}) \rightarrow$ Yes (45%), No (55%)
2. $\mathrm{MAP}(\mathrm{Flu}, \mathrm{Hayfever} \mid \mathrm{Spring}, \mathrm{Congestion}, \neg\mathrm{MusclePain}) \rightarrow$ Only flu
3. $\mathrm{MAP}(\mathrm{Flu} \mid \mathrm{Spring}, \mathrm{Congestion}, \neg\mathrm{MusclePain}) \rightarrow$ No flu (!)

Querying graphical models

- Graphical models induce conditional independences
- Queries reason about dependencies between variables

Can we evaluate queries more efficiently given a graphical model and its associated independences?

Example

Independence properties help inference!

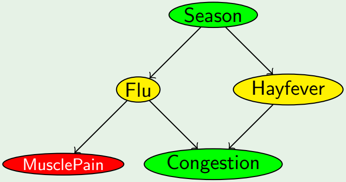


Table known to satisfy $(F \perp H \mid \mathbf{E})$

$\mathbb{P}(\mathbf{W} \mid \mathbf{e})$	Flu	¬Flu	
Hayfever	24%	16%	40%
¬Hayfever	36%	24%	60%
	60%	40%	

Thus, for example, monotonicity is now known to hold for MAP:

$$\mathrm{MAP}(\mathrm{Flu}, \mathrm{Hayfever} \mid \mathbf{E}) = (\mathrm{MAP}(\mathrm{Flu} \mid \mathbf{E}), \mathrm{MAP}(\mathrm{Hayfever} \mid \mathbf{E}))$$

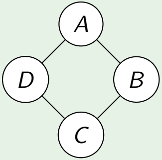
Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Misconception example

Example

- Alice, Bob, Charles, and Debbie study in pairs for the SUM exam



- Lecturer misspoke in class, giving rise to a possible misconception
- Some students figured out the problem, others did not

Which of the students has the misconception?

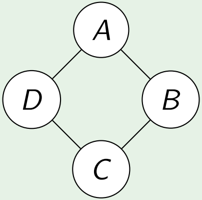
- If A does not have the misconception, he may help B and D
  $\rightarrow$ Students influence each other
- If A has the misconception, he may be helped by B and D
  $\rightarrow$ Influence has no natural "direction"
- A does not study with C $\rightarrow$ No *direct* influence between A and C

Markov network

Definition

A *Markov network* is an undirected graph $\mathcal{H} = (\mathcal{X}, \mathcal{E})$, where $\mathcal{X}$ is a set of random variables and $\mathcal{E} \subseteq \mathcal{X} \times \mathcal{X}$ is the set of edges.

Example



- $\mathcal{X} = \{ A, B, C, D \}$
- $\mathcal{E} = \{ (A, B), (B, C), (C, D), (D, A) \}$

We will see that Markov networks encode a set of conditional independence assumptions between its variables.

Local models

Definition

Let **D** be a set of random variables. A *factor* ϕ is a function from $\text{Dom}(\mathbf{D}) \to \mathbb{R}$. A factor is *nonnegative* if has range $\mathbb{R}^+$. The set **D** is called the *scope* of the factor and is denoted $\text{Scope}[\phi]$.

We restrict attention to nonnegative factors.

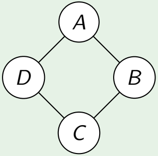
Example

A	B	ϕ_1
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

B	C	ϕ_2
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

C	D	ϕ_3
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

D	A	ϕ_4
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100



- Factors describe "compatibility" between values (not normalized)
- ϕ_1: More "weight" when A and B agree than when they disagree
- ϕ_1: More weight when A and B are both right than when both are wrong
- ϕ_1: If they disagree, more weight when A is right than when B is right

Combining local models

Definition

Let **X**, **Y**, **Z** be three disjoint sets of random variables and let $\phi_1(\mathbf{X}, \mathbf{Y})$ and $\phi_2(\mathbf{Y}, \mathbf{Z})$ be two factors. The *factor product* $\psi = \phi_1 \times \phi_2$ is given by the factor $\psi : \mathrm{Dom}(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) \to \mathbb{R}$ with

$$\psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = \phi_1(\mathbf{X}, \mathbf{Y}) \cdot \phi_2(\mathbf{Y}, \mathbf{Z}).$$

Example

A	B	ϕ_1
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

B	C	ϕ_2
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

A	B	C	ψ
a^0	b^0	c^0	3000
a^0	b^0	c^1	30
a^0	b^1	c^0	5
a^0	b^1	c^1	500
a^1	b^0	c^0	100
a^1	b^0	c^1	1
a^1	b^1	c^0	10
a^1	b^1	c^1	1000

Factor products combine local models by "joining" factors on the common part **Y**.

Factor products and the product rule of probability

Recall the product rule of probability

$$\mathbb{P}(\mathbf{X}, \mathbf{Y}) = \mathbb{P}(\mathbf{Y})\,\mathbb{P}(\mathbf{X} \mid \mathbf{Y}).$$

Example

MusclePain

M	$\mathbb{P}$
Yes	0.1
No	0.9

Flu | MusclePain

M	F	$\mathbb{P}$
Yes	Yes	0.8
Yes	No	0.2
No	Yes	0.1
No	No	0.9

Flu, MusclePain

M	F	$\mathbb{P}$
Yes	Yes	0.08
Yes	No	0.02
No	Yes	0.09
No	No	0.81

- Set $\phi_1(\text{MusclePain}) = \mathbb{P}(\text{MusclePain})$
- Set $\phi_2(\text{MusclePain}, \text{Flu}) = \mathbb{P}(\text{Flu} \mid \text{MusclePain})$
- Set $\psi(\text{MusclePain}, \text{Flu}) = \mathbb{P}(\text{MusclePain}, \text{Flu})$
- Then $\psi = \phi_1 \times \phi_2$

Factor products generalize the product rule of probability.

Gibbs distribution

Definition

A distribution $\mathbb{P}_{\Phi}$ is a *Gibbs distribution* parameterized by a set of factors $\Phi = \{\, \phi_1(\mathbf{D}_1), \ldots, \phi_m(\mathbf{D}_m) \,\}$ if it is defined by

$$\mathbb{P}_{\Phi}\,(\, X_1, \ldots, X_n \,) = \frac{1}{Z}\,\tilde{\mathbb{P}}_{\Phi}\,(\, X_1, \ldots, X_n \,)$$

$$\tilde{\mathbb{P}}_{\Phi}\,(\, X_1, \ldots, X_n \,) = \phi_1(\mathbf{D}_1) \times \phi_2(\mathbf{D}_2) \times \cdots \times \phi_m(\mathbf{D}_m)$$

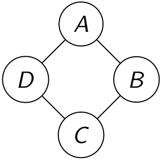
$$Z = \sum_{X_1, \ldots, X_n} \tilde{\mathbb{P}}_{\Phi}\,(\, X_1, \ldots, X_n \,)$$

Here, $\tilde{\mathbb{P}}_{\Phi}\,(\, X_1, \ldots, X_n \,)$ is an *unnormalized measure* and Z a normalizing constant called the *partitioning function*.

- Factors *contribute* to the overall joint distribution
- Overall dist. takes into consideration the contribution from *all* factors

A set of factors defines a Gibbs distribution, i.e., a joint probability distribution over all variables.

Gibbs distribution for Misconception example



A	B	ϕ_1
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

B	C	ϕ_2
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

C	D	ϕ_3
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

D	A	ϕ_4
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

A	B	C	D	$\tilde{\mathbb{P}}$	$\mathbb{P}$
a_0	b_0	c_0	d_0	300,000	0.04
a_0	b_0	c_0	d_1	300,000	0.04
a_0	b_0	c_1	d_0	300,000	0.04
a_0	b_0	c_1	d_1	30	$4.1 \cdot 10^{-6}$
a_0	b_1	c_0	d_0	500	$6.9 \cdot 10^{-5}$
a_0	b_1	c_0	d_1	500	$6.9 \cdot 10^{-5}$
a_0	b_1	c_1	d_0	5,000,000	0.69
a_0	b_1	c_1	d_1	500	$6.9 \cdot 10^{-5}$
a_1	b_0	c_0	d_0	100	$1.4 \cdot 10^{-5}$
a_1	b_0	c_0	d_1	1,000,000	0.14
a_1	b_0	c_1	d_0	100	$1.4 \cdot 10^{-5}$
a_1	b_0	c_1	d_1	100	$1.4 \cdot 10^{-5}$
a_1	b_1	c_0	d_0	10	$1.4 \cdot 10^{-6}$
a_1	b_1	c_0	d_1	100,000	0.014
a_1	b_1	c_1	d_0	100,000	0.014
a_1	b_1	c_1	d_1	100,000	0.014

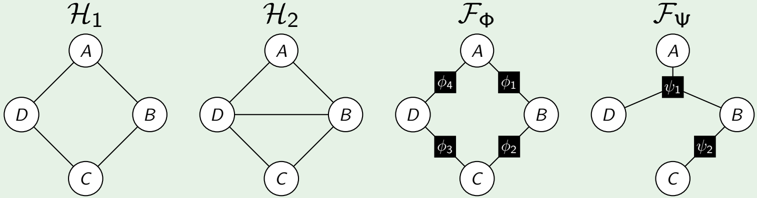
$Z = 7{,}201{,}840$

Factorization and factor graphs

Definition

A distribution $\mathbb{P}_\Phi$ with $\Phi = \{\, \phi_1(\mathbf{D}_1), \ldots, \phi_m(\mathbf{D}_m) \,\}$ *factorizes* over a Markov network $\mathcal{H}$ if each $\mathbf{D}_i$ is a complete subgraph of $\mathcal{H}$. The factors ϕ_i are often called *clique potentials*.

Example



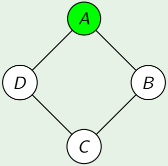
- $\Phi = \{\, \phi_1(A, B), \phi_2(B, C), \phi_3(C, D), \phi_4(D, A) \,\}$
- $\Psi = \{\, \psi_1(A, B, D), \psi_2(B, C) \,\}$
- $\mathbb{P}_\Phi$ factorizes over both $\mathcal{H}_1$ and $\mathcal{H}_2$
- $\mathbb{P}_\Psi$ factorizes over only $\mathcal{H}_2$

Active paths

Definition

Let $X_1 — \ldots — X_k$ be a path in $\mathcal{H} = (\mathcal{X}, \mathcal{E})$. Let $\mathbf{Z} \subseteq \mathcal{X}$ be a set of observed variables. The path $X_1 — \ldots — X_k$ is *active* given $\mathbf{Z}$ if $X_i \notin \mathbf{Z}$ for $1 \leq i \leq k$.

Example



All active paths given A:

- $D–C$
- $C–B$
- $D–C–B$

Some inactive paths given A:

- $D–A–B$
- $C–D–A–B$

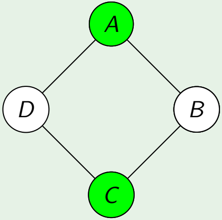
Separation and independencies for Markov networks

Definition

We say that a set of nodes $\mathbf{Z}$ *separates* $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{H}$, denoted $\text{sep}_{\mathcal{H}}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z})$, if there is no active path between any node in $\mathbf{X}$ and any node in $\mathbf{Y}$ given $\mathbf{Z}$. We associate with $\mathcal{H}$ the following set of independencies:

$$\mathcal{I}(\mathcal{H}) = \{ (\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z}) : \text{sep}_{\mathcal{H}}(\mathbf{X}; \mathbf{Y} \mid \mathbf{Z}) \}$$

Example



- $\emptyset$ does not separate any nodes
- $\{ A \}$ does not separate any nodes
- $\{ A, C \}$ separates $\{ B \}$ and $\{ D \}$
- $\{ A, B, C \}$ does not separate any nodes

$$\mathcal{I}(\mathcal{H}) = \{ (B \perp D \mid A, C), (D \perp B \mid A, C) \\ (A \perp C \mid B, D), (C \perp A \mid B, D) \}$$

Relationship Gibbs distributions and Markov networks

Definition

- Let $\mathbb{P}$ be a probability distribution over $\mathbf{X}$. Define $\mathcal{I}(\mathbb{P})$ to be the set of independence assertions of the form $(\mathbf{X} \perp \mathbf{Y} \mid \mathbf{Z})$ that hold in $\mathbb{P}$.
- A Markov network $\mathcal{H}$ is an *I-map* for $\mathbb{P}$ if $\mathcal{I}(\mathcal{H}) \subseteq \mathcal{I}(\mathbb{P})$.

Theorem

Soundness $(\rightarrow)$

Let $\mathbb{P}$ be a distribution and $\mathcal{H}$ be a Markov network over $\mathcal{X}$. If $\mathbb{P}$ is a Gibbs distribution that factorizes over $\mathcal{H}$, then $\mathcal{H}$ is an I-map for $\mathbb{P}$.

Theorem (Hammersley-Clifford theorem)

Soundness $(\leftarrow)$

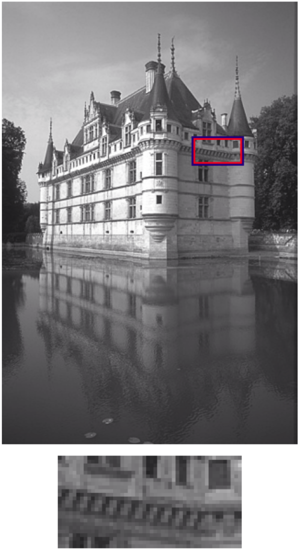
Let $\mathbb{P}$ be a positive distribution and $\mathcal{H}$ be a Markov network over $\mathcal{X}$. If $\mathcal{H}$ is an I-map for $\mathbb{P}$, then $\mathbb{P}$ is a Gibbs distribution that factorizes over $\mathcal{H}$.

Theorem

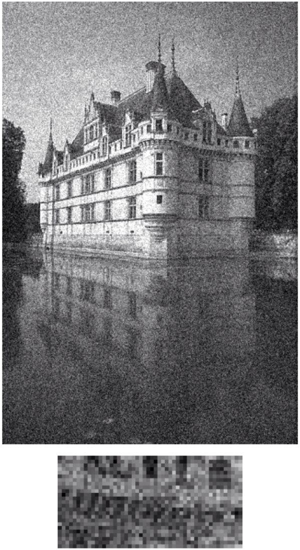
Completeness

If X and Y are not separated given $\mathbf{Z}$ in $\mathcal{H}$, then X and Y are dependent for some distribution $\mathbb{P}$ that factorizes over $\mathcal{H}$.

Application: Image denoising



Original

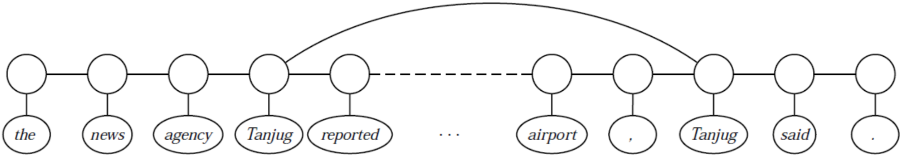


Noisy image

Denoised image

Application: Stanford Named Entity Recognizer

Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names.



- Local evidence often strong clue for label
- Long-range evidence (label consistency) helps when local evidence is insufficient

Finkel, Grenager, and Manning, ACL 2005

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Semantics of Markov logic networks

Definition

A Markov logic network $L = \{ (F_i, w_i) \}$ is a template for constructing Markov networks. Given a set of constants C, a *ground Markov logic* $M_{L,C}$ specifies a distribution over the possible worlds as follows

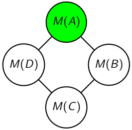
$$\mathbb{P}(\mathbf{X} = \mathbf{x}) \propto \exp\left[\sum_i w_i n_i(\mathbf{x})\right],$$

where $n_i(\mathbf{x})$ is the number of "true groundings" of formula F_i in the possible world $\mathbf{x}$.

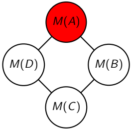
- A possible world $\mathbf{x}$ is likely if
    1. It satisfies many groundings with positive weight
    2. If satisfies few groundings with negative weight
    3. It satisfies groundings with high positive weight
    4. It does not satisfy groundings with high negative weight

How many true groundings does a formula have?

- $F_1 = M(A)$

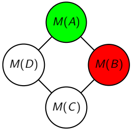


$n_1 = 1$

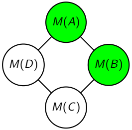


$n_1 = 0$

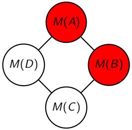
- $F_2 = M(A) \vee M(B)$



$n_2 = 1$



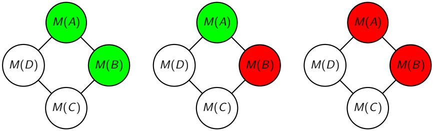
$n_2 = 1$



$n_2 = 0$

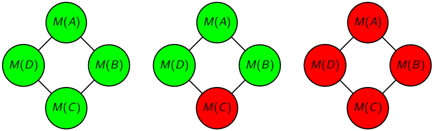
How many true groundings does a formula have? (2)

- $F_3 = M(A) \land M(B)$



FOL:	1	0	0	(strict)
MLN:	$n_3 = 2$	$n_3 = 1$	$n_3 = 0$	(smoothed)

- $F_4 = \forall x.M(x)$

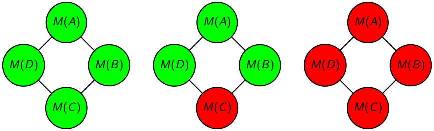


FOL:	1	0	0	(strict)
MLN:	$n_4 = 4$	$n_4 = 3$	$n_4 = 0$	(smoothed)

Conjunctions in FOL are sensitive to noise: If just one of the conjuncts is unsatisfied, the formula is also unsatisfied. MLNs count how many of the conjuncts are true and thus are less sensitive to noise.

How many true groundings does a formula have? (3)

- $F_5 = \exists x . M(x)$



FOL:	1	1	0	(strict)
MLN:	$n_4 = 1$	$n_4 = 1$	$n_4 = 0$	(strict)

Disjunctions in FOL are insensitive to noise, so we are fine.

Grounding a formula in Markov logic

Let F be a formula and $C = \{ c_1, \dots, c_d \}$ be a set of constants.
Conceptually, we obtain the set $G(F)$ of ground formulas as follows:

1. Whenever a subformulas of form $\exists x.F'(x)$ occurs, replace by $(F'(c_1) \vee \cdots \vee F'(c_d))$
2. Convert the formula to form $\forall \mathbf{x}.F'(\mathbf{x})$, where F' is in conjunctive normal form and is quantifier-free, optionally simplify, denote result by $\mathrm{cnf}(F)$
3. For all $\mathbf{c} \in C^{|\mathbf{x}|}$, set $G(F, \mathbf{c}) = \{ G : G \text{ is a clause in } F'(\mathbf{c}) \}$
4. Set $G(F) = \{ G(F, \mathbf{c}) : \mathbf{c} \in C^{|\mathbf{x}|} \}$

Example

- $C = \{ A, B \}$
- $F_1 = \forall x.\mathrm{Smokes}(x) \implies \mathrm{Cancer}(x)$
  1. No existential quantifiers → nothing to do
  2. $\mathrm{cnf}(F_1) = \forall x.\neg \mathrm{S}(x) \vee \mathrm{C}(x)$
  3. $G(F_1, A) = \{ \neg \mathrm{S}(A) \vee \mathrm{C}(A) \}$
     $G(F_1, B) = \{ \neg \mathrm{S}(B) \vee \mathrm{C}(B) \}$
  4. $G(F_1) = \{ \{ \neg \mathrm{S}(A) \vee \mathrm{C}(A) \}, \{ \neg \mathrm{S}(B) \vee \mathrm{C}(B) \} \}$

Grounding a formula (example)

Example

- $C = \{ A, B \}$
- $F_2 = \forall x . \forall y . \mathrm{Friends}(x, y) \implies (\mathrm{Smokes}(x) \iff \mathrm{Smokes}(y))$
  1. No existential quantifiers → nothing to do
  2. $\mathrm{cnf}(F_2) = \forall x . \forall y . [\neg \mathrm{F}(x, y) \vee \mathrm{S}(x) \vee \neg \mathrm{S}(y)] \wedge [\neg \mathrm{F}(x, y) \vee \neg \mathrm{S}(x) \vee \mathrm{S}(y)]$
  3. $$\begin{aligned} G(F_2, (A, A)) &= \{ \neg \mathrm{F}(A, A) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(A), \neg \mathrm{F}(A, A) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(A) \} \\ G(F_2, (A, B)) &= \{ \neg \mathrm{F}(A, B) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(B), \neg \mathrm{F}(A, B) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(B) \} \\ G(F_2, (B, A)) &= \{ \neg \mathrm{F}(B, A) \vee \mathrm{S}(B) \vee \neg \mathrm{S}(A), \neg \mathrm{F}(B, A) \vee \neg \mathrm{S}(B) \vee \mathrm{S}(A) \} \\ G(F_2, (B, B)) &= \{ \neg \mathrm{F}(B, B) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(B), \neg \mathrm{F}(B, B) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(B) \} \end{aligned}$$
  4. $$\begin{aligned} G(F_2) = \{ &\{ \neg \mathrm{F}(A, A) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(A), \neg \mathrm{F}(A, A) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(A) \}, \\ &\{ \neg \mathrm{F}(A, B) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(B), \neg \mathrm{F}(A, B) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(B) \}, \\ &\{ \neg \mathrm{F}(B, A) \vee \mathrm{S}(B) \vee \neg \mathrm{S}(A), \neg \mathrm{F}(B, A) \vee \neg \mathrm{S}(B) \vee \mathrm{S}(A) \}, \\ &\{ \neg \mathrm{F}(B, B) \vee \mathrm{S}(A) \vee \neg \mathrm{S}(B), \neg \mathrm{F}(B, B) \vee \neg \mathrm{S}(A) \vee \mathrm{S}(B) \} \} \end{aligned}$$

Grounding a Markov logic network

Given an MLN $\{\,(F_i, w_i)\,\}$ and a set of constants C.

1. Create a Boolean variable $R(\mathbf{c})$ for each predicate that occurs in one of the formulas and each $\mathbf{c} \in C^m$, where m is the arity of the relation
2. For each formula F_i
   1. Ground F_i to obtain $G(F_i)$
   2. For each ground set of clauses $G(F_i, \mathbf{c}) \in G(F_i)$
      1. Split weight evenly among clauses: $w_i' = w_i / |G(F_i, \mathbf{c})|$
      2. For each clause F_{ij} in $G(F_i, \mathbf{c})$, create a factor

$$\phi(\mathbf{D}_{ij}) = w_i' f_{ij}(\mathbf{D}_{ij}),$$

where $\mathbf{D}_{ij}$ is the set of variables that occur in F_{ij}, and

$$f_{ij}(\mathbf{D}_{ij}) = \begin{cases} 1 & \text{if } j\text{-th clause in in } G(F_i, \mathbf{c}) \text{ is satisfied for assignment } \mathbf{D}_{ij} \\ 0 & \text{otherwise} \end{cases}$$

is an “indicator feature” with weight w_i'.

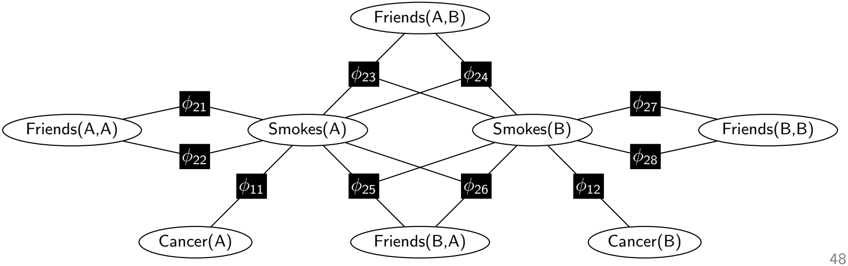
The weight of a ground CNF formula is split evenly among its clauses.

Grounding a Markov logic network (example)

F_1: 1.5 $\Big\{$ Smoking causes cancer
$\forall x.\mathrm{Smokes}(x) \implies \mathrm{Cancer}(x)$

F_2: 1.1 $\Big\{$ Friends have similar smoking habits
$\forall x.\forall y.\mathrm{Friends}(x, y) \implies (\mathrm{Smokes}(x) \iff \mathrm{Smokes}(y))$

$$
\begin{array}{rlr}
C = & \{A, B\} & \\
G(F_1) = & \{\{\neg \mathrm{S}(A) \lor \mathrm{C}(A)\}, & f_{11}, w'_{11} = 1.50 \\
 & \;\{\neg \mathrm{S}(B) \lor \mathrm{C}(B)\}\} & f_{12}, w'_{12} = 1.50 \\
G(F_2) = & \{\{\neg \mathrm{F}(A, A) \lor \mathrm{S}(A) \lor \neg \mathrm{S}(A), & f_{21}, w'_{21} = 0.55 \\
 & \quad \neg \mathrm{F}(A, A) \lor \neg \mathrm{S}(A) \lor \mathrm{S}(A)\}, & f_{22}, w'_{22} = 0.55 \\
 & \;\{\neg \mathrm{F}(A, B) \lor \mathrm{S}(A) \lor \neg \mathrm{S}(B), & f_{23}, w'_{23} = 0.55 \\
 & \quad \neg \mathrm{F}(A, B) \lor \neg \mathrm{S}(A) \lor \mathrm{S}(B)\}, & f_{24}, w'_{24} = 0.55 \\
 & \;\{\neg \mathrm{F}(B, A) \lor \mathrm{S}(B) \lor \neg \mathrm{S}(A), & f_{25}, w'_{25} = 0.55 \\
 & \quad \neg \mathrm{F}(B, A) \lor \neg \mathrm{S}(B) \lor \mathrm{S}(A)\}, & f_{26}, w'_{26} = 0.55 \\
 & \;\{\neg \mathrm{F}(B, B) \lor \mathrm{S}(A) \lor \neg \mathrm{S}(B), & f_{27}, w'_{27} = 0.55 \\
 & \quad \neg \mathrm{F}(B, B) \lor \neg \mathrm{S}(A) \lor \mathrm{S}(B)\}\} & f_{28}, w'_{28} = 0.55
\end{array}
$$



Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
   - Introduction
   - Preliminaries
3. Markov Networks
4. Markov Logic Networks
   - Grounding Markov logic networks
   - Log-Linear Models
5. Inference in MLNs
   - Basics
   - Exact Inference
   - Approximate Inference
6. Summary

Log-linear model

Definition

A positive distribution $\mathbb{P}$ is a log-linear model over a Markov network $\mathcal{H}$ if it is associated with

- a set of *features* $\mathcal{F} = \{ f_1(\mathbf{D}_1), \dots, f_m(\mathbf{D}_m) \}$, where each $\mathbf{D}_i$ is a complete subgraph in $\mathcal{H}$
- a set of weights $w_1, \dots, w_m$

such that

$$\mathbb{P}(X_1, \dots, X_n) \propto \exp \left[\sum_{i=1}^{m} w_i f_i(\mathbf{D}_i) \right] .$$

The terms $\epsilon_i(\mathbf{D}_i) = -w_i f_i(\mathbf{D}_i)$ are called *energy functions*.

$\log \mathbb{P}(X_1, \dots, X_n)$ is a linear combination of the the features. The linearity allows us to detect and *eliminate* redundancy in the features (using standard linear algebra techniques).

From factors to features

Definition

Let **D** be a subset of variables. An *indicator feature* is a function $f(\mathbf{D}) : \mathbf{D} \to \{\,0, 1\,\}$.

Theorem

Every factor of a graphical model on discrete variables can be expressed in terms of a linear combination of weighted indicator features.

Proof (Boolean case).

Consider a factor $\phi(X_1, \ldots, X_k)$ on k Boolean variables. Let Θ be the set of all assignments of values to $X_1, \ldots, X_k$. Set

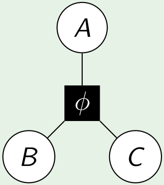
$$\begin{aligned} w_\theta &= \ln \phi(X_1[\theta], \ldots, X_k[\theta]) && \text{(constants)} \\ f_\theta(X_1, \ldots, X_k) &= \begin{cases} 1 & \text{if } X_1 = X_1[\theta], \ldots, X_k = X_k[\theta] \\ 0 & \text{otherwise} \end{cases} && \text{(indicator features)} \\ \ln \phi(X_1, \ldots, X_k) &= \textstyle\sum_{\theta \in \Theta} w_\theta f_\theta(X_1, \ldots, X_k) && \text{(decomposition)} \end{aligned}$$

□

From factors to features (example)

Example

Consider three friends with similiar interests and let A, B, C be Boolean variables that indicate whether each of the friends likes football.



ϕ

X_1	X_2	X_3	ϕ	w
F	F	F	10	2.3
F	F	T	1	0
⋮	⋮	⋮	⋮	⋮
T	F	T	1	0
T	T	F	1	0
T	T	T	10	2.3

We have

$$\ln \phi(A, B, C) = \sum_{\theta} w_\theta f_\theta(A, B, C) = 2.3 \cdot f_{FFF}(A, B, C) + 2.3 \cdot f_{TTT}(A, B, C).$$

Even more compact: $\ln \phi(A, B, C) = 2.3 \cdot I_{ABC \vee \neg A \neg B \neg C}$

From Gibbs distribution to log-linear models

Theorem

Every positive Gibbs distribution $\mathbb{P}$ *over* $\mathcal{H}$ *on Boolean variables* $X_1, \ldots, X_n$ *has a log-linear model over* $\mathcal{H}$ *with only indicator features and vice versa.*

Proof.

$$\begin{aligned} \mathbb{P}(X_1, \ldots, X_n) &= \frac{1}{Z} \prod_{i=1}^{m} \phi_i(\mathbf{D}_i) \\ &= \frac{1}{Z} \exp\left[\sum_{i=1}^{m} \ln \phi_i(\mathbf{D}_i)\right] \\ &= \frac{1}{Z} \exp\left[\sum_{i=1}^{m} \sum_{\theta \in \Theta_{\mathbf{D}_i}} w_\theta f_\theta(\mathbf{D}_i)\right]. \end{aligned}$$

☐

Markov logic networks are "templates" for constructing log-linear models. Any positive Gibbs distribution with finite-domain variables can be modeled.

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
    - Introduction
    - Preliminaries
3. Markov Networks
4. Markov Logic Networks
    - Grounding Markov logic networks
    - Log-Linear Models
5. Inference in MLNs
    - Basics
    - Exact Inference
    - Approximate Inference
6. Summary

Outline

Inference in probabilistic graphical models

- Recall the queries of interest
  1. Conditional probability query
  2. MAP query
  3. Marginal MAP query

Definition

Let $\mathbb{P}_\Phi$ be a Gibbs distribution over variables $\{\, X, X_1, \ldots, X_n \,\}$.

1. The $\mathbb{P}_\Phi$-*decision problem* asks whether $\mathbb{P}_\Phi\,(\, X = x \,) > 0$,
2. The $\mathbb{P}_\Phi$-*probability computation problem* asks for $\mathbb{P}_\Phi\,(\, X = x \,)$.

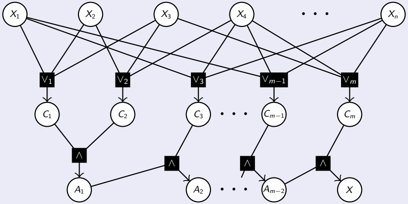
Complexity of inference in probabilistic graphical models

Theorem

The $\mathbb{P}_\Phi$-decision problem is NP-complete, $\mathbb{P}_\Phi$-probability computation is #P-hard.

Proof (by reduction from 3-SAT and #3-SAT).

Take a 3-SAT formula $\Psi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ over variables $\mathcal{X} = \{ X_1, X_2, \ldots, X_n \}$. Consider the following Gibbs distribution $\mathbb{P}_\Phi$ over Boolean variables:



Here, $\vee_i(C_i, \mathbf{X}_i) = 1$ if for assignment $\mathbf{X}_i$ the truth value of clause C_i equals variable C_i, else $\vee_i(C_i, \mathbf{X}_i) = 0$; similarly for $\wedge$-factors. $\mathbb{P}_\Phi$ can be computed in polynomial time in the size of Ψ. Assertion 1 follows since $\mathbb{P}_\Phi (X = \mathtt{TRUE}) > 0$ if and only if Ψ is satisfiable. $\mathbb{P}_\Phi (X = \mathtt{TRUE}) = \mathbb{P} (\Psi)$ where $\mathbb{P} (X_i = \mathtt{TRUE}) = 1/2$ and the $\{ X_i \}$ are i.i.d. Assertion 2 follows since $\#\Psi = 2^n \, \mathbb{P} (\Psi) = 2^n \, \mathbb{P}_\Phi (X = \mathtt{TRUE})$. □

Queries in Markov logic

- Standard PGM queries, e.g.,
  $\mathbb{P}(\,\text{Smokes(B)}, \text{Cancer(B)} \mid \text{Smokes(A)} \wedge \text{Friends(A,B)} \wedge \dots\,)$
  $\rightarrow$ #P-hard
- More general queries of form "What is the probability that formula F_1 holds given that formula F_2 holds?", e.g.,
  $\mathbb{P}(\,\exists x.\text{Cancer(x)} \mid \forall x.\text{Smokes(x)}\,)$
- Let L be an MLN and C be a set of constants

$$\begin{aligned}\mathbb{P}(\,F_1 \mid F_2, L, C\,) &= \mathbb{P}(\,F_1 \mid F_2, M_{L,C}\,)\\ &= \frac{\mathbb{P}(\,F_1 \wedge F_2 \mid M_{L,C}\,)}{\mathbb{P}(\,F_2 \mid M_{L,C}\,)}\\ &= \frac{\sum_{\mathbf{x}\in\mathcal{X}_{F_1}\cap\mathcal{X}_{F_2}} \mathbb{P}(\,\mathcal{X}=\mathbf{x} \mid M_{L,C}\,)}{\sum_{\mathbf{x}\in\mathcal{X}_{F_2}} \mathbb{P}(\,\mathcal{X}=\mathbf{x} \mid M_{L,C}\,)},\end{aligned}$$

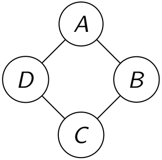
where $\mathcal{X}_F$ is the set of worlds in which F holds

We focus on standard PGM queries.

Outline

Naive approach

Exponential in number of variables!



A	B	ϕ_1
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

B	C	ϕ_2
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

C	D	ϕ_3
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

D	A	ϕ_4
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100

A	B	C	D	$\tilde{\mathbb{P}}$	$\mathbb{P}$
a_0	b_0	c_0	d_0	300,000	0.04
a_0	b_0	c_0	d_1	300,000	0.04
a_0	b_0	c_1	d_0	300,000	0.04
a_0	b_0	c_1	d_1	30	$4.1 \cdot 10^{-6}$
a_0	b_1	c_0	d_0	500	$6.9 \cdot 10^{-5}$
a_0	b_1	c_0	d_1	500	$6.9 \cdot 10^{-5}$
a_0	b_1	c_1	d_0	5,000,000	0.69
a_0	b_1	c_1	d_1	500	$6.9 \cdot 10^{-5}$
a_1	b_0	c_0	d_0	100	$1.4 \cdot 10^{-5}$
a_1	b_0	c_0	d_1	1,000,000	0.14
a_1	b_0	c_1	d_0	100	$1.4 \cdot 10^{-5}$
a_1	b_0	c_1	d_1	100	$1.4 \cdot 10^{-5}$
a_1	b_1	c_0	d_0	10	$1.4 \cdot 10^{-6}$
a_1	b_1	c_0	d_1	100,000	0.014
a_1	b_1	c_1	d_0	100,000	0.014
a_1	b_1	c_1	d_1	100,000	0.014

$Z = 7{,}201{,}840$

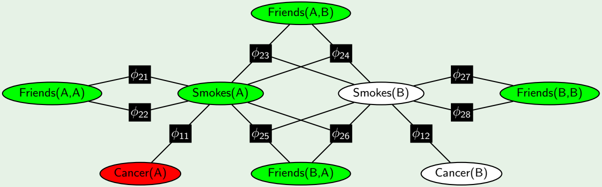
Grounding with evidence (1)

Denote by M the weighted ground clauses in a ground Markov logic network $M_{L,C}$. Given evidence $\mathbf{E}$, we can partition M into:

1. Clauses M_1 that involve only observed variables
2. Clauses M_2 that involve both observed and latent variables
3. Clauses M_3 that involve only latent variables

$$\begin{aligned}\log \mathbb{P}(\,\mathbf{W} \mid \mathbf{E}\,) &= -\log Z + \sum_{\phi=(f,w)\in M} wf(\mathbf{W}_f, \mathbf{E}_f) \\ &= -\log Z + \underbrace{\sum_{(f,w)\in M_1} wf(\mathbf{E}_f)}_{\text{Constant}} + \sum_{(f,w)\in M_2} wf(\mathbf{W}_f, \mathbf{E}_f) + \sum_{(f,w)\in M_3} wf(\mathbf{W}_f)\end{aligned}$$

Example



$$M_1 = \{\, \phi_{11}, \phi_{21}, \phi_{22} \,\}$$

$$M_2 = \{\, \phi_{23}, \phi_{24}, \phi_{25}, \phi_{26}, \phi_{27}, \phi_{28} \,\}$$

$$M_3 = \{\, \phi_{12} \,\}$$

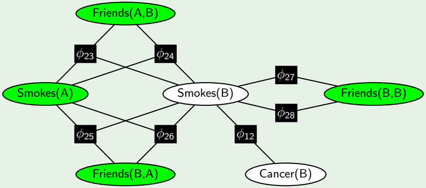
Grounding with evidence (2)

Denote by M the weighted ground clauses in a ground Markov logic network $M_{L,C}$. Given evidence $\mathbf{E}$, we can partition M into:

1. Clauses M_1 that involve only observed variables
2. Clauses M_2 that involve both observed and latent variables
3. Clauses M_3 that involve only latent variables

$$\begin{aligned}\log \mathbb{P}(\mathbf{W} \mid \mathbf{E}) &= -\log Z + \sum_{\phi=(f,w)\in M} wf(\mathbf{W}_f, \mathbf{E}_f) \\ &= -\log Z' + \underbrace{\sum_{(f,w)\in M_2} wf(\mathbf{W}_f, \mathbf{E}_f)}_{\text{Replace observed variables by their values}} + \sum_{(f,w)\in M_3} wf(\mathbf{W}_f)\end{aligned}$$

Example



$M_2 = \{ \phi_{23}, \phi_{24}, \phi_{25}, \phi_{26}, \phi_{27}, \phi_{28} \}$

$M_3 = \{ \phi_{12} \}$

Grounding with evidence (3)

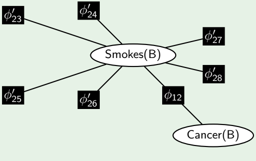
Denote by M the weighted ground clauses in a ground Markov logic network $M_{L,C}$. Given evidence $\mathbf{E}$, we can partition M into:

1. Clauses M_1 that involve only observed variables
2. Clauses M_2 that involve both observed and latent variables
3. Clauses M_3 that involve only latent variables

$$\begin{aligned}\log \mathbb{P}(\mathbf{W} \mid \mathbf{E}) &= -\log Z + \sum_{\phi=(f,w)\in M} wf(\mathbf{W}_f, \mathbf{E}_f) \\ &= -\log Z' + \sum_{(f,w)\in M_2'} wf(\mathbf{W}_f) + \sum_{(f,w)\in M_3} wf(\mathbf{W}_f) \\ &= -\log Z' + \sum_{(f,w)\in M'} wf(\mathbf{W}_f)\end{aligned}$$

No observed variables are left. Gives rise to efficient grounding methods.

Example



$$M_2' = \{\, \phi'_{23}, \phi'_{24}, \phi'_{25}, \phi'_{26}, \phi'_{27}, \phi'_{28} \,\}$$

$$M_3 = \{\, \phi_{12} \,\}$$

$$M' = M_2' \cup M_3$$

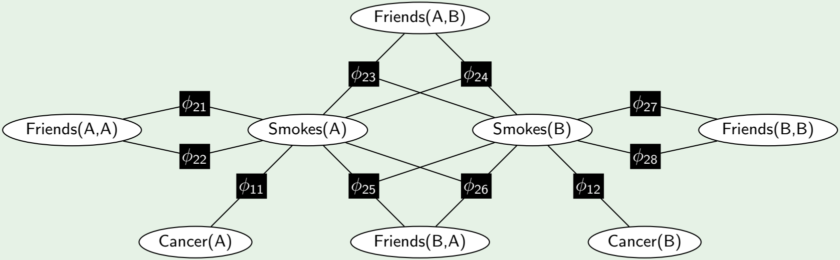
$$\phi_{24} = \neg F(A,B) \vee \neg S(A) \vee S(B)$$

$$\begin{aligned}\phi'_{24} &= \texttt{FALSE} \vee \texttt{FALSE} \vee S(B) \\ &= S(B)\end{aligned}$$

MAP inference for MLNs (1)

Example

What is the most likely world for a given Markov logic network?



Corresponds to weighted CNF formula:

$$\Psi = f_{11} \wedge f_{12} \wedge f_{23} \wedge f_{24} \wedge f_{25} \wedge f_{26} \wedge f_{27} \wedge f_{28}$$

MAP inference for MLNs (2)

Definition

Consider a CNF formula F over variables $\mathcal{X}$, in which each of the clauses $f_1, \ldots, f_m$ is associated with a corresponding weight $w_1, \ldots, w_m$. The *Weighted MAX-SAT problem* is to find an assignment $\mathbf{x}^* \in \mathcal{X}_F$ that maximizes the sum of the weights of satisfied clauses, i.e., $\mathbf{x}^* = \text{argmax}_{\mathbf{x}} \sum_i w_i f_i$.

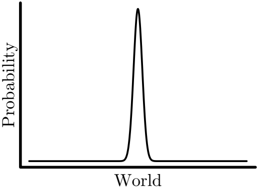
Consider the following transformation:

$$\underset{\mathbf{x}}{\text{argmax}}\, \mathbb{P}(\mathbf{x}) = \underset{\mathbf{x}}{\text{argmax}} \left[\frac{1}{Z} \exp \sum_{(f,w) \in M_{L,C}} wf(\mathbf{x}) \right]$$

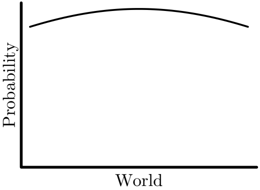
$$= \underset{\mathbf{x}}{\text{argmax}} \sum_{(f,w) \in \underbrace{M_{L,C}}_{F}} \underbrace{w}_{w_i} \underbrace{f(\mathbf{x})}_{f_i} = \mathbf{x}^*$$

There are many algorithms and solvers for Weighted MAX-SAT, both exact and approximate. Specialized algorithms for MLNs do exist; they try to reduce grounding by computing $M_{L,C}$ only partially.

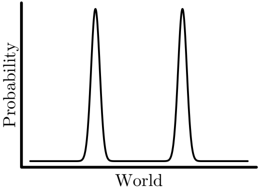
MAP inference for MLNs (3)



MAP world characterizes distribution well



MAP world not distinguished from other words

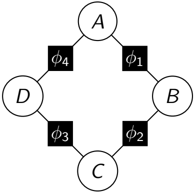


MAP world(s) characterize only a part of the distribution

MAP estimates provide the “most consistent” world, i.e., the world that satisfies most of the rules. This world *may or may not* characterize the entire distribution well.

Variable elimination (idea)

Goal: Eliminate non-query variables from the graph.



ϕ_1

A	B	ϕ
a^0	b^0	30
a^0	b^1	5
a^1	b^0	1
a^1	b^1	10

ϕ_2

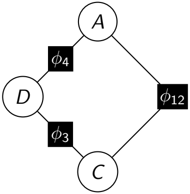
B	C	ϕ
b^0	c^0	100
b^0	c^1	1
b^1	c^0	1
b^1	c^1	100

ϕ_3

C	D	ϕ
c^0	d^0	1
c^0	d^1	100
c^1	d^0	100
c^1	d^1	1

ϕ_4

D	A	ϕ
d^0	a^0	100
d^0	a^1	1
d^1	a^0	1
d^1	a^1	100



$\phi_1 \times \phi_2$

A	B	C	ϕ
a^0	b^0	c^0	3000
a^0	b^0	c^1	30
a^0	b^1	c^0	5
a^0	b^1	c^1	500
a^1	b^0	c^0	100
a^1	b^0	c^1	1
a^1	b^1	c^0	10
a^1	b^1	c^1	1000

ϕ_{12}

A	C	ϕ
a^0	c^0	3005
a^0	c^1	530
a^1	c^0	110
a^1	c^1	1001

B has been eliminated ("marginalized out"). The resulting factor graph represents $\mathbb{P}(A, C, D)$.

Variable elimination (why it works)

Recall that

$$\mathbb{P}\left(A, B, C, D \right) = \frac{1}{Z} \phi_1(A, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, A)$$

and thus

$$\begin{aligned}
\mathbb{P}\left(A, C, D \right) &= \mathbb{P}\left(A, b^0, C, D \right) + \mathbb{P}\left(A, b^1, C, D \right) \\
&= \frac{1}{Z} [\phi_1(A, b^0) \times \phi_2(b^0, C) \times \phi_3(C, D) \times \phi_4(D, A) \\
&\quad + \phi_1(A, b^1) \times \phi_2(b^1, C) \times \phi_3(C, D) \times \phi_4(D, A)] \\
&= \frac{1}{Z} \left[\left\{ \sum_{b \in \{ b^0, b^1 \}} \phi_1(A, b) \times \phi_2(b, C) \right\} \times \phi_3(C, D) \times \phi_4(D, A) \right] \\
&= \frac{1}{Z} \left[\phi_{12}(A, C) \times \phi_3(C, D) \times \phi_4(D, A) \right]
\end{aligned}$$

Variable elimination (remarks)

- Also called sum-product variable elimination
- Whenever we eliminate a variable B
  - We remove all factors connected to B
  - We introduce a single factor that is connected to the neighbors of B
  - If B has k neighbors, the new factor has 2^k rows
    $\rightarrow$ Potentially exponential blow-up
- Computational cost
  - Dominated by sizes of intermediate factors
  - Depends strongly on elimination ordering
  - NP-hard to find optimal ordering
  - Lots of useful heuristics exist
  - "Conditioning" can be used to avoid large factors for increased processing time
- Similar observations give rise to other important algorithms, e.g., "message passing" in "clique trees"

Outline

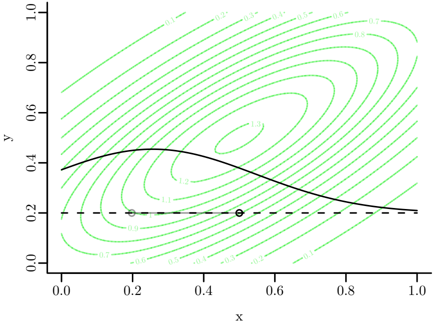
Sampling methods

- Also called *particle-based approximate inference*
- Idea: Obtain samples from the distribution underlying the graphical model
- If samples were independent, we could count how often each variables is true/false and apply the sampling theorem
- Sampling is much more difficult in Markov networks → samples are generally dependent
  - Goal is to minimize the dependencies
  - More samples needed than "implied" by the sampling theorem
  - If dependencies vanish between far-apart samples → correctness and convergence
- Many techniques
  - Forward sampling (for directed models)
  - Likelihood weighting
  - Importance sampling
  - Gibbs sampling
  - Other Markov Chain Monte Carlo (MCMC) methods
  - Collapsed particles

Gibbs sampling (idea)

Gibbs sampling is a simple algorithm to sample from $\mathbb{P}(\, X, Y \,)$. It is used when it is hard to sample from $\mathbb{P}(\, X, Y \,)$, but easy to sample from $\mathbb{P}(\, X \mid Y \,)$ and $\mathbb{P}(\, Y \mid X \,)$.

1. Pick an initial point (x_0, y_0)
2. For $n = 1, 2, \ldots$
    1. Generate $x_n \sim \mathbb{P}(\, X \mid Y = y_{n-1} \,)$

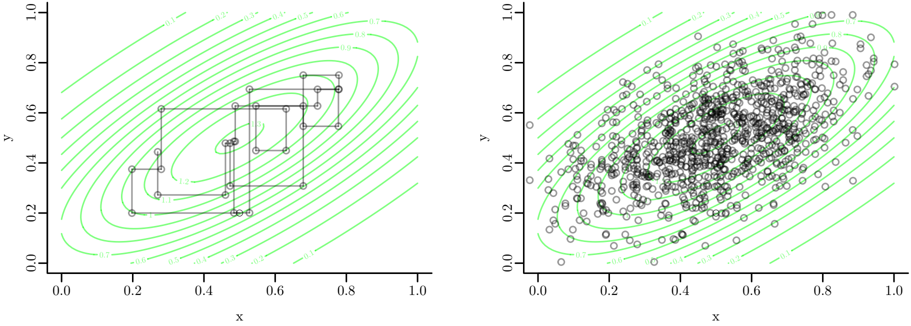


$n = 1$

Gibbs sampling (idea)

Gibbs sampling is a simple algorithm to sample from $\mathbb{P}(X, Y)$. It is used when it is hard to sample from $\mathbb{P}(X, Y)$, but easy to sample from $\mathbb{P}(X \mid Y)$ and $\mathbb{P}(Y \mid X)$.

1. Pick an initial point (x_0, y_0)
2. For $n = 1, 2, \ldots$
    1. Generate $x_n \sim \mathbb{P}(X \mid Y = y_{n-1})$
    2. Generate $y_n \sim \mathbb{P}(Y \mid X = x_n)$



$n = 15$ $n = 500$

Gibbs sampling for Markov networks

Recall that

$$\mathbb{P}(A, B, C, D) = \frac{1}{Z}\phi_1(A, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, A).$$

Sampling from $\mathbb{P}(A, B, C, D)$ is hard but sampling from

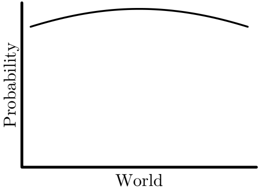
$$\begin{aligned}\mathbb{P}(A \mid B, C, D) &= \frac{\mathbb{P}(A, B, C, D)}{\mathbb{P}(B, C, D)} \\ &= \frac{\frac{1}{Z}[\phi_1(A, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, A)]}{\frac{1}{Z}\sum_{a \in \{a^0, a^1\}}[\phi_1(a, B) \times \phi_2(B, C) \times \phi_3(C, D) \times \phi_4(D, a)]} \\ &= \frac{\phi_1(A, B) \times \phi_4(D, A)}{\sum_{a \in \{a^0, a^1\}} \phi_1(a, B) \times \phi_4(D, a)}\end{aligned}$$

is easy. Only the factors connected to A remain.

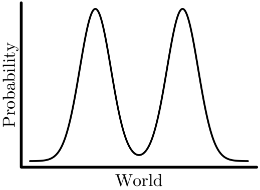
When resampling a variable A, we only have to look at the factors connected to A, and thus only the subset of variables connected to A. These variables are called the *Markov blanket* of A.

Gibbs sampling for Markov networks (remarks)

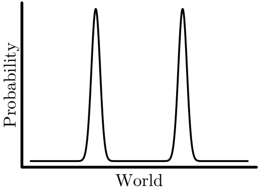
- Variables are picked according to a *schedule*
  → sequential, random, . . .
- An instance of the more general class of MCMC methods
  - Markov chains describe how the sampling process moves through the set of worlds
  - Irreducible if all worlds can be reached from all other worlds
  - Convergence speed depends on how fast the sampling process moves (*mixing time*)



Gibbs sampling works well (fast mixing)



Gibbs sampling works reasonable (slow mixing)



Gibbs sampling does not work (not irreducible)

- MCMC methods can perform "bigger" steps than Gibbs sampling; they change multiple variables simultaneously

Outline

1. Introduction to Markov Logic Networks
2. Probabilistic Graphical Models
    - Introduction
    - Preliminaries
3. Markov Networks
4. Markov Logic Networks
    - Grounding Markov logic networks
    - Log-Linear Models
5. Inference in MLNs
    - Basics
    - Exact Inference
    - Approximate Inference
6. **Summary**

Lessons learned

- Probabilistic databases and graphical models focus on different aspects of probabilistic reasoning
- Probabilistic graphical models
  - Describe and reason about probability distributions and independencies
  - Exploit locality structure (conditional independence)
  - Main components: representation, inference, learning
- Markov logic
  - Combines first-order logic and probability theory
  - Set of formulas with weights
  - Template for generating undirected graphical models
- Inference
  - #P-hard in general
  - MAP inference on MLNs corresponds to Weighted MAX-SAT
  - Exact methods for probability computation (e.g., variable elimination) may work when graph has no dense regions
  - Approximate methods often based on MCMC sampling
  - Gibbs sampling is the simplest MCMC method; it changes one variable at a time

Suggested reading

- Daphne Koller, Nir Friedman
  Probabilistic Graphical Models: Principles and Techniques
  The MIT Press, 2009

- Matthew Richardson and Pedro Domingos
  Markov Logic Networks
  Machine Learning, 62(1-2), pp. 107–136, 2006

- Michael Mitzenmacher, Eli Upfal
  Probability and Computing: Randomized Algorithms and Probabilistic Analysis
  Cambridge University Press, 2005

- `http://alchemy.cs.washington.edu/`