Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken

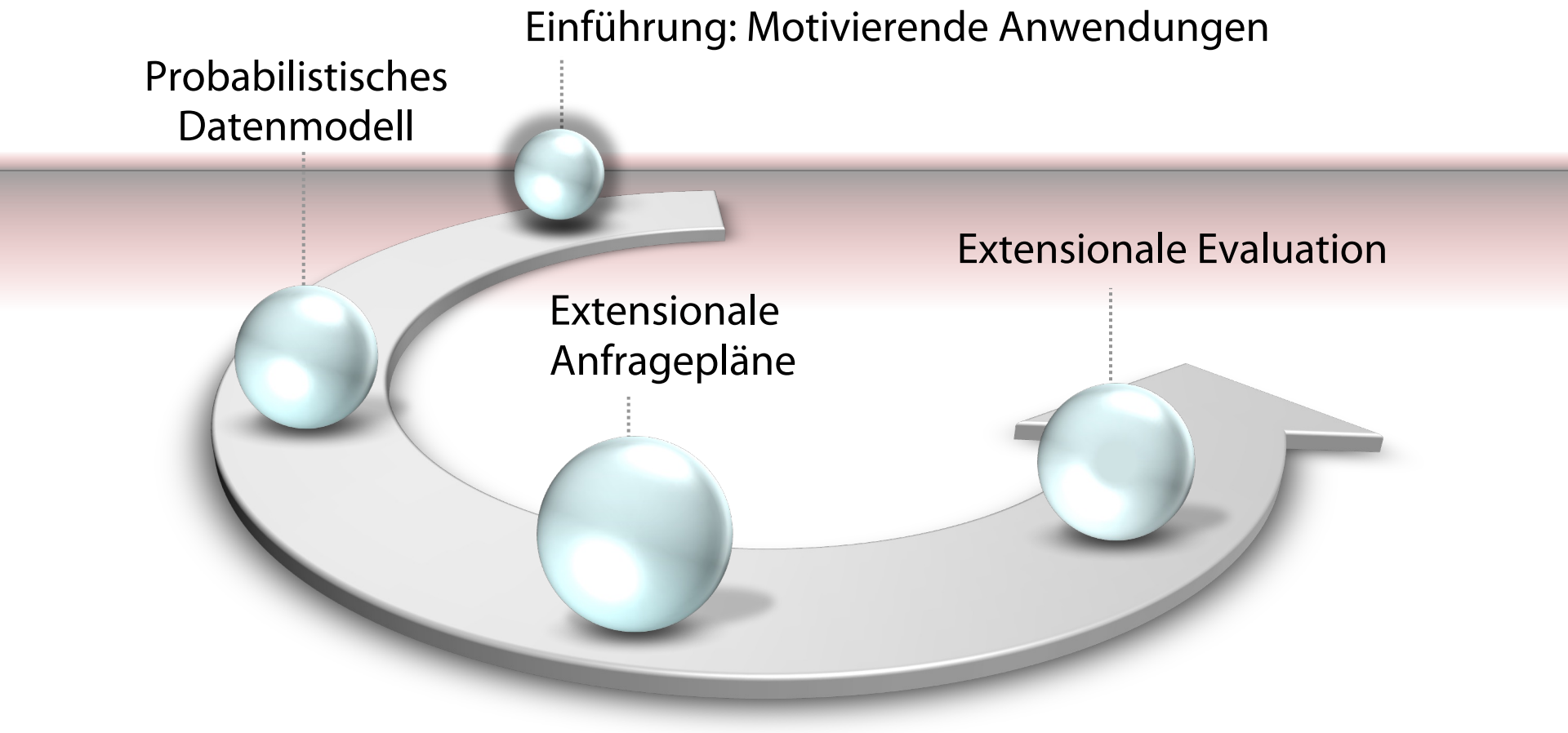
Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Übersicht

- Semistrukturierte Datenbanken (JSON, XML) und Volltextsuche
- Information Retrieval
- Mehrdimensionale Indexstrukturen
- Cluster-Bildung
- First-n-, Top-k-, und Skyline-Anfragen
- Probabilistische Datenbanken, Anfragebeantwortung, Top-k-Anfragen und Open-World-Annahme
- Probabilistische Modellierung, Bayes-Netze, Anfragebeantwortungsalgorithmen, Lernverfahren,
- Temporale Datenbanken und das relationale Modell,
- Probabilistische Temporale Datenbanken
- SQL: neue Entwicklungen (z.B. JSON-Strukturen und Arrays), Zeitreihen (z.B. TimeScaleDB)
- Stromdatenbanken, Prinzipien der Fenster-orientierten inkrementellen Verarbeitung
- Approximationstechniken für Stromdatenverarbeitung, Stream-Mining
- Probabilistische raum-zeitliche Datenbanken und Stromdatenverarbeitungsssysteme: Anfragen und Indexstrukturen, Raum-zeitliches Data Mining, Probabilistische Skylines
- Von NoSQL- zu NewSQL-Datenbanken, CAP-Theorem, Blockchain-Datenbanken

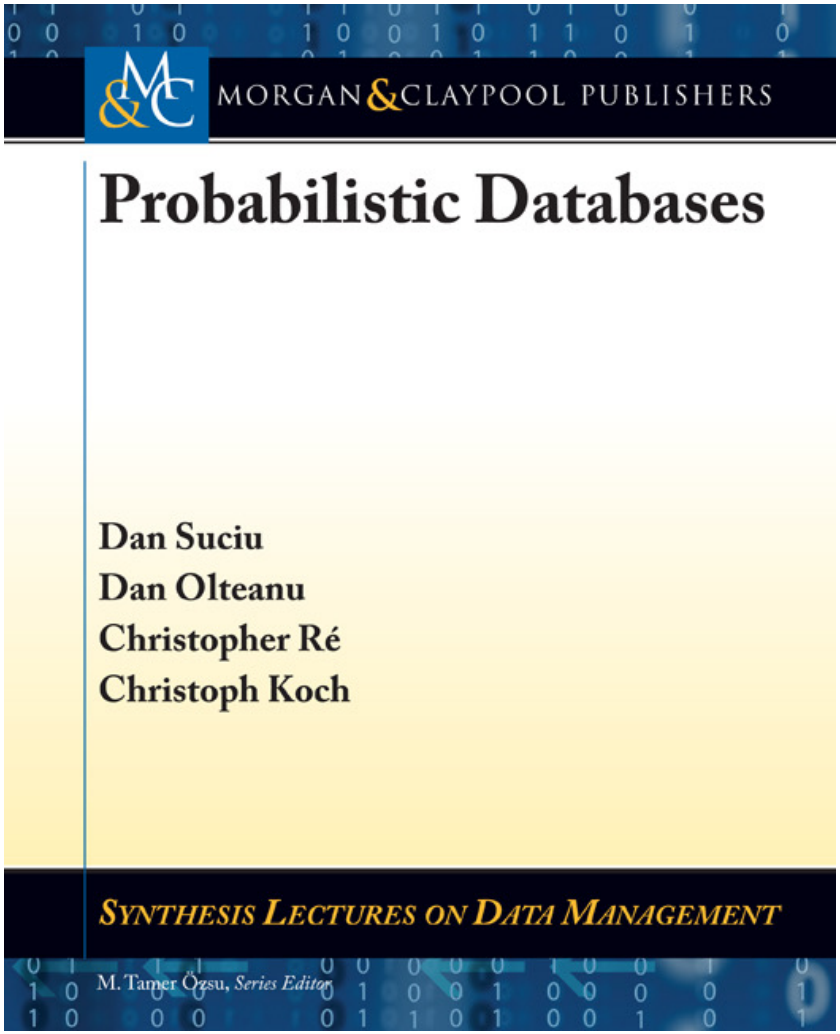
Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Danksagung

Die Präsentationen sind nach einem Tutorial von Dan Suciu gestaltet und basieren auf dem Lehrbuch Probabilistic Databases



Probabilistische Datenbanken

- Daten: Relationale Daten plus Wahrscheinlichkeiten, um Grad der Unsicherheit auszudrücken

- Anfragen: SQL-Anfragen, deren Antworten annotiert sind mit Ausgabewahrscheinlichkeiten

- Ermöglicht Ihnen einen neuen Blick auf beides, Datenbanken und Wahrscheinlichkeiten

Beispiel 1: Informationsextraktion

```
52-A Goregaon West Mumbai 400 076
```

Standard DB: Speichere nur wahrscheinlichste Extraktion

Id	House_no	Area	City	Pincode	Prob
1	52	Goregaon West	Mumbai	400 062	0.1
1	52-A	Goregaon	West Mumbai	400 062	0.2
1	52-A	Goregaon West	Mumbai	400 062	0.5
1	52	Goregaon	West Mumbai	400 062	0.2

Probabilistische DB: Speichert die meisten/alle Extraktionen um Recall zu erhöhen

Kernidee: Wahrscheinlichkeiten gegeben durch Extraktion korrelieren mit der Präzision der Extraktion

Gupta, Sarawagi: Creating Probabilistic Databases from Information Extraction Models. VLDB **2006**

Beispiel 2: Modellierung fehlender Daten

id	age	edu	inc	nw
t_1	20	HS	?	?
t_2	20	BS	50K	100K
t_3	20	?	50K	?
t_4	20	HS	100K	500K
t_5	20	?	?	?
t_6	20	HS	50K	100K
t_7	20	HS	50K	500K
t_8	?	HS	?	?
t_9	30	BS	100K	100K
t_{10}	30	?	100K	?
t_{11}	30	HS	?	?
t_{12}	30	MS	?	?
t_{13}	40	BS	100K	100K
t_{14}	40	HS	?	?
t_{15}	40	BS	50K	500K
t_{16}	40	HS	?	500K
t_{17}	40	HS	100K	500K

Standard-DB: NULL

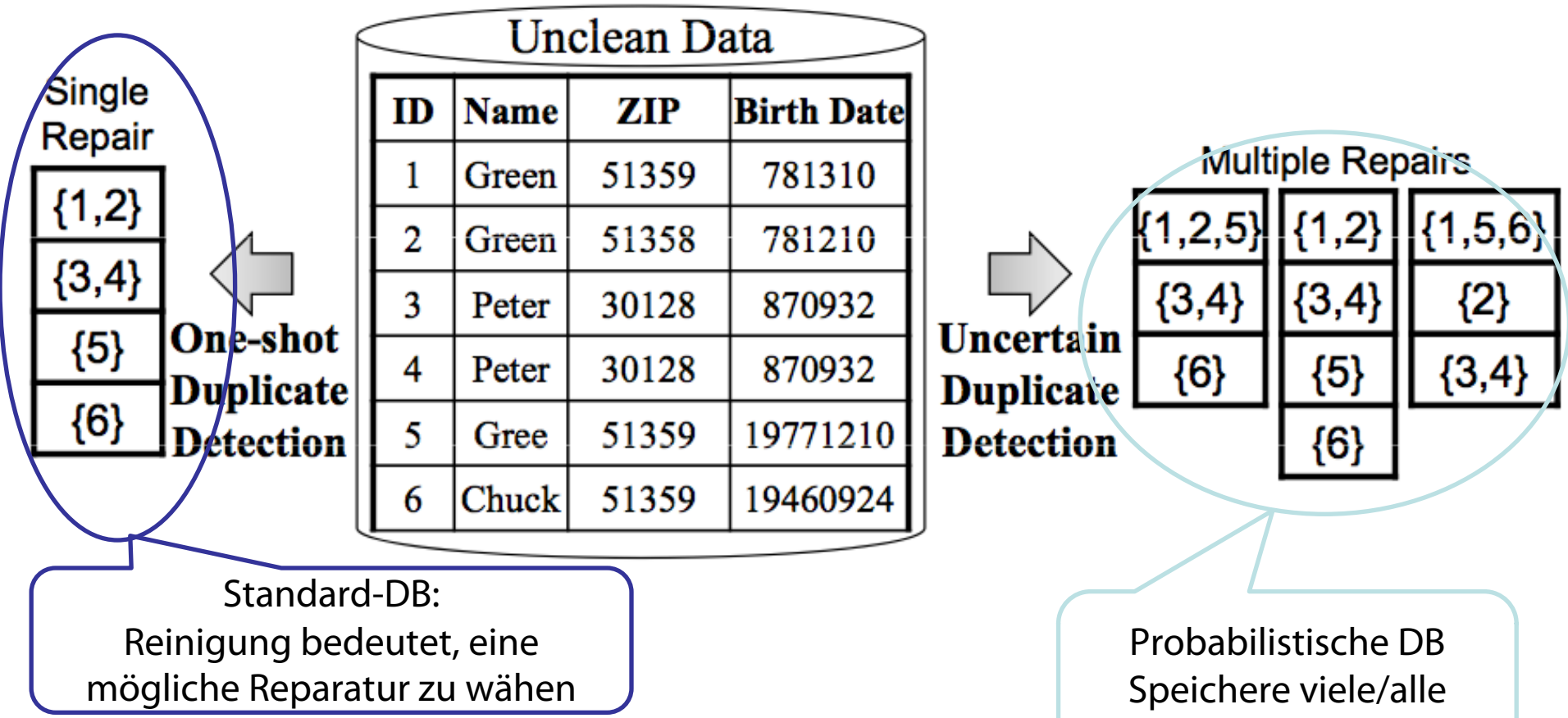
Kernidee:
Inferiere Verteilung für fehlende Daten.

Probabilistische DB:
Verteilung auf mögl. Werten

id	age	edu	inc	nw	prob
$t_{12}.1$	30	MS	50K	100K	0.30
$t_{12}.2$	30	MS	50K	500K	0.45
$t_{12}.3$	30	MS	100K	100K	0.10
$t_{12}.4$	30	MS	100K	500K	0.15

Stoyanovich, Davidson, Milo, Tannen: Deriving probabilistic databases with inference ensembles. ICDE **2011**

Beispiel 3: Datenbereinigung



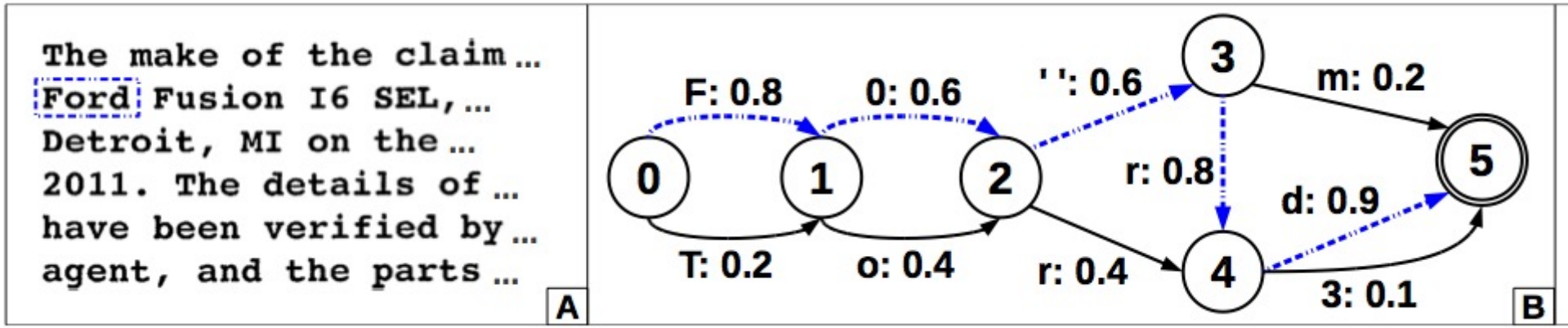
Standard-DB:
Reinigung bedeutet, eine mögliche Reparatur zu wähen

Probabilistische DB
Speichere viele/alle möglichen Reparaturen

Herausforderung: Representation von multiplen Reparaturen

Beskales, Soliman, Ilyas, Ben-David: Modeling and Querying Possible Repairs in Duplicate Detection. PVLDB **2009**

Beispiel 4: OCR



Verwendung von OCRopus von Google Books: Ausgabe ist stochastischer Automat
Üblicherweise wird nur Maximum a posteriori Probability (MAP) gespeichert
Mit probabilistischer Databasis: Speicherung verschiedener Möglichkeiten: Erhöhe Recall

```
SELECT DocId, Loss
FROM Claims
WHERE Year = 2011
    AND DocData LIKE '%Ford%';
```

Kumar, Ré: Probabilistic Management of OCR Data using an RDBMS. PVLDB **2011**

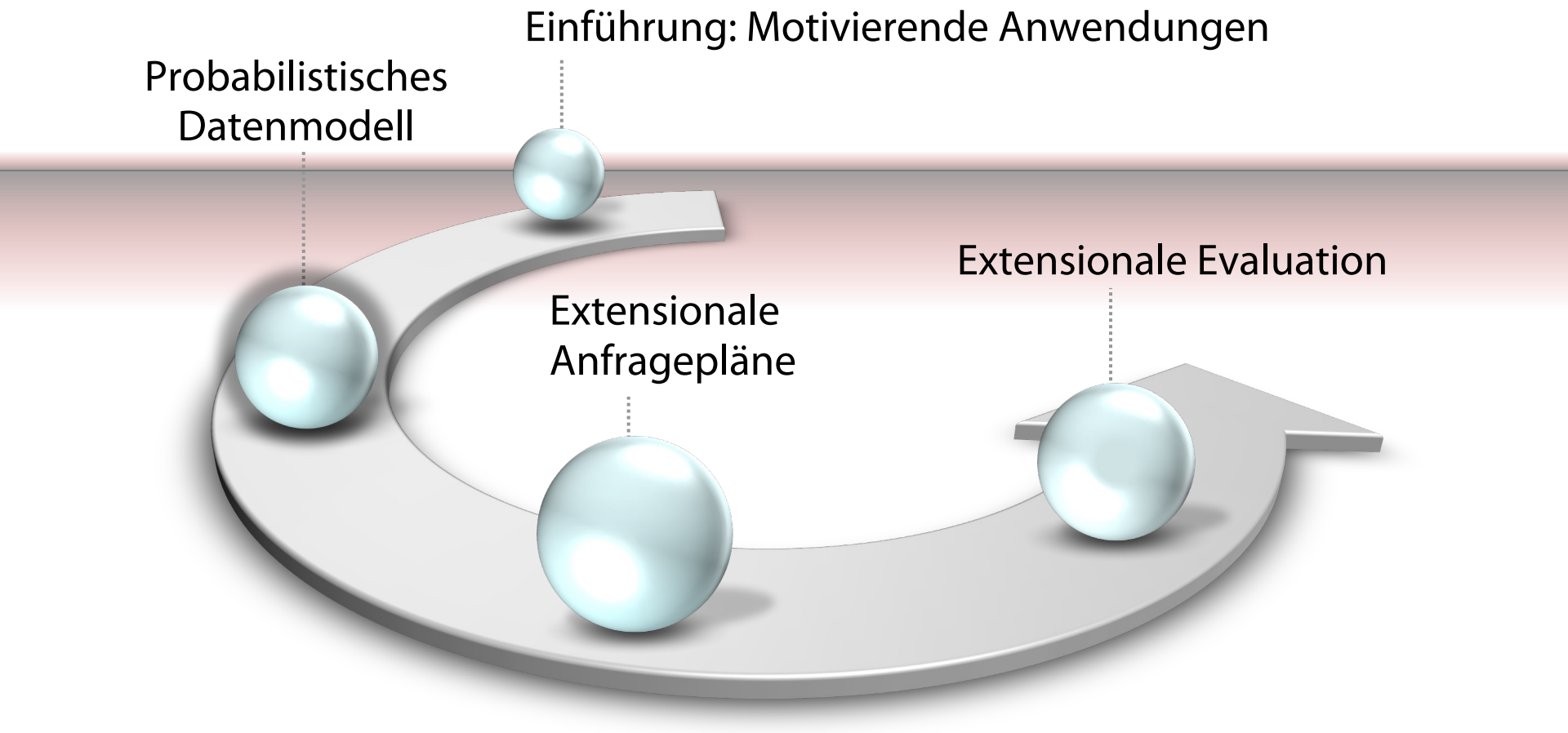
Zusammenfassung der Anwendungen

- Strukturierte, aber unsichere Daten
- Modelliert als probabilistische Daten
- Antworten für SQL queries annotiert mit Wahrscheinlichkeiten

Probabilistische Datenbank:
- Kombination aus Standard-Datenmanagement mit probabilistischer Inferenz

Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Wiederholung: Relationales Datenmodell

Daten:
gespeichert in Relationen (= Tabellen)

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Wiederholung: Relationales Datenmodell

Daten:
gespeichert in Relationen (= Tabellen)

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Anfragen: SQL,

Find all owners of objects in the Office

```
-- SQL: z.B. Postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

Wiederholung: Relationales Datenmodell

Daten:
gespeichert in Relationen (= Tabellen)

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Anfragen: SQL,

Vereinigung konjunktiver Anfragen
Unions of Conjunctive Queries (UCQs)

Find all owners of objects in the Office

```
-- SQL: z.B. Postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

$$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,y), y=\text{'Office'}$$

NB x,t sind existenzquantifiziert:

$$Q(z) = \exists x \exists t\ (\text{Owner}(z,x), \text{Location}(x,t,\text{'Office'}))$$

Wiederholung: Relationales Datenmodell

Daten:
gespeichert in Relationen (= Tabellen)

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Anfragen: SQL,

Vereinigung konjunktiver Anfragen
Unions of Conjunctive Queries (UCQs)

Find all owners of objects in the Office

```
-- SQL: z.B. Postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

$Q(z) = Owner(z,x), Location(x,t,y), y='Office'$

NB x,t sind existenzquantifiziert:

$Q(z) = \exists x \exists t\ (Owner(z,x), Location(x,t,'Office'))$

Antwort: Q=

Name
Joe
Jim

Wiederholung: Relationales Datenmodell

Daten:
gespeichert in Relationen (= Tabellen)

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Anfragen: SQL,

Vereinigung konjunktiver Anfragen
Unions of Conjunctive Queries (UCQs)

Find all owners of objects in the Office

```
-- SQL: z.B. Postgres

SELECT DISTINCT Owner.name
FROM Owner, Location
WHERE Owner.object = Location.object
    and  Location.loc = 'Office'
```

$Q(z) = Owner(z,x), Location(x,t,y), y='Office'$

NB x,t sind existenzquantifiziert:

$Q(z) = \exists x \exists t\ (Owner(z,x), Location(x,t,'Office'))$

Antwort: Q=

Name
Joe
Jim

Wiederholung: Komplexität der Anfragebeantwortung

Anfrage Q, Datenbank D

- Datenkomplexität:
  fix Q, Komplexität = f(D)

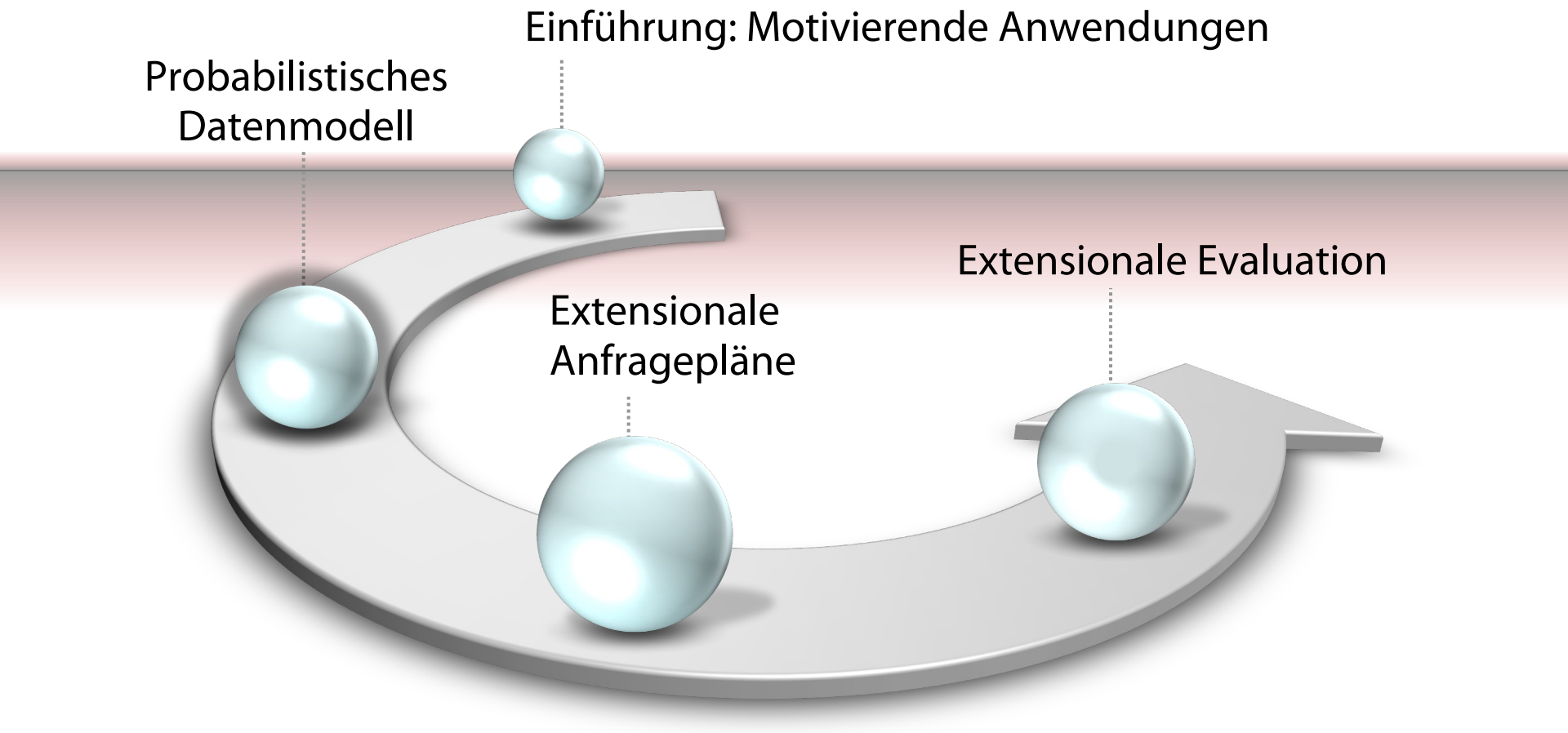
- Anfragekomplexität:
  fix D, Komplexität = f(Q)

- Kombinierte Komplexität: Komplexität = f(D,Q)

Datenkomplexität wird im Bereich der Datenbankforschung betrachtet

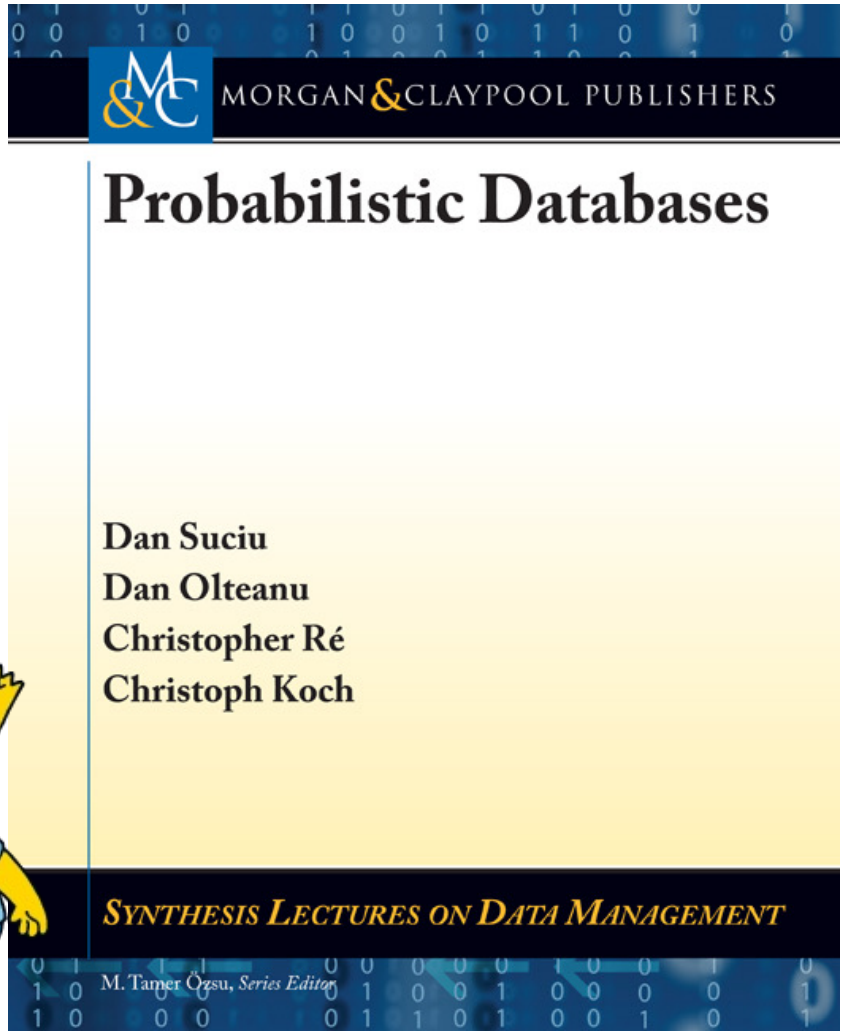
Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Danksagung

Die Präsentationen sind nach einem Tutorial von Dan Suciu gestaltet und basieren auf dem Lehrbuch Probabilistic Databases



Unvollständige Datenbank

Definition Eine unvollständige Datenbank ist eine endliche Menge von Datenbankinstanzen
$\mathbf{W} = (W_1, W_2, \ldots, W_n)$

Jedes W_i heißt mögliche Welt

Unvollständige Datenbank

Definition Eine unvollständige Datenbank ist eine endliche Menge von Datenbankinstanzen
$\mathbf{W} = (W_1, W_2, \ldots, W_n)$

Jedes W_i heißt mögliche Welt

W_1	W_2	W_3	W_4

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Unvollständige Datenbank

Definition Eine unvollständige Datenbank ist eine endliche Menge von Datenbankinstanzen
$\mathbf{W} = (W_1, W_2, \ldots, W_n)$

Jedes W_i heißt mögliche Welt

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

W_4

Unvollständige Datenbank

Definition Eine unvollständige Datenbank ist eine endliche Menge von Datenbankinstanzen
$\mathbf{W} = (W_1, W_2, \ldots, W_n)$

Jedes W_i heißt mögliche Welt

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

Owner

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Unvollständige Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine unvollständige DB **W**:

- Eine Antwort t ist sicher (certain), falls $\forall W_i,\ t \in Q(W_i)$
- Eine Antwort t ist möglich (possible) falls $\exists W_i,\ t \in Q(W_i)$

Unvollständige Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine unvollständige DB **W**:

- Eine Antwort t ist sicher (certain), falls $\forall W_i,\ t \in Q(W_i)$
- Eine Antwort t ist möglich (possible) falls $\exists W_i,\ t \in Q(W_i)$

$$Q(z) = \text{Owner}(z,x), \text{Location}(x,t,\text{'Office'})$$

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Unvollständige Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine unvollständige DB **W**:

- Eine Antwort t ist sicher (certain), falls $\forall W_i,\ t \in Q(W_i)$
- Eine Antwort t ist möglich (possible) falls $\exists W_i,\ t \in Q(W_i)$

Q(z) = Owner(z,x),
Location(x,t,'Office')

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Unvollständige Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, und eine unvollständige Datenbasis **W**:

- Eine Antwort t heißt sicher (certain), falls $\forall W_i$: $t \in Q(W_i)$
- Eine Antwort t heißt possible, falls $\exists W_i$: $t \in Q(W_i)$

Q(z) = Owner(z,x),
Location(x,t,'Office')

Sichere Antwort für Q: Joe
Mögliche Antwort f. Q: Joe, Jim

W_1

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Probabilistische Datenbank

Definition Eine probabilistische DB ist ein Tupel (**W**, P), wobei **W** eine unvollständige DB und P: **W** → [0,1] eine Wahrscheinlichkeitsverteilung ist: $\Sigma_{i=1,n} P(W_i) = 1$

Probabilistische Datenbank: Distribution Semantics

Definition Eine probabilistische DB ist ein Tupel (**W**, P), wobei **W** eine unvollständige DB und P: **W** → [0,1] eine Wahrscheinlichkeitsverteilung ist: $\Sigma_{i=1,n} P(W_i) = 1$

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3 — 0.2

Owner

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Sato, T.: A statistical learning method for logic programs with distribution semantics. Proc. of the 12th Int. Conf. on Logic Programming (ICLP95), Tokyo, pp.715–729, **1995**.

Probabilistische Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine probabilistische DB (**W**,P):

- Die Randwahrscheinlichkeit einer Antwort t ist:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i) \}$

Probabilistische Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine probabilistische DB (**W**,P):

- Die Randwahrscheinlichkeit einer Antwort t ist:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i) \}$

Q(z) = Owner(z,x),
Location(x,t,'Office')

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

W_3 — 0.2

Owner

Name	Object
Jim	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Probabilistische Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine probabilistische DB (**W**,P):

- Die Randwahrscheinlichkeit einer Antwort t ist:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i) \}$

Q(z) = Owner(z,x),
Location(x,t,'Office')

W_1 — 0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2 — 0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3 — 0.2

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4 — 0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Probabilistische Datenbank: Anfragesemantik

Definition Gegeben eine Anfrage Q, eine probabilistische DB (**W**,P):

- Die Randwahrscheinlichkeit einer Antwort t ist:
  $P(t) = \Sigma \{ P(W_i) \mid W_i \in \mathbf{W}, t \in Q(W_i) \}$

Q(z) = Owner(z,x),
Location(x,t,'Office')

P(Joe) = 1.0
P(Jim) = 0.4

W_1

0.3

Owner

Name	Object
Joe	Book302
Joe	Laptop77
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

W_2

0.4

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Book302	8:18	Office

Q=

Joe

W_3

0.2

Owner

Name	Object
Joe	Laptop77

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office

Q=

Joe

W_4

0.1

Owner

Name	Object
Joe	Book302
Jim	Laptop77
Fred	GgleGlass

Location

Object	Time	Loc
Laptop77	5:07	Hall
Laptop77	9:05	Office
Book302	8:18	Office

Q=

Joe
Jim

Diskussion

- Intuition: Eine probabilistische Datenbank sagt aus, dass eine Datenbank in einem von verschiedenen möglichen Zuständen ist. Jeder Zustand hat eine Wahrscheinlichkeit

- Mögliche Anfrageantworten: Eine Menge von Antworten, annotiert mit Wahrscheinlichkeiten:

  $(t_1, p_1), (t_2, p_2), (t_3, p_3), \ldots$

  Üblicherweise: $p_1 \geq p_2 \geq p_3 \geq \ldots$

- Problem: Die Anzahl der möglichen Welten in einer probabilistischen Datenbank ist sehr groß.

- Ziel: Anfragebeantwortung ohne explizite Generierung aller möglichen Welten (eventuell Einschränkungen in der Ausdrucksstärke hinnehmen)

Unabhängige und disjunkte Tupel

Definition Gegeben eine probabilistische DB (**W**, P).
Zwei Tupel t_1, t_2 heißen:

- unabhängig, falls: $P(t_1\ t_2) = P(t_1)\ P(t_2)$
- disjunkt (or exklusiv), falls: $P(t_1\ t_2) = 0$

Unabhängige und disjunkte Tupel

Definition Gegeben eine probabilistische DB (**W**, P).
Zwei Tupel t_1, t_2 heißen:

- unabhängig, falls: $P(t_1\ t_2) = P(t_1)\ P(t_2)$
- disjunkt (or exklusiv), falls: $P(t_1\ t_2) = 0$

Definition Eine probabilistische DB heißt *block-unabhängig-disjunkt* (BUD), falls die Tupel in Blöcke gruppiert werden können, so dass:

- Tupel vom gleichen Block disjunkt sind
- Tupel von verschiedenen Blöcken unabhängig sind.

Beispiel: BUD-Tabelle

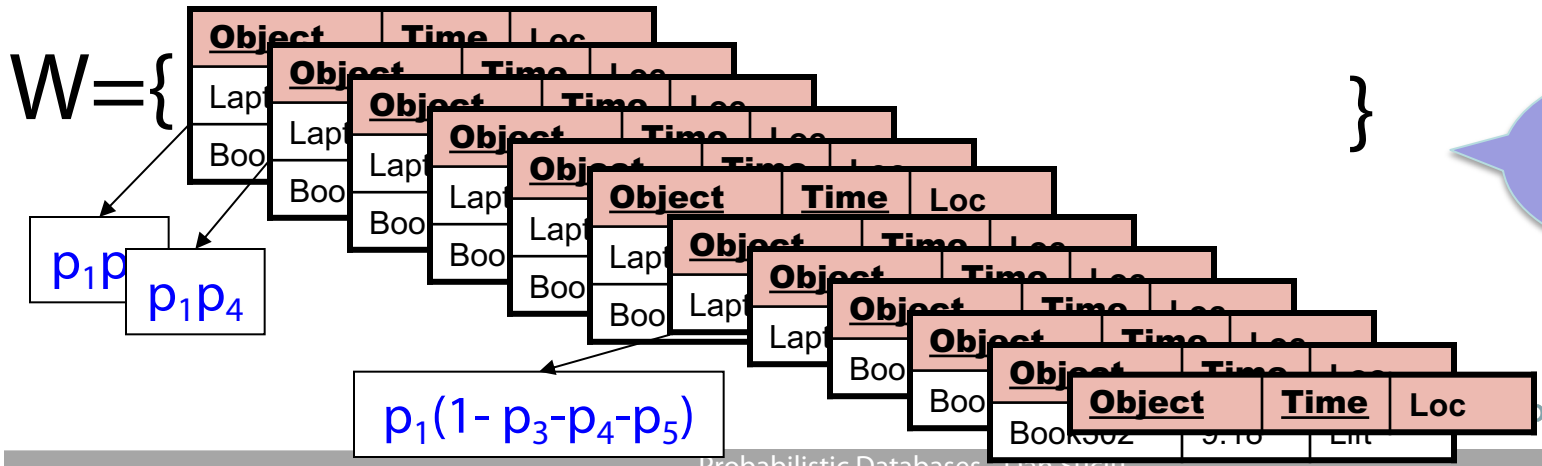
BUD Tabelle

Object	Time	Loc	P
Laptop77	9:07	Rm444	p_1
Laptop77	9:07	Hall	p_2
Book302	9:18	Office	p_3
Book302	9:18	Rm444	p_4
Book302	9:18	Lift	p_5

disjunkt

disjunkt

Unab-hängig



Mögliche Welten

Das Anfrage-Evaluationsproblem

Gegeben: BUD-Datenbank D, Anfrage Q, Ausgabetupel t

Berechne: P(t)

NB: D habe, sagen wir, 1.000.000 Tupel,
dann ist die Anzahl der möglichen Welten: $2^{1.000.000}$

Herausforderung: Berechne P(t) effizient, in der Größe von D

Datenkomplexität: die Komplexität von P
hängt von D, der Anzahl der Datenelemente, ab.

Ein Beispiel

Boolesche Anfrage:
Join-Tupel vorhanden?

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1 - \{1 - p1 * [\ 1-(1-q1)*(1-q2)\ ]\} *$$

$$\{1 - p2 * [\ 1-(1-q3)*(1-q4)*(1-q5)\ ]\}$$

Man kann P(Q) in PTIME
bzgl. der Größe der DB D bestimmen

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Zusammenfassung: Das probabilistische Datenmodell

- Mögliche-Welten-Semantik:
  Mächtig, aber schwierig zu repräsentieren

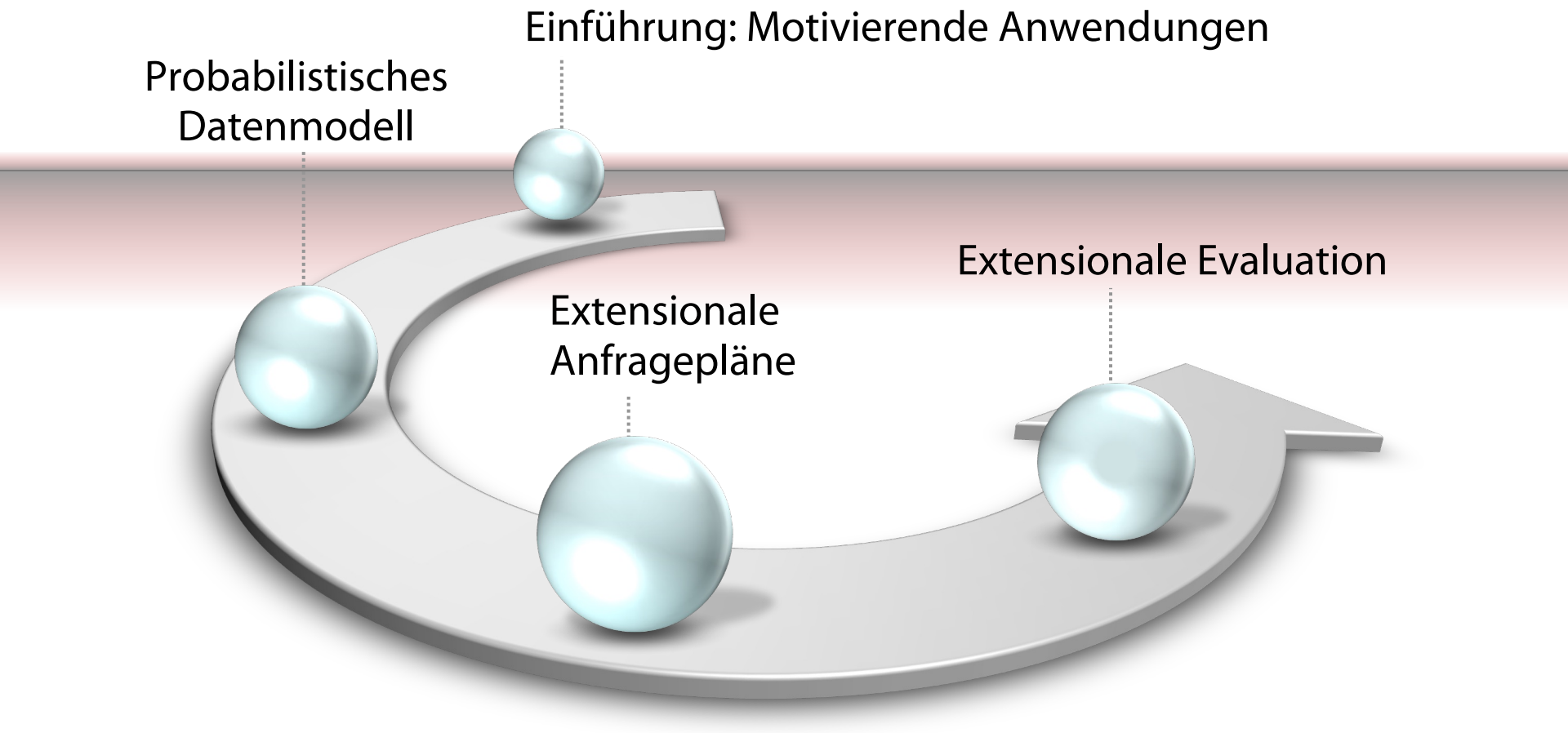
- Block-unabhängig-disjunkte Datenbasen haben effiziente Repräsentationen:
  D wird in traditioneller DB gespeichert

- Unabhängige Datenbasen: noch einfacher

Herausforderung: evaluiere Q effizient
bzgl. der Größe von D

Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Relationale Algebra

1. Verbund (join) $\bowtie$
2. Projektion (mit Duplikat-Elimination) Π
3. Vereinigung $\cup$
1. Auswahl (selection) σ
2. Differenz: hier nicht verwendet $-$

Wiederholung: Anfragebearbeitungspläne

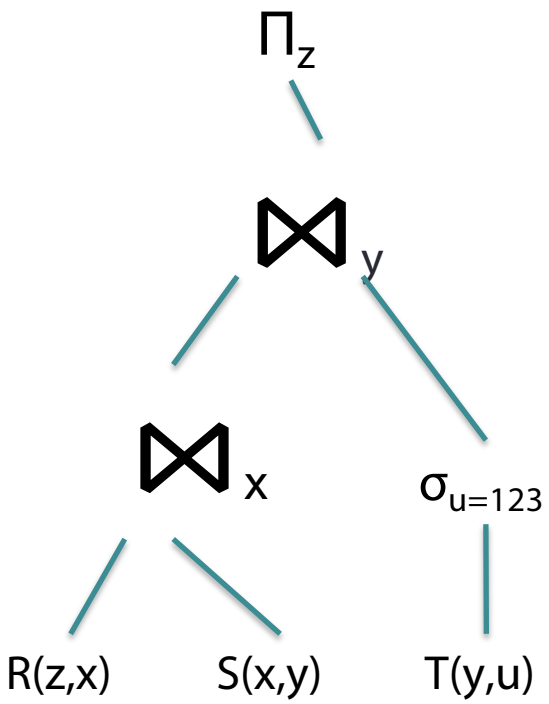
```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

$$Q(z) = R(z,x), S(x,y), T(y,u), u=123$$

Wiederholung: Anfragebearbeitungspläne

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

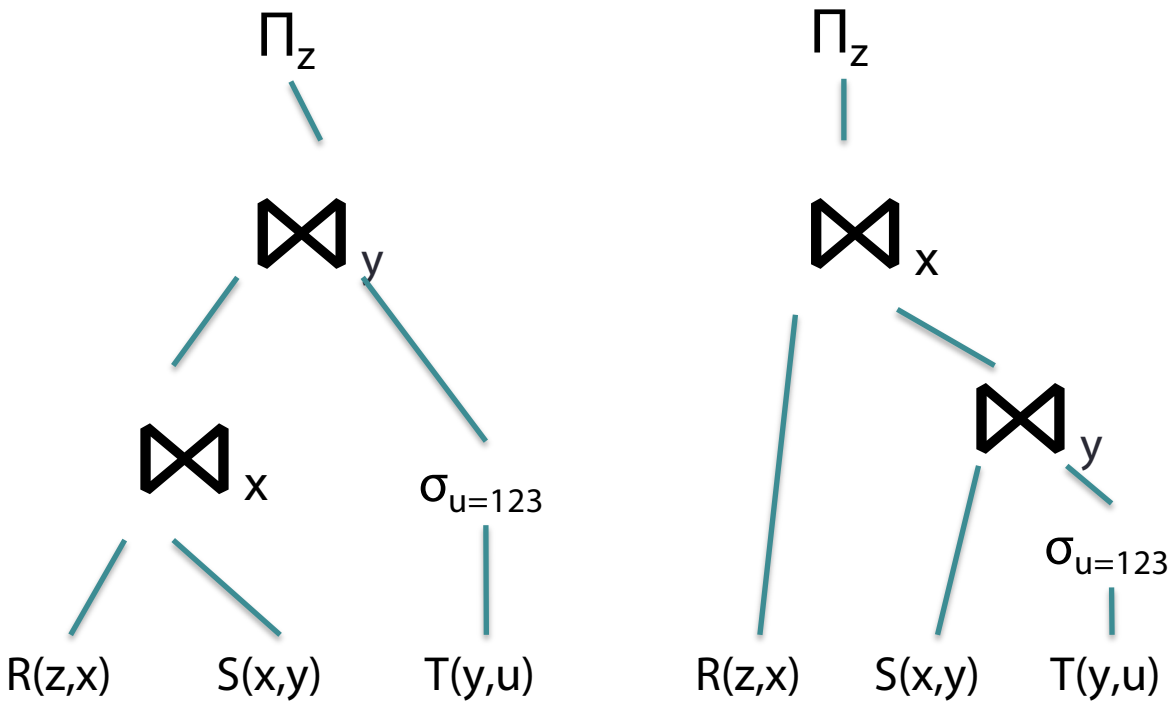
$Q(z) = R(z,x), S(x,y), T(y,u), u=123$



Wiederholung: Anfragebearbeitungspläne

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

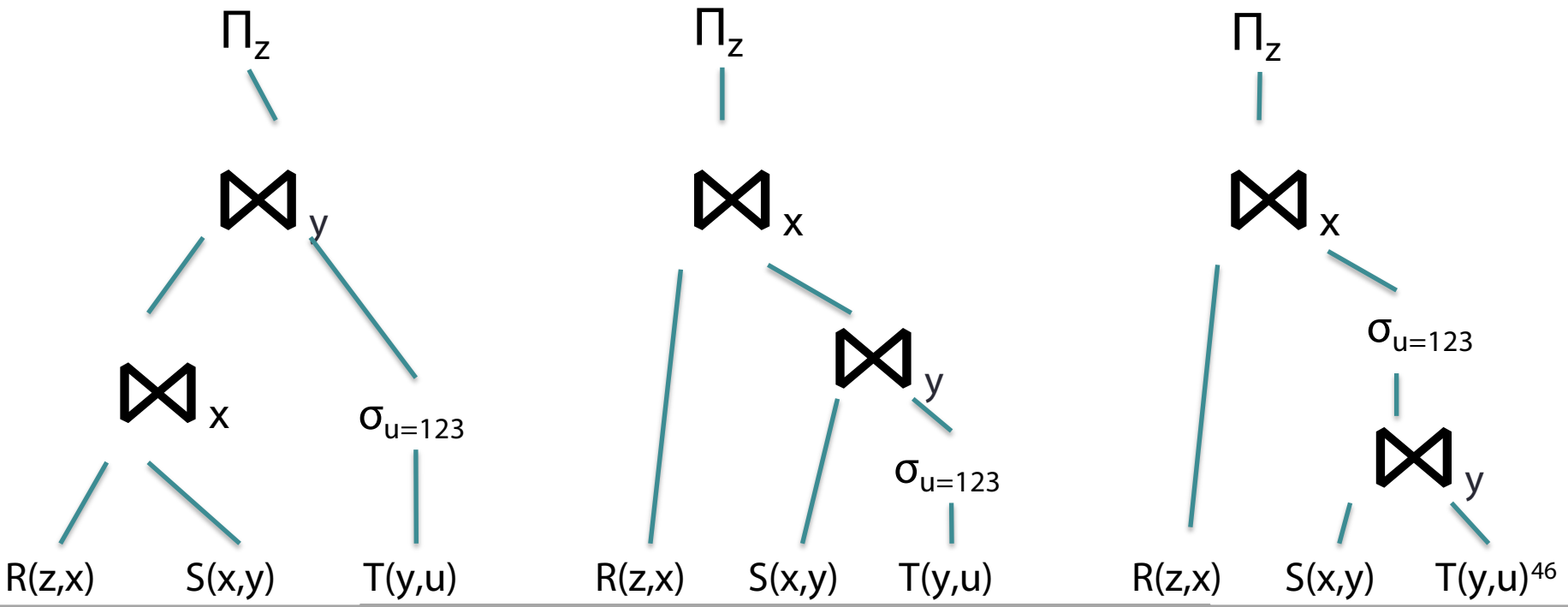
$Q(z) = R(z,x), S(x,y), T(y,u), u=123$



Wiederholung: Anfragebearbeitungspläne

```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

$Q(z) = R(z,x), S(x,y), T(y,u), u=123$



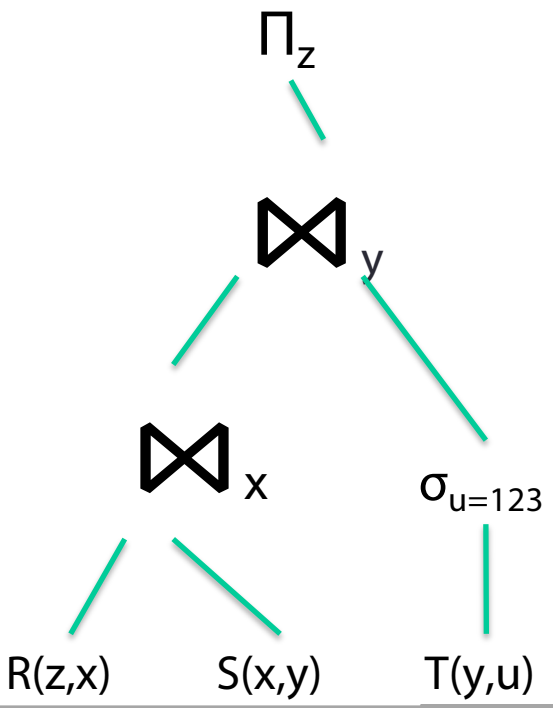
Wiederholung: Anfragebearbeitungspläne

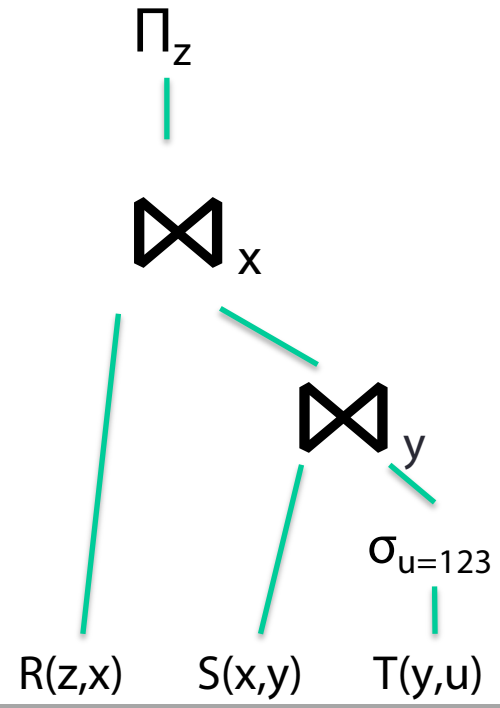
```
SELECT DISTINCT R.z
FROM R, S, T
WHERE R.x = S.x
      and S.y=T.y
      and T.u = 123
```

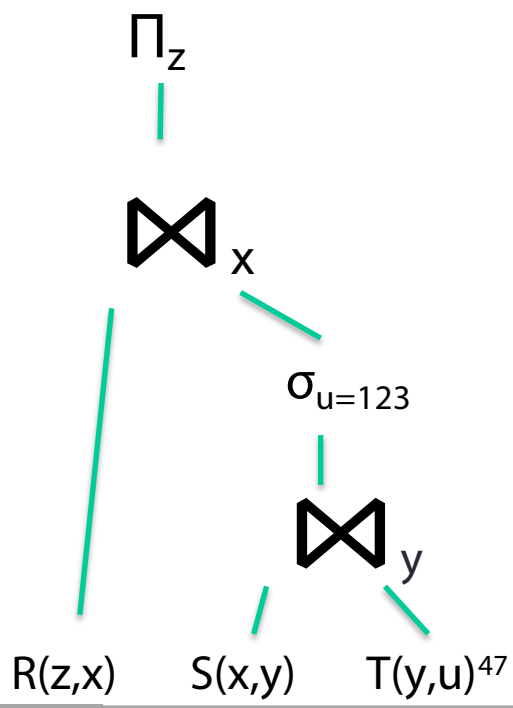
$Q(z) = R(z,x), S(x,y), T(y,u), u=123$

Diese Pläne sind äquivalent (liefern gleiche Ergebnisse)

Der Anfrageoptimierer wählt den Plan mit den geschätzt geringsten Kosten







Extensionale Pläne

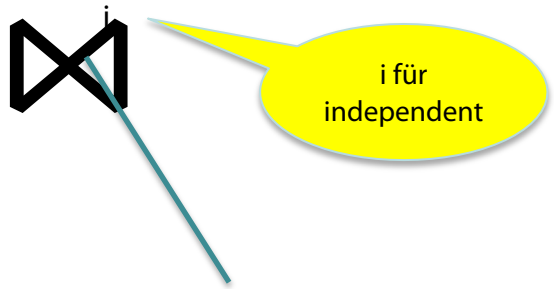
- Kernidee:
  - Modifiziere jeden Operator, sodass Wahrscheinlichkeiten für die Ausgabe berechnet werden

- Annahmen notwendig:
  - Ereignisse sind
    - unabhängig oder
    - disjunkt (exklusive)

Extensionale Operatoren

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5



R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Extensionale Operatoren

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

i für independent

Independent project

A	P
a1	1 - (1-q1)*(1-q2)
a2	1 - (1-q3)*(1-q4)*(1-q5)

i für independent

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Extensionale Operatoren

Independent join

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

i für independent

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Independent project

A	P
a1	1 - (1-q1)*(1-q2)
a2	1 - (1-q3)*(1-q4)*(1-q5)

Π^{i}_{A}

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

i für independent

Selection

A	B	P
a2	b2	q3
a2	b3	q4
a2	b2	q5

$\sigma_{A=a2}$

S(A,B)

A	B	P
a1	b1	q1
a1	b1	q2
a2	b2	q3
a2	b3	q4
a2	b2	q5

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))]$
$*[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]$

Beispiel

R

x	P
a_1	p_1
a_2	p_2
a_3	p_3

S

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

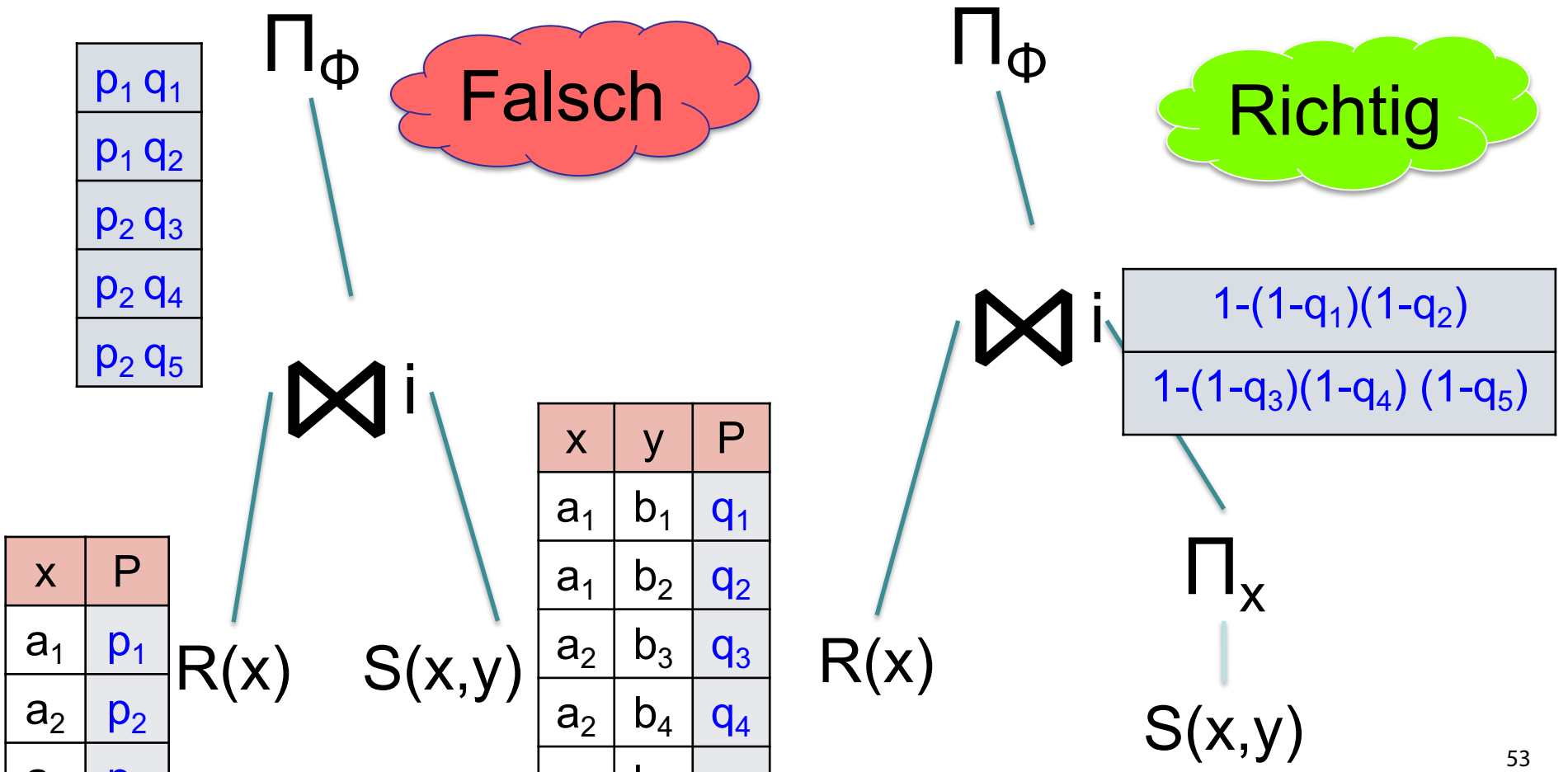
$P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] *[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]$

$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$

$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}* \{1-p_2[1-(1-q_3)(1-q_4)\ (1-q_5)]\}$

Falsch

Richtig



p_1	q_1
p_1	q_2
p_2	q_3
p_2	q_4
p_2	q_5

x	P
a_1	p_1
a_2	p_2
a_3	p_3

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

$1-(1-q_1)(1-q_2)$
$1-(1-q_3)(1-q_4)\ (1-q_5)$

Sichere Pläne

- Sei ein Schema für eine probabilist. DB gegeben
  - Relationen tupel-unabhängig oder BUD bei gegebenem Schlüssel

Definition: Ein Plan heißt *sicher*, wenn er die Wahrscheinlichkeiten für die Ausgabe richtig berechnet

- Anfrageoptimierung: Finde kostengünstigen aber sicheren Plan
- Gibt es für jede Anfrage einen sicheren Plan?

Unsichere Anfragen

R

X	P
x1	p1
x2	p2

S

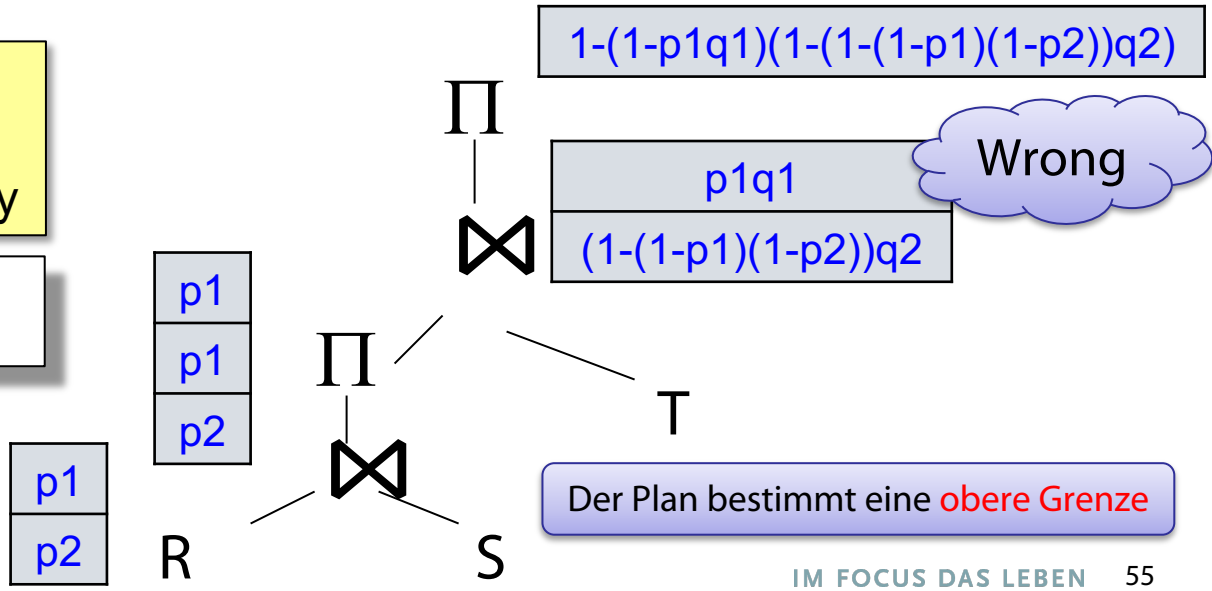
X	Y
x1	y1
x1	y2
x2	y2

T

Y	P
y1	q1
y2	q2

```
SELECT DISTINCT 'yes'
FROM R, S, T
WHERE R.x = S.x and S.y = T.y
```

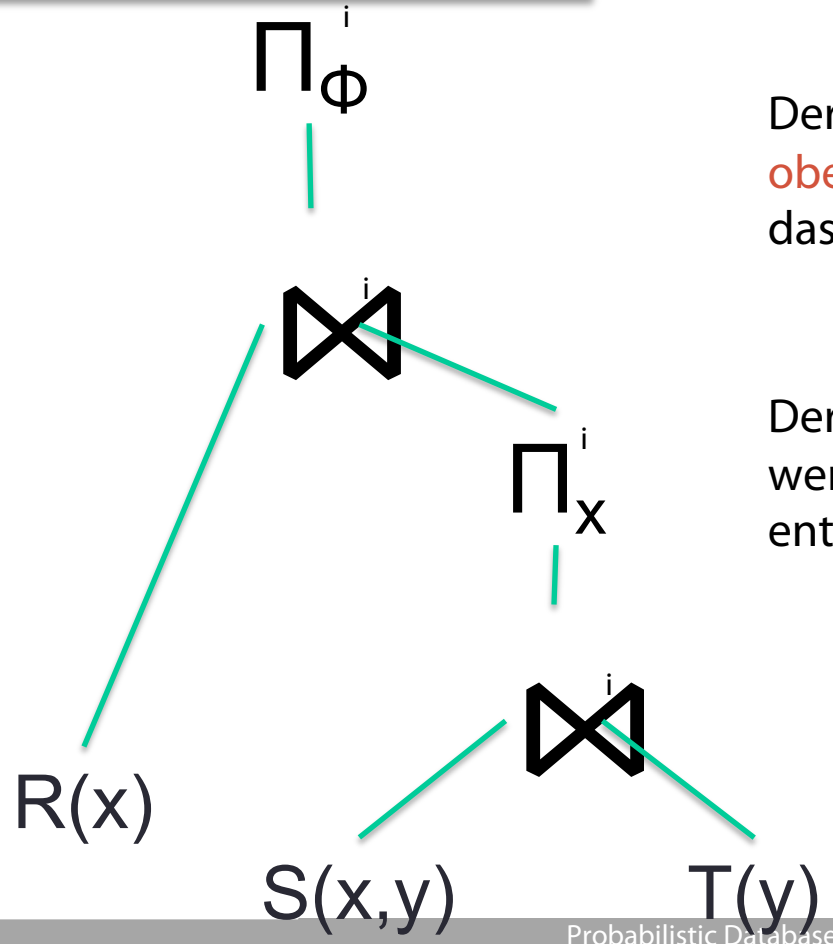
H_0 :- R(x),S(x,y),T(y)



W Gatterbauer, D Suciu, Oblivious bounds on the probability of Boolean functions, ACM Transactions on Database Systems (TODS) 39 (1), 5, **2013**

Extensionale Pläne in PostgreSQL

```
SELECT DISTINCT 'yes'
FROM R, S, T
WHERE R.x = S.x and S.y = T.y
```



Diese Anfrage ist unsicher.

Der Plan ist unsicher, aber er liefert eine obere Grenze für den Wahrscheinlichkeitswert, dass die Anfrage mit 'yes' beantwortet wird.

Der Plan generiert untere Grenze, wenn die Wahrscheinlichkeiten in T entsprechend angepasst werden

W Gatterbauer, D Suciu, Oblivious bounds on the probability of Boolean functions, ACM Transactions on Database Systems (TODS) 39 (1), 5, **2013**

Untere Grenzen

S

X	Y
x1	y1
x1	y2
x2	y2

$\bowtie^{i}$

T

Y	P
y1	q1
y2	q2

Ohne Beweis

T'

```
SELECT T.y, 1-exp((ln(1-T.p))/count(*)) AS p
  FROM S,T
  WHERE S.y=T.y
  GROUP by T.y, T.p
```

W Gatterbauer, D Suciu, Oblivious bounds on the probability of Boolean functions, ACM Transactions on Database Systems (TODS) 39 (1), 5, **2013**

Diskussion

- Sichere Anfragen haben einen sicheren Plan und können effektiv berechnet werden

- Für unsichere Anfragen kann kein sicherer Plan bestimmt werden, und es kann gezeigt werden, dass sie nicht effizient berechnet werden können

- Jeder extensionale Plan (sicher oder unsicher) kann direkt in SQL ausgedrückt werden – gezeigt am Beispiel von PostgreSQL

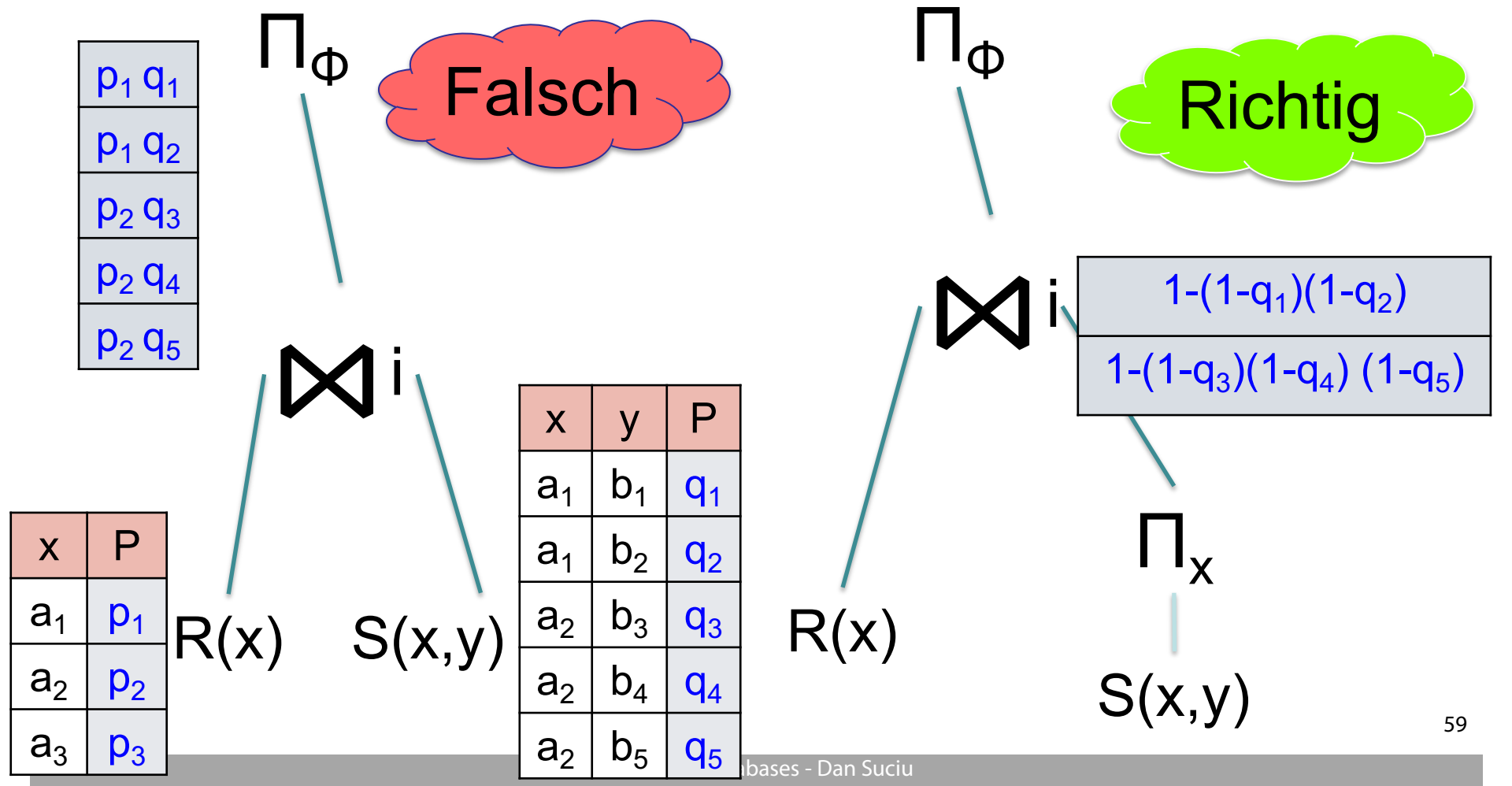
```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$P(Q) = 1 - [1-p_1*(1-(1-q_1)*(1-q_2))] *[1- p_2*(1-(1-q_3)*(1-q_4)*(1-q_5))]$

$1-(1-p_1q_1)(1-p_1q_2)(1-p_2q_3)(1-p_2q_4)(1-p_2q_5)$

$1-\{1-p_1[1-(1-q_1)(1-q_2)]\}* \{1-p_2[1-(1-q_3)(1-q_4)\ (1-q_5)]\}$



x	P
a_1	p_1
a_2	p_2
a_3	p_3

x	y	P
a_1	b_1	q_1
a_1	b_2	q_2
a_2	b_3	q_3
a_2	b_4	q_4
a_2	b_5	q_5

Warum Variante 1 nicht funktioniert

Boolesche Anfrage:
Join-Tupel vorhanden?

```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```

Q() = R(x), S(x,y)

$$P(Q) = 1 - \{1 - p1 * [\ 1-(1-q1)*(1-q2)\ ]\} *$$

$$\{1 - p2 * [\ 1-(1-q3)*(1-q4)*(1-q5)\ ]\}$$

Man kann P(Q) in PTIME
bzgl. der Größe der DB D bestimmen

R

x	P
a1	p1
a2	p2
a3	p3

S

x	y	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Obergrenzen und Untergrenzen

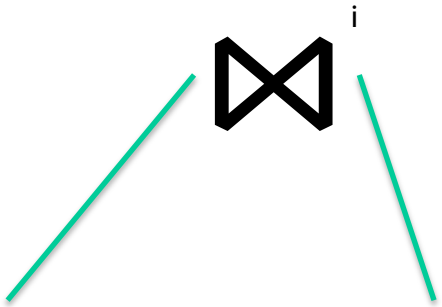
- Sichere vs. unsichere Pläne
- Unsichere Pläne berechnen Obergrenze für Wahrscheinlichkeiten der Ergebnistupel
- Man kan unsichere Pläne so transformieren, dass (auch) Untergrenzen berechnet werden können

- Heute: Anfragepläne in PostgreSQL
- Heute: Herleitung von sicheren Plänen (sofern mögl.)
- Behauptung:
  Für bestimmte Anfragen existieren nur unsichere Pläne

Extensionale Pläne in PostgreSQL

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

```
SELECT R.A, S.B, R.P*S.P
FROM    R, S
WHERE R.A=S.A
```



R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

Extensionale Pläne in PostgreSQL

A	B	P
a1	b1	p1*q1
a1	b2	p1*q2
a2	b3	p2*q3
a2	b4	p2*q4
a2	b5	p2*q5

```
SELECT R.A, S.B, R.P*S.P
FROM    R, S
WHERE R.A=S.A
```

```
SELECT S.A, 1.0-prod(1.0 - S.p)
FROM    S
GROUP BY S.A
```

A	P
a1	1 - (1-q1)*(1-q2)
a2	1 - (1-q3)*(1-q4)*(1-q5)

R(A)

A	P
a1	p1
a2	p2
a3	p3

S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

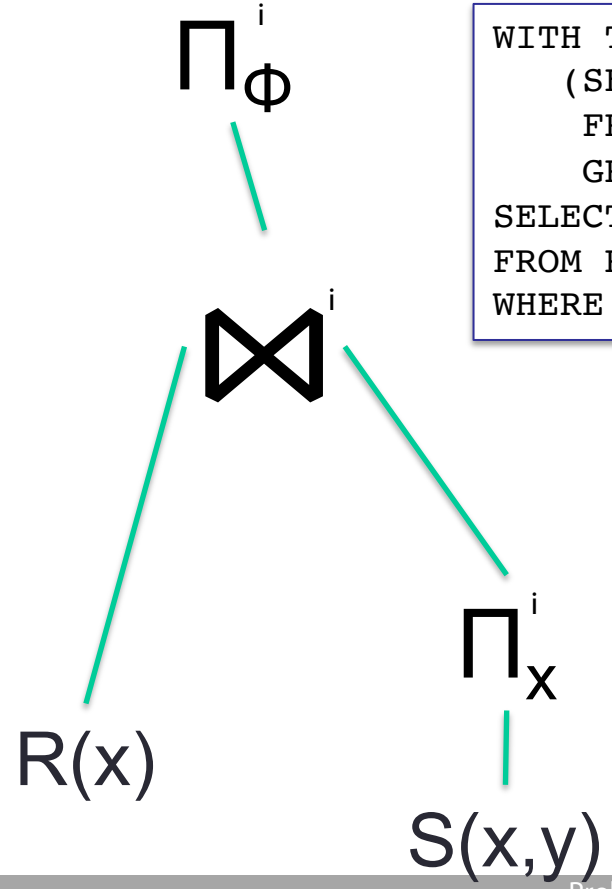
S(A,B)

A	B	P
a1	b1	q1
a1	b2	q2
a2	b3	q3
a2	b4	q4
a2	b5	q5

```
create or replace
  function combine_prod(float, float)
  returns float as
  'select $1 * $2' language SQL;
create or replace
  function final_prod(float)
  returns float as
  'select $1' language SQL;
drop aggregate if exists prod (float);
create aggregate prod(float)
(   sfunc = combine_prod,
    stype = float,
    finalfunc = final_prod,
    initcond = '1.0'
);
```

Extensionale Pläne in PostgreSQL

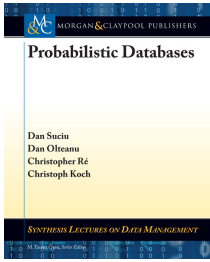
```
SELECT DISTINCT 'true'
FROM R, S
WHERE R.x = S.x
```




```
WITH Temp AS
   (SELECT S.x, 1.0-prod(1.0 - S.p) as p
    FROM S
    GROUP BY S.x)
SELECT 'true' as z, 1.0-prod(1.0 – R.P * Temp.P) as p
FROM R, Temp
WHERE R.x = Temp.x
```

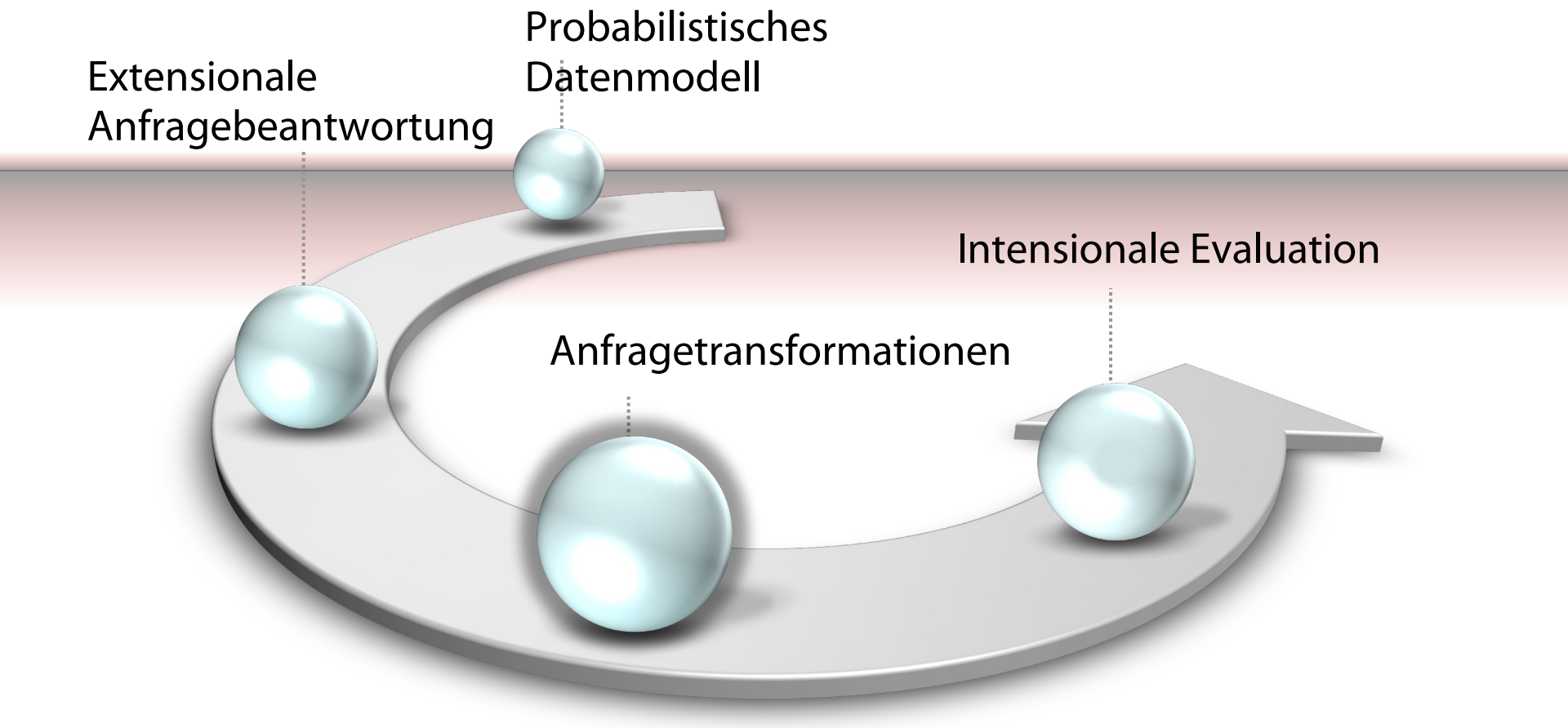
Einsichten

- Man benötigt kein neues probabilistisches DB-System für eine probabilistische Datenbasis, wenn man mit Approximationen leben kann
- Deterministische Approximationen (kein Sampling)
- Wie gut sind die?
- Es interessieren vielleicht gar nicht alle Ergebnisse, sondern nur die wahrscheinlichen
- Wir beleuchten später, wie man durch Sampling Top-k-Resultate bestimmen kann (Probability Ranking Principle)



Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Sichere Anfragen: Syntaktische Kriterien

- Für beliebige FOL-Anfragen nicht entscheidbar, ob sie sicher sind (und damit in P) oder nicht (in #P).

- Betrachte Fragmente
  - Konjunktive Anfragen (CQs) ohne Self-Joins
    Einfaches Kriterium: „Hierarchisch zu sein“ heißt sicher
  - Unions von CQs
    Kriterium: Erfolgreiche Anwendbarkeit von bestimmten Regeln (die direkt zu extensionalen Plänen führen)

Komplexitätsklassen und Dichotomie

- #P = Probleme, für die die Anzahl von Lösungen zu einer Instanz durch eine NTM in polynomieller Zeit bestimmt werden kann. (Bsp: #SAT)
- P ist in #P enthalten
- #P-vollständige Probleme: Schwierigste Probleme in #P

- Man vermutet, dass P echt enthalten ist in #P
- In diesem Fall sind zwischen P und den #P-vollständigen nachweislich weitere verschachtelte Klassen zu finden

- Dichotomietheoreme für Kriterium K: Verschachtelte Klassen werden übersprungen

N. Dalvi, D. Suciu, The dichotomy of probabilistic inference for unions of conjunctive queries, Journal of the ACM (JACM) 59 (6), 30, **2012**

Wiederholung: Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

Atom

$Q(z) = \exists x_1 \exists t_1 \, (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{”Office444”}))$

Ohne Quantoren:

$Q(z) = \text{Owner}(z,x_1), \text{Location}(x_1,t_1,\text{”Office444”})$

Eine CQ hat self-joins wenn Relationen wiederholt vorkommen

$Q(z) = R(z,x_1), S(z,x_1,y_1), T(z,x_2), S(z,x_2,y_2)$

$Q() = R(x,y), R(y,z)$

Bsp: CQ ohne self-joins

$Q() = R(x), S(x,y)$

$H_0() = R(x), S(x,y), T(y)$

Hierarchische Anfragen

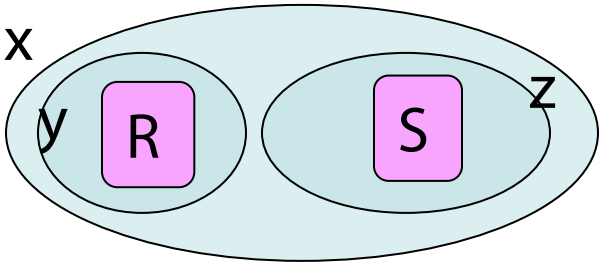
at(x) = Menge von Atomen mit Variable x

Definition Eine Anfrage Q ist hierarchisch falls für alle existentiellen Variablen x, y gilt:

$$at(x) \supseteq at(y) \quad \text{oder} \quad at(y) \supseteq at(x) \quad \text{oder} \quad at(x) \cap at(y) = \varnothing$$

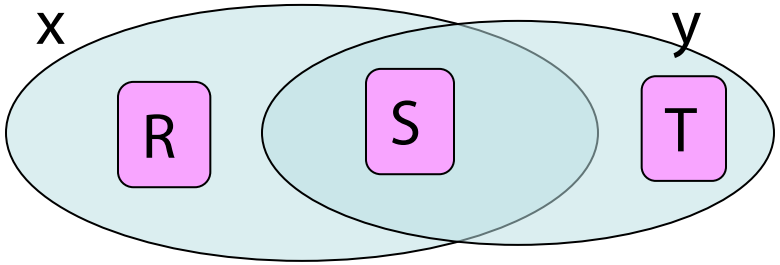
Hierarchisch

$Q = R(x,y),S(x,z)$



Nicht-hierarchisch

$H_0 = R(x), S(x, y), T(y)$



N. Dalvi, D. Suciu, The dichotomy of probabilistic inference for unions of conjunctive queries, Journal of the ACM (JACM) 59 (6), 30, **2012**

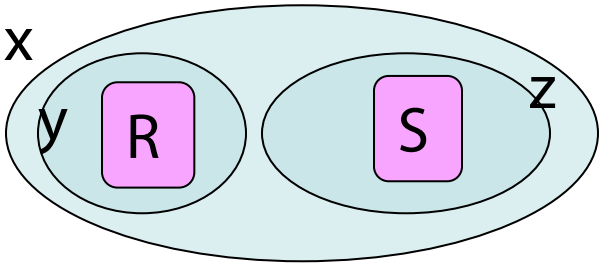
Das kleine Dichotomietheorem

Theorem [Dalvi & Suciu 0.4] Für CQs Q ohne self-joins:

- Falls Q hierarchisch, dann is Q sicher und man kann man P(Q) in PTIME berechnen
- Falls Q nicht hierarchisch, dann ist Q nicht sicher und das Problem P(Q) zu berechnen ist #P-vollständig.

Hierarchisch

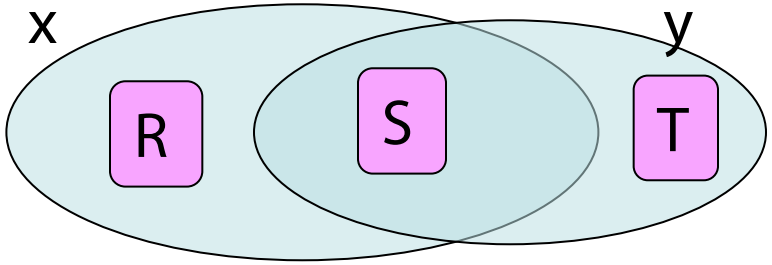
$Q = R(x,y),S(x,z)$



PTIME

Nicht-hierarchisch

$H_0 = R(x), S(x, y), T(y)$



#P

Für beliebige UCQs ist Hierarchiebedingung notwendig, aber nicht hinreichend

N. Dalvi, D. Suciu, The dichotomy of probabilistic inference for unions of conjunctive queries, Journal of the ACM (JACM) 59 (6), 30, **2012**

Wiederholung: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

Atom

$$Q(z) = \exists x_1 \exists t_1\ (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{”Office444”})) \vee$$
$$\exists x_2 \exists t_2\ (\text{Owner}(z,x_2) \wedge \text{Location}(x_2,t_2,\text{”Hall7”}))$$

Ohne Quantoren:

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{”Office444”}) \vee \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{”Hall7”})$$

Wiederholung: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

$$Q(z) = \exists x_1 \exists t_1\ (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{”Office444”})) \vee \exists x_2 \exists t_2\ (\text{Owner}(z,x_2) \wedge \text{Location}(x_2,t_2,\text{”Hall7”}))$$

Ohne Quantoren:

Union of conjunctive queries

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{”Office444”}) \vee \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{”Hall7”})$$

Wiederholung: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

$$Q(z) = \exists x_1 \exists t_1\ (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{”Office444”})) \vee \exists x_2 \exists t_2\ (\text{Owner}(z,x_2) \wedge \text{Location}(x_2,t_2,\text{”Hall7”}))$$

Ohne Quantoren:

Union of conjunctive queries

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{”Office444”}) \vee \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{”Hall7”})$$

Nach Umformung:

$$Q(z) = \text{Owner}(z,x) \wedge \exists t\ [\text{Location}(x,t,\text{”Office444”}) \vee \text{Location}(x,t,\text{”Hall7”})]$$

Wiederholung: Unions of Conjunctive Queries

Owners of items in either “Office444” or “Hall7”:

$$Q(z) = \exists x_1 \exists t_1\ (\text{Owner}(z,x_1) \wedge \text{Location}(x_1,t_1,\text{”Office444”})) \vee \exists x_2 \exists t_2\ (\text{Owner}(z,x_2) \wedge \text{Location}(x_2,t_2,\text{”Hall7”}))$$

Ohne Quantoren:

Union of conjunctive queries

$$Q(z) = \text{Owner}(z,x_1),\text{Location}(x_1,t_1,\text{”Office444”}) \vee \text{Owner}(z,x_2),\text{Location}(x_2,t_2,\text{”Hall7”})$$

Nach Umformung:

$$Q(z) = \text{Owner}(z,x) \wedge \exists t\ [\text{Location}(x,t,\text{”Office444”}) \vee \text{Location}(x,t,\text{”Hall7”})]$$

Unter Verwendung von:

1. Distributivgesetz für $\vee$, $\wedge$
2. Gesetz für $\exists$, $\vee$: $(\exists x\ P(x)) \vee (\exists y\ T(y)) = \exists z\ (P(z) \vee T(z))$

Vier Regeln, um sichere Anfragen zu erzeugen

- Independent join
- Independent project
- Independent union
- Inclusion/exclusion

"Lifted Inference:"
Vermeide die Betrachtung sämtlicher möglicher Welten

Wir beschränken uns auf Boolesche Anfragen.

Regel 1: Independent Join

$$P(Q1 \wedge Q2) = P(Q1)P(Q2)$$

Wenn Q1 und Q2 unabhängig sind
(also*) keine gemeinsamen Atome haben)

*) Hinter dem „also“ steckt ein Resultat:

Wir wollen die Umformung nur dann, wenn Q1 und Q2 probabilistisch unabhängig sind bzgl. der W-Verteilung P.

Man kann zeigen, dass das der Fall ist, wenn
Q1 und Q2 keine gemeinsamen Atome enthalten

Regel 1: Independent Join

$$P(Q1 \wedge Q2) = P(Q1)P(Q2)$$

Wenn Q1 und Q2 unabhängig sind
(also keine gemeinsamen Atome haben)

Regel 2: Independent Project

$$P(\exists z\, Q) = 1 - \Pi_{a \in Domain} (1 - P(Q[a/z])$$

Wenn z eine "Separatorvariable" in Q
ist, also für Konstanten a,b, Q[a/z]
und Q[b/z] unabhängig sind

Das ist die einzige Regel, die tatsächlich von der DB abhängt
(wegen der aktiven Domäne).
Dies führt aber nur zu einer polynomiellen Datenkomplexität
für sichere extensionale Pläne

Warum sind nicht-hierarchische Regeln schwierig?

Lifted Inference Sometimes Fails

H_0= (Smoker(x) , Friend(x,y) , Jogger(y))

No rule applies here!

The ∃ rule does not apply, because H_0[Alice/x] and H_0[Bob/x] are dependent:

H_0[Alice/x] = (Smoker(Alice) , Friend(Alice,y) , Jogger(y))
H_0[Bob/x] = (Smoker(Bob) , Friend(Bob,y) , Jogger(y))

Dependent

Theorem. [Dalvi'04] Computing P(H_0 | D) is #P-hard in |D|

Regel 1: Independent Join

$$P(Q1 \wedge Q2) = P(Q1)P(Q2)$$

Wenn Q1 und Q2 unabhängig sind
(also (modulo Unifikation)
keine gemeinsamen Atome haben)

Regel 2: Independent Project

$$P(\exists z\, Q) = 1 - \prod_{a \in Domain} (1 - P(Q[a/z]))$$

Wenn z eine "Separatorvariable" in Q
ist, also für Konstanten a,b, Q[a/z]
und Q[b/z] unabhängig sind

Regel 3: Independent Union

$$P(Q1 \vee Q2) = 1 - (1 - P(Q1))(1 - P(Q2))$$

Wenn Q1 und Q2 unabhängig sind
(also keine gemeinsamen Atome haben)

Beispiel

$Q_U = R(x_1), S(x_1,y_1) \vee T(x_2), S(x_2,y_2)$

$= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

Beispiel

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$=\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$

Kommutiere ∃ mit ∨

Beispiel

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$

Kommutiere $\exists$ mit $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[R(a) \wedge \underline{S(a,y_1)} \vee T(a) \wedge \underline{S(a,y_2)})]$

Independent project: Für $a \neq b$, sind $Q_U[a/z]$ und $Q_U[b/z]$ unabhängig weil die Atome $R(a),S(a,y_1),T(a),S(a,y_2)$ disjunkt sind von $R(b),S(b,y_1),T(b),S(b,y_2)$

Beispiel

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$=\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$

Kommutiere $\exists$ mit $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[R(a) \wedge \underline{S(a,y_1)} \vee T(a) \wedge \underline{S(a,y_2)})]$

Independent project: Für $a \neq b$, sind $Q_U[a/z]$ und $Q_U[b/z]$ unabhängig weil die Atome $R(a),S(a,y_1),T(a),S(a,y_2)$ disjunkt sind von $R(b),S(b,y_1),T(b),S(b,y_2)$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[\underline{(R(a) \vee T(a))} \wedge \underline{\exists y.\, S(a,y)}])$

Distribution $\wedge$ über $\vee$ und $\exists$

Beispiel

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$

$= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$

Kommutiere $\exists$ mit $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[R(a) \wedge \underline{S(a,y_1)} \vee T(a) \wedge \underline{S(a,y_2)})]$

Independent project: Für $a \neq b$, sind $Q_U[a/z]$ und $Q_U[b/z]$ unabhängig weil die Atome $R(a),S(a,y_1),T(a),S(a,y_2)$ disjunkt sind von $R(b),S(b,y_1),T(b),S(b,y_2)$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[\underline{(R(a) \vee T(a))} \wedge \underline{\exists y.\, S(a,y)}])$

Distribution $\wedge$ über $\vee$ und $\exists$

$P(Q_U) = 1 - \Pi_{a \in Domain}\, (1 - P[R(a) \vee T(a)]\; P[\exists y.\, S(a,y)])$

Independent join

Beispiel

$Q_U = R(x_1),S(x_1,y_1) \vee T(x_2),S(x_2,y_2)$ $= \exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1) \vee \exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)$

$Q_U = \exists z\, [R(z) \wedge S(z,y_1) \vee T(z) \wedge S(z,y_2)]$ Kommutiere $\exists$ mit $\vee$

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[R(a) \wedge \underline{S(a,y_1)} \vee T(a) \wedge \underline{S(a,y_2)})]$

Independent project: Für $a \neq b$, sind $Q_U[a/z]$ und $Q_U[b/z]$ unabhängig weil die Atome $R(a),S(a,y_1),T(a),S(a,y_2)$ disjunkt sind von $R(b),S(b,y_1),T(b),S(b,y_2)$

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[\underline{(R(a) \vee T(a))} \wedge \underline{\exists y.\, S(a,y)}])$

Distribution $\wedge$ über $\vee$ und $\exists$

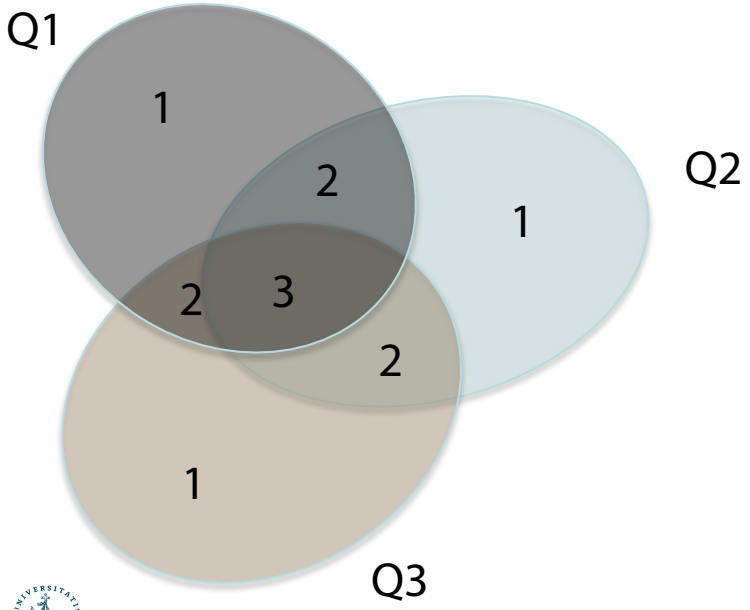
$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - P[R(a) \vee T(a)]\, P[\exists y.\, S(a,y)])$

Independent join

$P(Q_U) = 1 - \Pi_{a \in Domain} (1 - (1-(1-P[R(a)])(1-P[T(a)]))\, (1-\Pi_{b \in Domain} (1 - P[S(a,b)])))$

Regel 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$

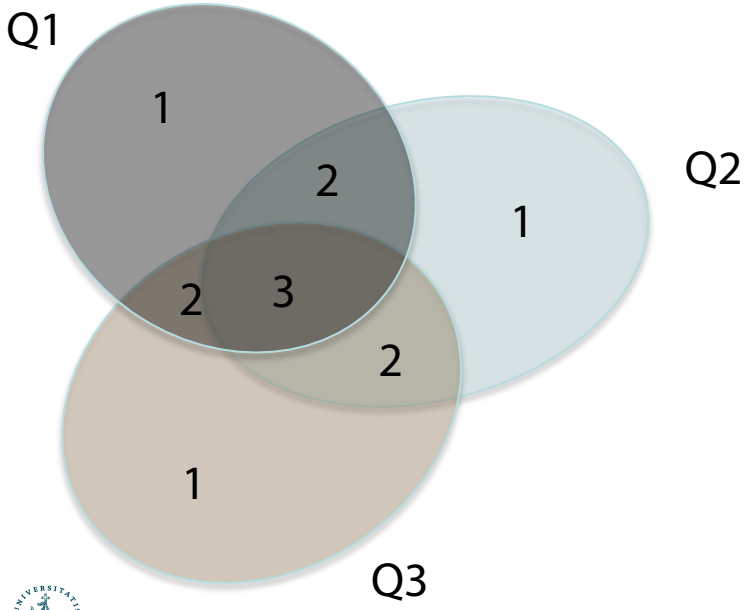


NB: Dieses ist dual zur häufiger verwendeten Formel:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

Regel 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$

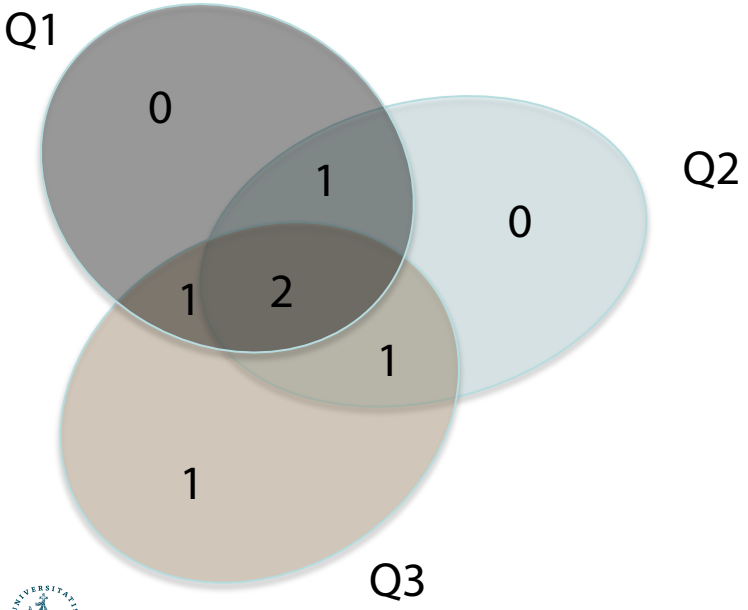


NB: Dieses ist dual zur häufiger verwendeten Formel:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

Regel 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$

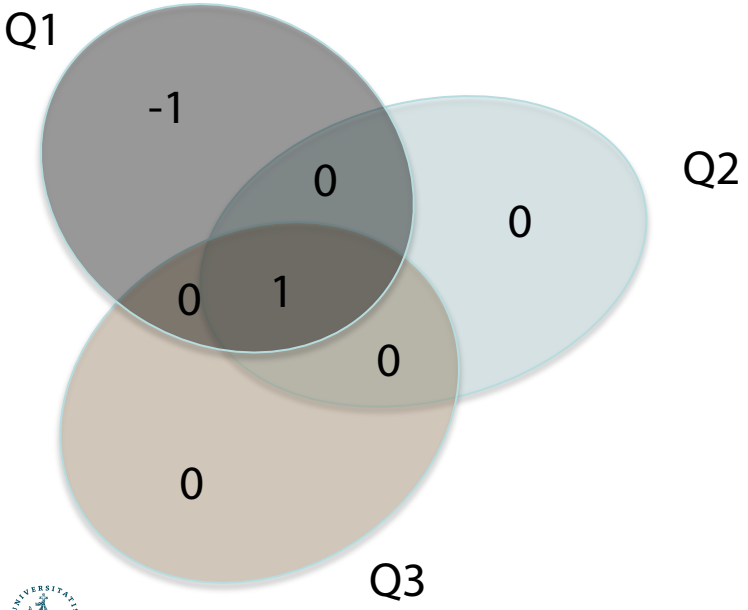


NB: Dieses ist dual zur häufiger verwendeten Formel:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

Regel 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$

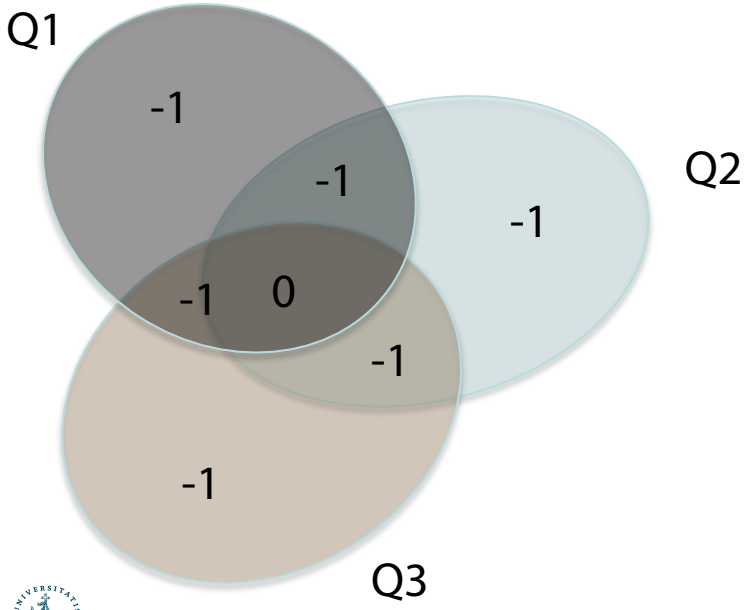


NB: Dieses ist dual zur häufiger verwendeten Formel:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

Regel 4: Inclusion-Exclusion

$$P(Q1 \wedge Q2 \wedge Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \vee Q2) - P(Q1 \vee Q3) - P(Q2 \vee Q3) + P(Q1 \vee Q2 \vee Q3)$$



NB: Dieses ist dual zur häufiger verwendeten Formel:

$$P(Q1 \vee Q2 \vee Q3) = P(Q1) + P(Q2) + P(Q3) - P(Q1 \wedge Q2) - P(Q1 \wedge Q3) - P(Q2 \wedge Q3) + P(Q1 \wedge Q2 \wedge Q3)$$

Beispiel

$Q_J = R(x_1), S(x_1,y_1), T(x_2), S(x_2,y_2)$

$= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \wedge [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

Beispiel

$Q_J = R(x_1), S(x_1,y_1), T(x_2), S(x_2,y_2)$ $= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \wedge [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

$Q_J = Q_1 \wedge Q_2$ wobei $Q_1 = R(x_1), S(x_1,y_1)$ $Q_2 = T(x_2), S(x_2,y_2)$

Beispiel

$Q_J = R(x_1), S(x_1,y_1), T(x_2), S(x_2,y_2)$ $= [\exists x_1 \exists y_1 R(x_1) \wedge S(x_1,y_1)] \wedge [\exists x_2 \exists y_2 T(x_2) \wedge S(x_2,y_2)]$

$Q_J = Q_1 \wedge Q_2$ wobei $Q_1 = R(x_1), S(x_1,y_1)$ $Q_2 = T(x_2), S(x_2,y_2)$

$$P(Q_J) = P(Q_1) + P(Q_2) - P(Q_1 \vee Q_2)$$

Q_1 = eine hierarchische CQ ohne Self-Joins

Q_2 = dito

$Q_1 \vee Q_2 = Q_U$, siehe vorige Folien

Sichere Anfragen formal

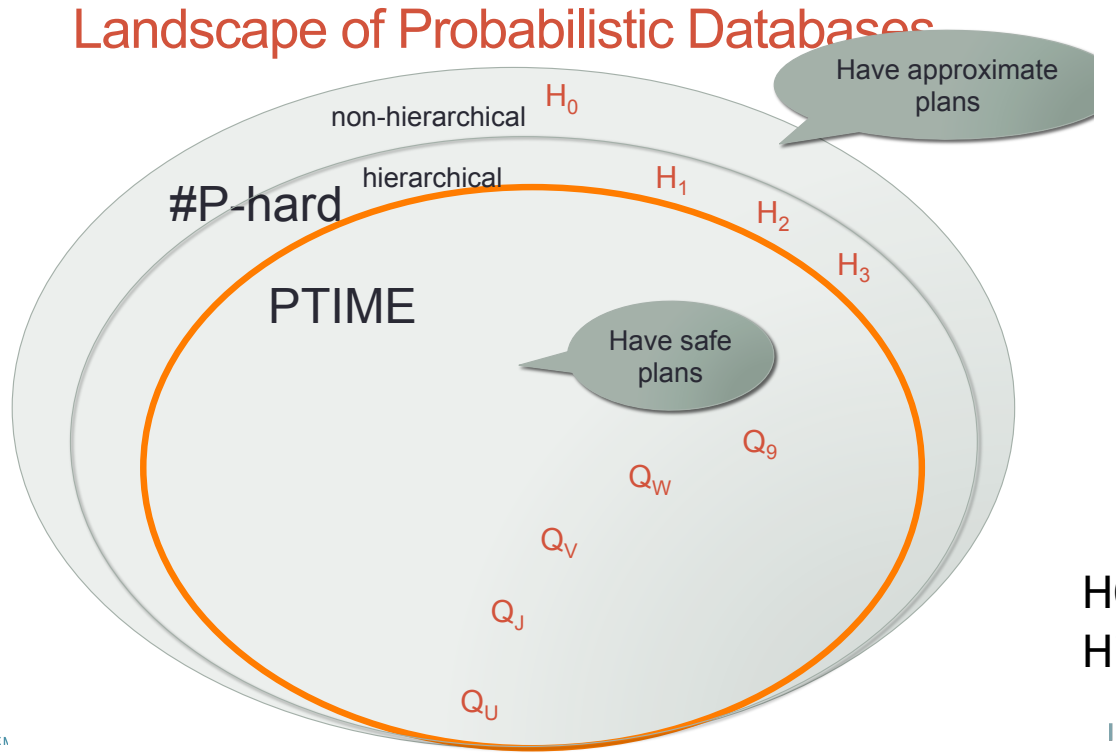
Definition Anfrage Q ist bzgl. Regelmenge R sicher gdw
man die Regeln in R erfolgreich anwenden kann, d.h.,
in jeder Verzweigung nach endlich vielen Schritten nur noch für atomare Anfragen (Abfrage in Tabelle) Wahrscheinlichkeit P zu berechnen ist

- Insbesondere: Termination erforderlich
- Die Regeln lassen sich 1-zu-1 in algebraische Transformationsregeln übersetzen (extensionale Pläne)
  - Konjunktionsregel -> Regel für independent join etc.
- D.h.: Falls erfolgreiche Anwendung der Regeln, dann sicherer Plan konstruierbar
- Aber: Müssen Inclusion-/Exclusion generalisieren (Möbius-Formel) und 2 weitere hinzunehmen

Das große Dichotomietheorem

Theorem [Dalvi &Suciu, JACM'2012] Für UCQs Q:

- Falls die 4 + 2 Regeln erfolgreich anwendbar auf Q, dann ist Q sicher und man kann man P(Q) in PTIME berechnen
- Falls die 4 + 2 Regeln nicht erfolgreich anwendbar auf Q, dann ist Q nicht sicher und das Problem P(Q) zu berechnen ist #P-vollständig.



H0 ist eine CQ
Hi (i > 0) sind UCQs

Einsicht 3

Vereinigung (union) ist nötig, um Self-Joins zu behandeln!

(Inclusion/Exclusion-Regel führt zu Union)

Daher:

- Conjunctive Queries = Keine “natürliche” Klasse von Anfragen an probabilistische DBs
- Unions of Conjunctive Queries = die “natürliche” Klasse von Anfragen

Zusammenfassung

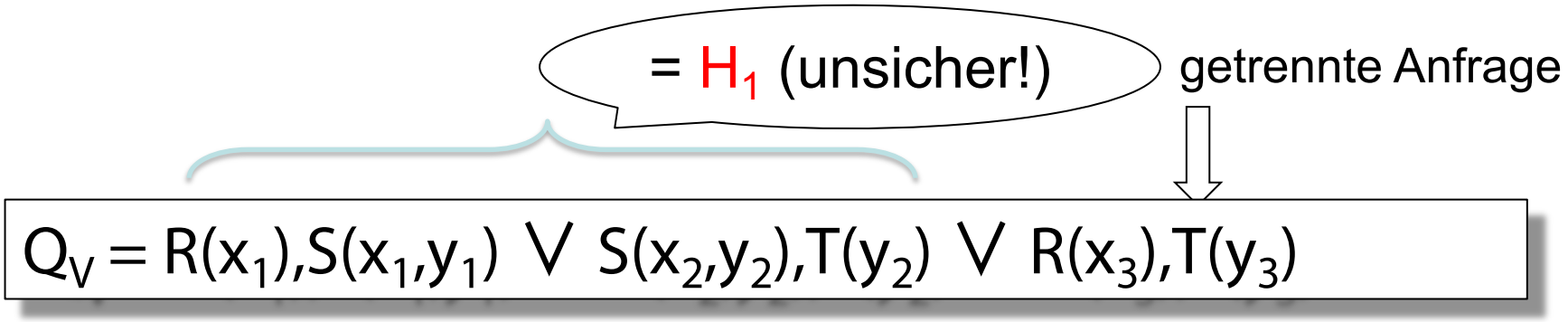
- Falls Anfrageformel mit Regeln transformiert werden kann, dann ist das Ergebnis sicher (→ Lifted Inference)
- Für Vollständigkeit (alle sicheren Pläne inkl. untere Grenzen werden erzeugt) muss Inclusion/Exclusion durch Möbius-Regel ersetzt werden und zwei weitere Regel aufgenommen werden
- Aber: Nicht alle Anfragen lassen sich mit den Regeln behandeln (→ nicht immer gibt es einen sicheren Plan)
  - Für bestimmte Anfragen müssen potentiell die Welten betrachtet werden (→ Grounded Inference)

L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and Simple Relational Processing of Uncertain Data, Proc. 24th International Conference on Data Engineering, ICDE 2008, 983-992, **2008**

Sichere Anfrage mit unsicherer Unteranfrage H_1

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

Sichere Anfrage mit unsicherer Unteranfrage H_1



Sichere Anfrage mit unsicherer Unteranfrage H_1

= H_1 (unsicher!)

getrennte Anfrage

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

(Richtung) KNF

$$Q_V = [S(x_2,y_2),T(y_2) \vee R(x_3)] \wedge [R(x_1),S(x_1,y_1) \vee T(y_3)]$$

Sichere Anfrage mit unsicherer Unteranfrage H_1

= H_1 (unsicher!)

getrennte Anfrage

DNF

$$Q_V = R(x_1),S(x_1,y_1) \vee S(x_2,y_2),T(y_2) \vee R(x_3),T(y_3)$$

(Richtung) KNF

$$Q_V = [S(x_2,y_2),T(y_2) \vee R(x_3)] \wedge [R(x_1),S(x_1,y_1) \vee T(y_3)]$$

Inclusion/exclusion:

PTIME !

$$P(Q_V) = P(q_1 \wedge q_2) = P(q_1) + P(q_2) - P(q_1 \vee q_2)$$

$$= R(x_3) \vee T(y_3)$$

Inclusion/Exclusion ist nicht ausreichend

$$
\begin{aligned}
Q_W = [&R(x_0), S_1(x_0,y_0) \quad \vee \quad S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \quad /*\ Q1\ */ \\
&[R(x_0), S_1(x_0,y_0) \quad \vee \quad S_3(x_3,y_3), T(y_3)] \quad \wedge \quad /*\ Q2\ */ \\
&[S_1(x_1,y_1), S_2(x_1,y_1) \quad \vee \quad S_3(x_3,y_3), T(y_3)] \quad /*\ Q3\ */
\end{aligned}
$$

Inclusion/Exclusion ist nicht ausreichend

$$
\begin{array}{llllll}
Q_W = & [R(x_0), S_1(x_0,y_0) & \vee & S_2(x_2,y_2), S_3(x_2,y_2)] & \wedge & /* \ Q1 \ */ \\
 & [R(x_0), S_1(x_0,y_0) & \vee & S_3(x_3,y_3), T(y_3)] & \wedge & /* \ Q2 \ */ \\
 & [S_1(x_1,y_1), S_2(x_1,y_1) & \vee & S_3(x_3,y_3), T(y_3)] & & /* \ Q3 \ */
\end{array}
$$

$$
\begin{aligned}
P(Q_W) = \; & P(Q_1) + P(Q_2) + P(Q_3) + \\
& - P(Q_1 \vee Q_2) - P(Q_2 \vee Q_3) - P(Q_1 \vee Q_3) \\
& + P(Q_1 \vee Q_2 \vee Q_3)
\end{aligned}
$$

Auch = H_3

= H_3 (hard !)

Inclusion/Exclusion ist nicht ausreichend

$$Q_W = [R(x_0), S_1(x_0,y_0) \vee S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \text{/* Q1 */}$$
$$[R(x_0), S_1(x_0,y_0) \vee S_3(x_3,y_3), T(y_3)] \wedge \text{/* Q2 */}$$
$$[S_1(x_1,y_1), S_2(x_1,y_1) \vee S_3(x_3,y_3), T(y_3)] \text{ /* Q3 */}$$

PTIME

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) +$$
$$- P(Q_1 \vee Q_2) - P(Q_2 \vee Q_3) - P(Q_1 \vee Q_3)$$
$$+ P(Q_1 \vee Q_2 \vee Q_3)$$

= H_3 (hard !)

Auch = H_3

#P-hart

Inclusion/Exclusion ist nicht ausreichend

$$Q_W = [R(x_0), S_1(x_0,y_0) \vee S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \quad /* Q1 */$$
$$[R(x_0), S_1(x_0,y_0) \vee S_3(x_3,y_3), T(y_3)] \wedge \quad /* Q2 */$$
$$[S_1(x_1,y_1), S_2(x_1,y_1) \vee S_3(x_3,y_3), T(y_3)] \quad /* Q3 */$$

PTIME

$$P(Q_W) = P(Q_1) + P(Q_2) + P(Q_3) + \\ - P(Q_1 \vee Q_2) - P(Q_2 \vee Q_3) - P(Q_1 \vee Q_3) \\ + P(Q_1 \vee Q_2 \vee Q_3)$$

PTIME

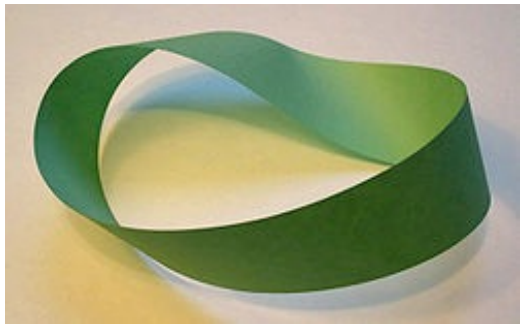
= H_3 (hard !)

Auch = H_3

#P-hart

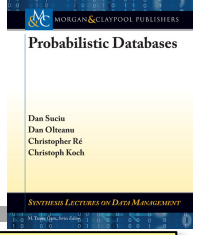
August Ferdinand Möbius 1790-1868

- Möbiusband
- Möbiusfunktion μ in Zahlentheorie
- Verallgemeinert auf Verbände [Stanley'97,Rota'09]
- Lässt sich auch auf Anfragen anwenden!



Der KNF-Verband

Formale Definition im Buch



Definition. KNF-Verband von Q = Q1 ∧ Q2 ∧ …:

Der KNF-Verband

Definition. KNF-Verband von Q = Q1 ∧ Q2 ∧ …:

Beispiel

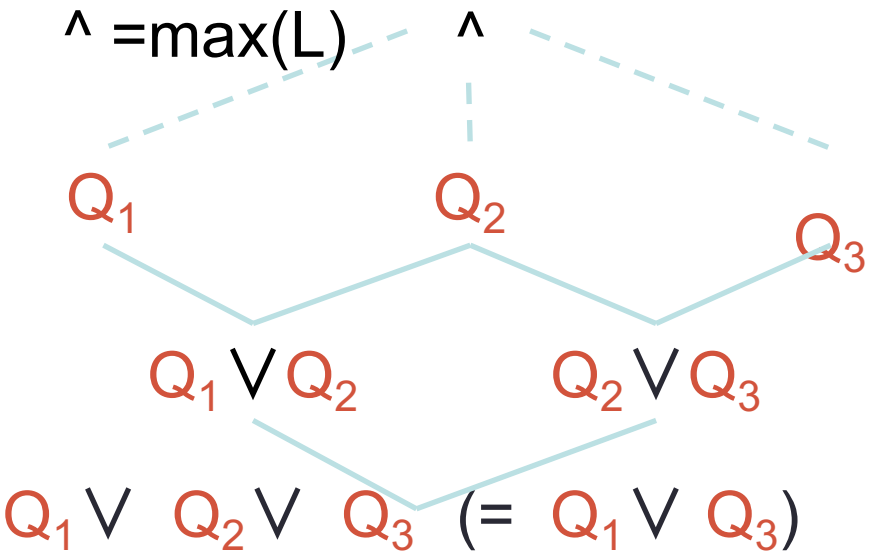
$$
\begin{aligned}
Q_W = [&R(x_0), S_1(x_0,y_0) \quad \vee \quad S_2(x_2,y_2), S_3(x_2,y_2)] \wedge \quad /* \text{ Q1 } */ \\
&[R(x_0), S_1(x_0,y_0) \quad \vee \quad S_3(x_3,y_3), T(y_3)] \quad \wedge \quad /* \text{ Q2 } */ \\
&[S_1(x_1,y_1), S_2(x_1,y_1) \quad \vee \quad S_3(x_3,y_3), T(y_3)] \quad /* \text{ Q3 } */
\end{aligned}
$$

Der KNF-Verband

Definition. KNF-Verband von Q = Q1 ∧ Q2 ∧ …:

Beispiel

$$
\begin{array}{llll}
Q_W = [R(x_0), S_1(x_0,y_0) & \vee & S_2(x_2,y_2), S_3(x_2,y_2)] \wedge & /* \text{ Q1 } */ \\
\quad [R(x_0), S_1(x_0,y_0) & \vee & S_3(x_3,y_3), T(y_3)] \wedge & /* \text{ Q2 } */ \\
\quad [S_1(x_1,y_1), S_2(x_1,y_1) & \vee & S_3(x_3,y_3), T(y_3)] & /* \text{ Q3 } */
\end{array}
$$

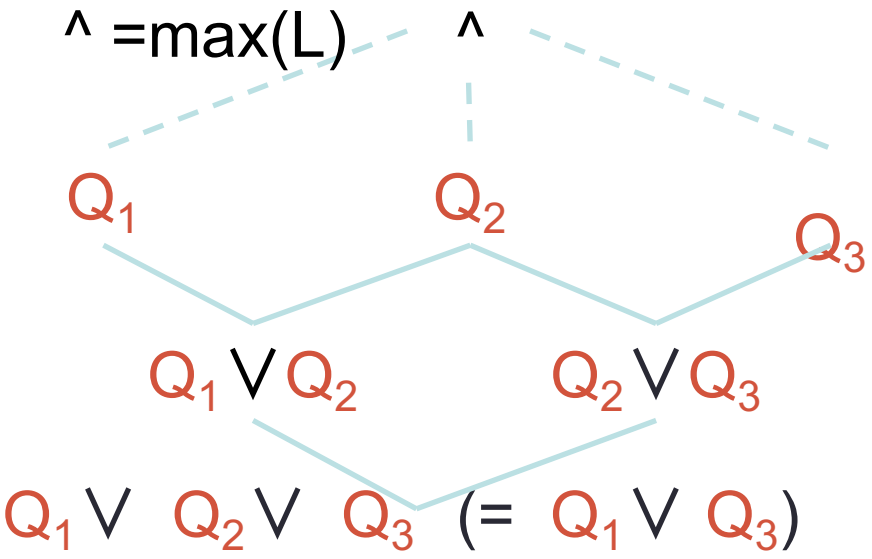


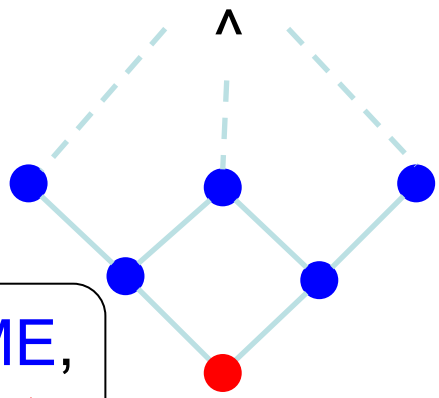
Der KNF-Verband

Definition. KNF-Verband von Q = Q1 ∧ Q2 ∧ …:

Beispiel

$$
\begin{array}{llll}
Q_W = [R(x_0), S_1(x_0,y_0) & \vee & S_2(x_2,y_2), S_3(x_2,y_2)] \wedge & /* \text{ Q1 } */ \\
\quad [R(x_0), S_1(x_0,y_0) & \vee & S_3(x_3,y_3), T(y_3)] \wedge & /* \text{ Q2 } */ \\
\quad [S_1(x_1,y_1), S_2(x_1,y_1) & \vee & S_3(x_3,y_3), T(y_3)] & /* \text{ Q3 } */
\end{array}
$$





Knoten• in PTIME,
Knoten • #P-hart.

Die Möbius-Funktion

<, ≤ für
Verbandsordnung

Def. Die Möbius-Funktion:

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

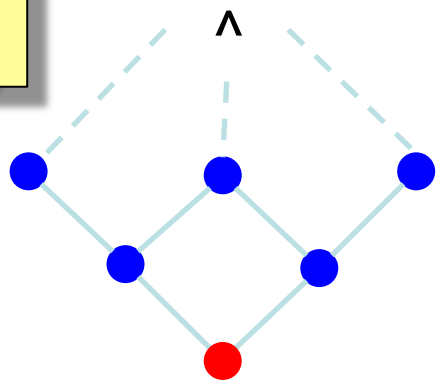
$\mu(u, \hat{}) = - \Sigma_{u < v \leq \hat{}} \mu(v, \hat{})$

Möbius-Inversionsformel:

$$P(Q) = - \Sigma_{Qi < \hat{}} \mu(Qi, \hat{}) P(Qi)$$

Qi sind nun die Elemente im Verband

Unsichere Teilanfragenanfragen Qu werden eventuell aufgrund von $\mu(Qu, \hat{}) = 0$ rausgeschmissen



Die Möbius-Funktion

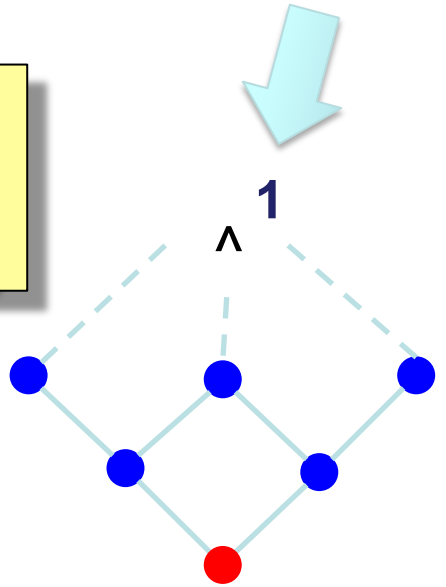
Def. Die Möbius-Funktion:

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

$\mu(u, \hat{}) = - \Sigma_{u < v \leq \hat{}} \mu(v, \hat{})$

Möbius-Inversionsformel:

$$P(Q) = - \Sigma_{Qi < \hat{}} \mu(Qi, \hat{}) P(Qi)$$



Die Möbius-Funktion

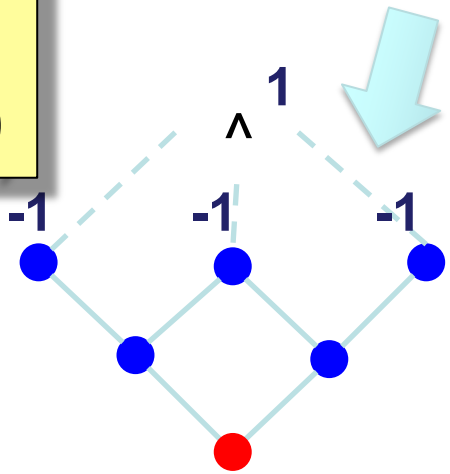
Def. Die Möbius-Funktion:

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

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$

Möbius-Inversionsformel:

$$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) \, P(Qi)$$



Die Möbius-Funktion

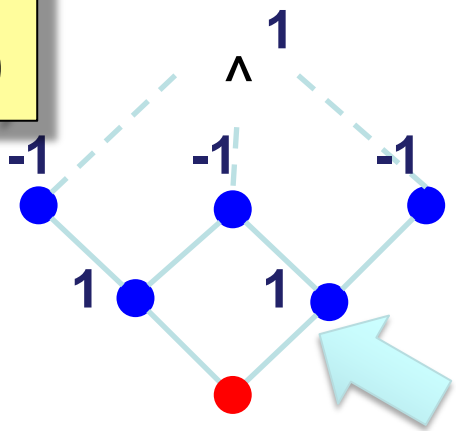
Def. Die Möbius-Funktion:

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

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$

Möbius-Inversionsformel:

$$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) \, P(Qi)$$



Die Möbius-Funktion

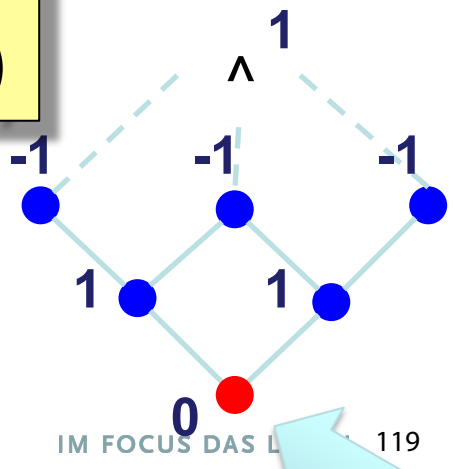
Def. Die Möbius-Funktion:

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

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$

Möbius-Inversionsformel:

$$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) \ P(Qi)$$

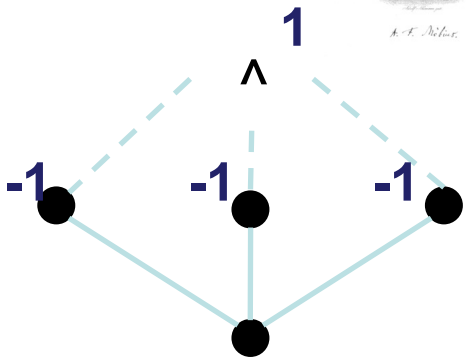


Die Möbius-Funktion

Def. Die Möbius-Funktion:

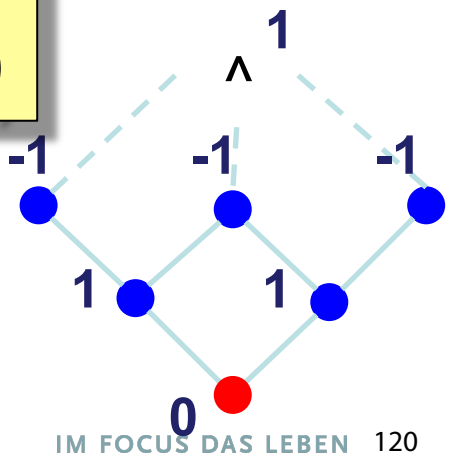
$\mu(\hat{} \ , \hat{} \ ) = 1$

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$



Möbius-Inversionsformel:

$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) P(Qi)$

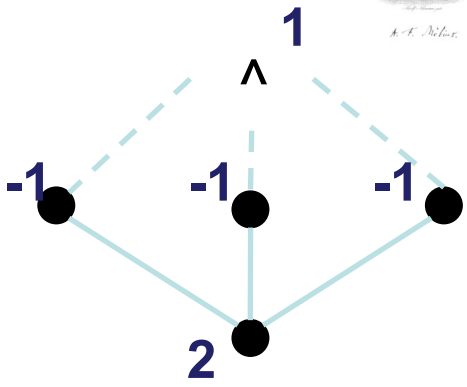


Die Möbius-Funktion

Def. Die Möbius-Funktion:

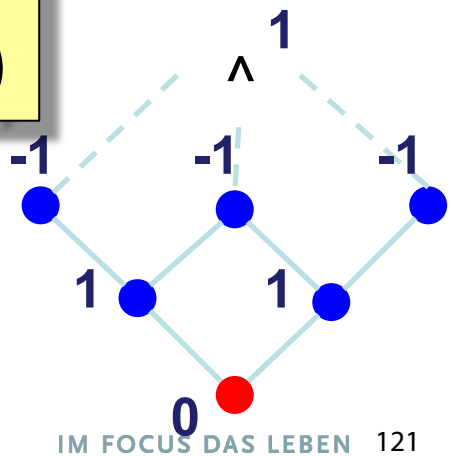
$\mu(\hat{} \ , \hat{} \ ) = 1$

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$



Möbius-Inversionsformel:

$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) P(Qi)$

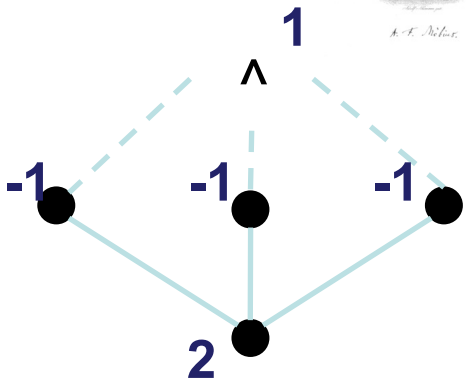


Die Möbius-Funktion

Def. Die Möbius-Funktion:

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

$\mu(u, \hat{} \ ) = - \Sigma_{u < v \leq \hat{}} \ \mu(v, \hat{} \ )$



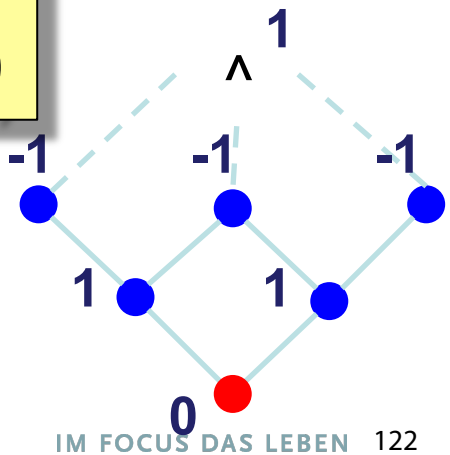
Möbius-Inversionsformel:

$P(Q) = - \Sigma_{Qi < \hat{}} \ \mu(Qi, \hat{} \ ) P(Qi)$

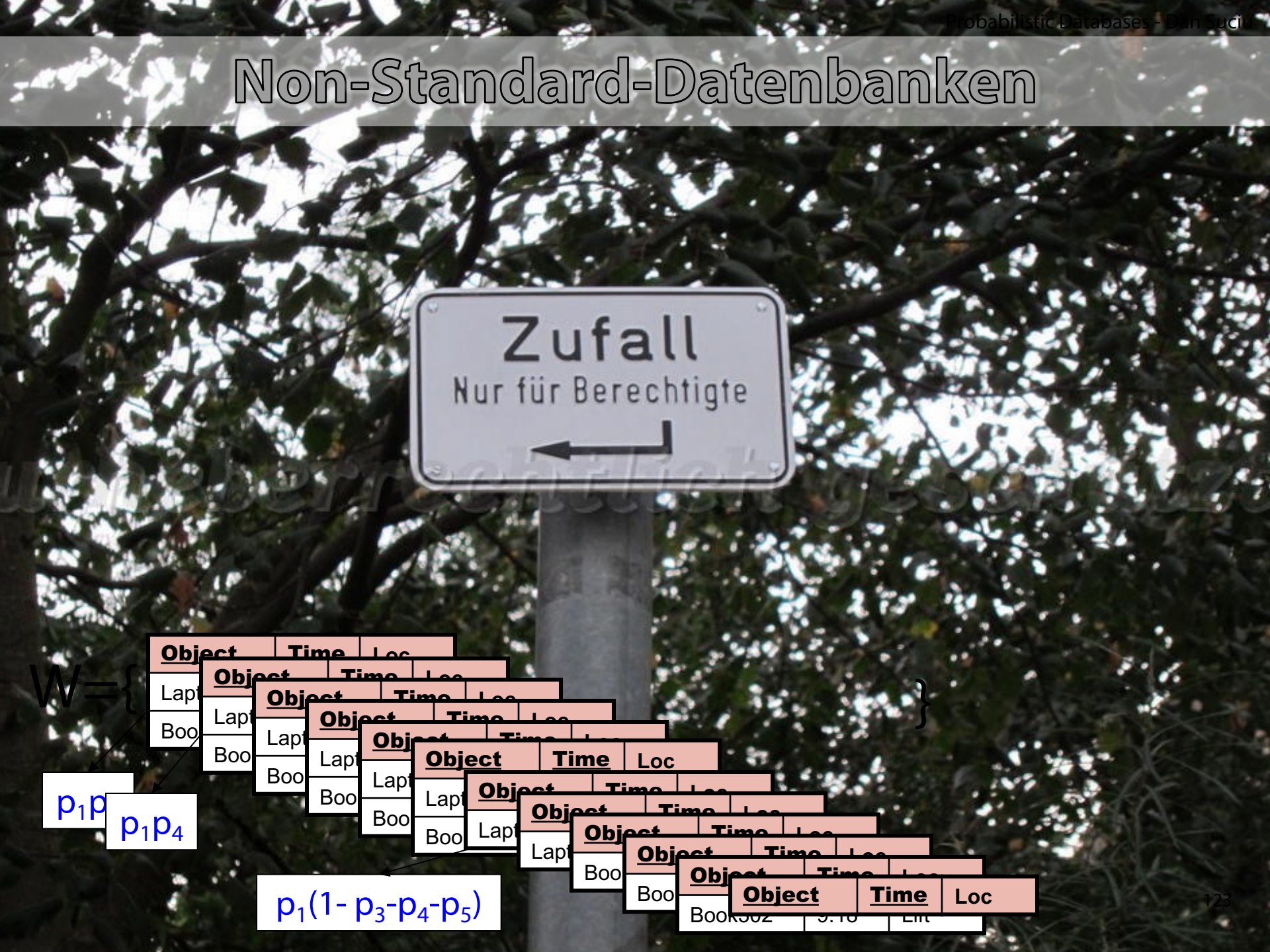
Neue Regel

Inclusion/Exclusion

→ Möbius-Inversionsformel

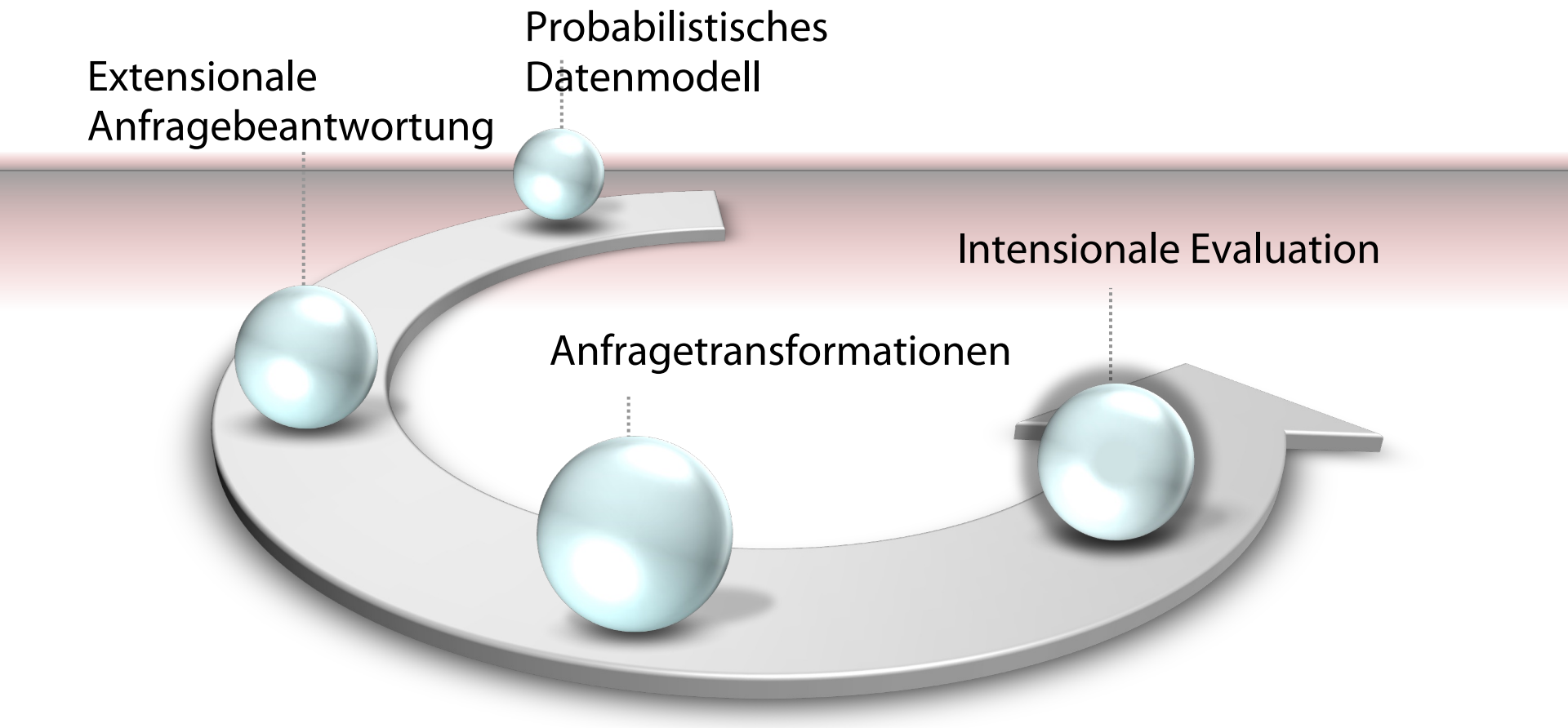


Non-Standard-Datenbanken



Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Zusammenfassung

- Falls Anfrageformel mit Regeln transformiert werden kann, dann ist das Ergebnis sicher (→ Lifted Inference)
- Aber: Nicht alle Anfragen lassen sich mit den Regeln behandeln (→ nicht immer gibt es einen sicheren Plan)
  - Für bestimmte Anfragen müssen potentiell die Welten betrachtet werden (→ Grounded Inference)

L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and Simple Relational Processing of Uncertain Data, Proc. 24th International Conference on Data Engineering, ICDE 2008, 983-992, **2008**

04.2013
LOST?

Anfragebeantwortung mit Herkunftsformel

- Intensionale Anfrage-beantwortung [FR97]
- Assoziiere mit jedem Tupel ein Ereignis
- Wahrscheinlichkeit, dass Ereignis eintritt / erfüllt ist = Anfragewert

Object	Time	Loc	P	
Laptop77	9:07	Rm444	p_1	e_1
Laptop77	9:07	Hall	p_2	e_2
Book302	9:18	Office	p_3	e_3
Book302	9:18	Rm444	p_4	e_4
Book302	9:18	Lift	p_5	e_5

Anfrageverarbeitung generiert Ereignisausdruck

Join	$\times((t_1,[e_1]),(t_2,[e_2]))$	$=((t_1,t_2),[e_1 \wedge e_2])$
Selektion	$\sigma_c((t,[e]))$	$= \begin{cases} (t,[e]) & c(t) = \text{true} \\ \bot & c(t) = \text{false} \end{cases}$
Projektion	$\Pi_A((t_1,[e_1]),\ldots,(t_n,[e_n]))$	$=(t_1[A],[e_1 \vee \cdots \vee e_n])$

Projektion sorgt für DNF

N. Fuhr; T. Rölleke, A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. ACM Transactions on Information Systems 14(1), **1997**.

Neuere Forschung

- Anwendung von Techniken des Weighted Model Counting zur Lösung von #SAT-Problemen
- Umwandlung von Wahrscheinlichkeiten in Gewichten

- Folgende Präsentation sind angelehnt an ein Tutorial:

Lifted Probabilistic Inference in Relational Models

Guy Van den Broeck KU Leuven
Dan Suciu U. of Washington

SAT und #SAT

Gegeben eine KNF-Formel

- SAT: Finde erfüllende Belegung
  - NP-vollständig: SAT, graph coloring, Hamiltonian cycle, …
- #SAT: Zähle erfüllende Belegungen
  - #P-vollständig: #SAT, …
- Beispiel: $F = (x \vee y) \wedge (y \vee \neg z)$
  - 5 erfüllende Belegungen
  - (0,1,0), (0,1,1), (1,1,0), (1,1,1), (1, 0, 0)

Wiederholung: DPLL(KNF-Formel)

DPLL(F)

if F is empty **then** return 1

if F contains an empty clause **then** return 0

else choose a variable x to branch

return $DPLL(F|_{x=1}) \vee DPLL(F|_{x=0})$

#DPLL(F) // computes expected satisfying probability of F

if F is empty then **return** 1

if F contains an empty clause **then** return 0

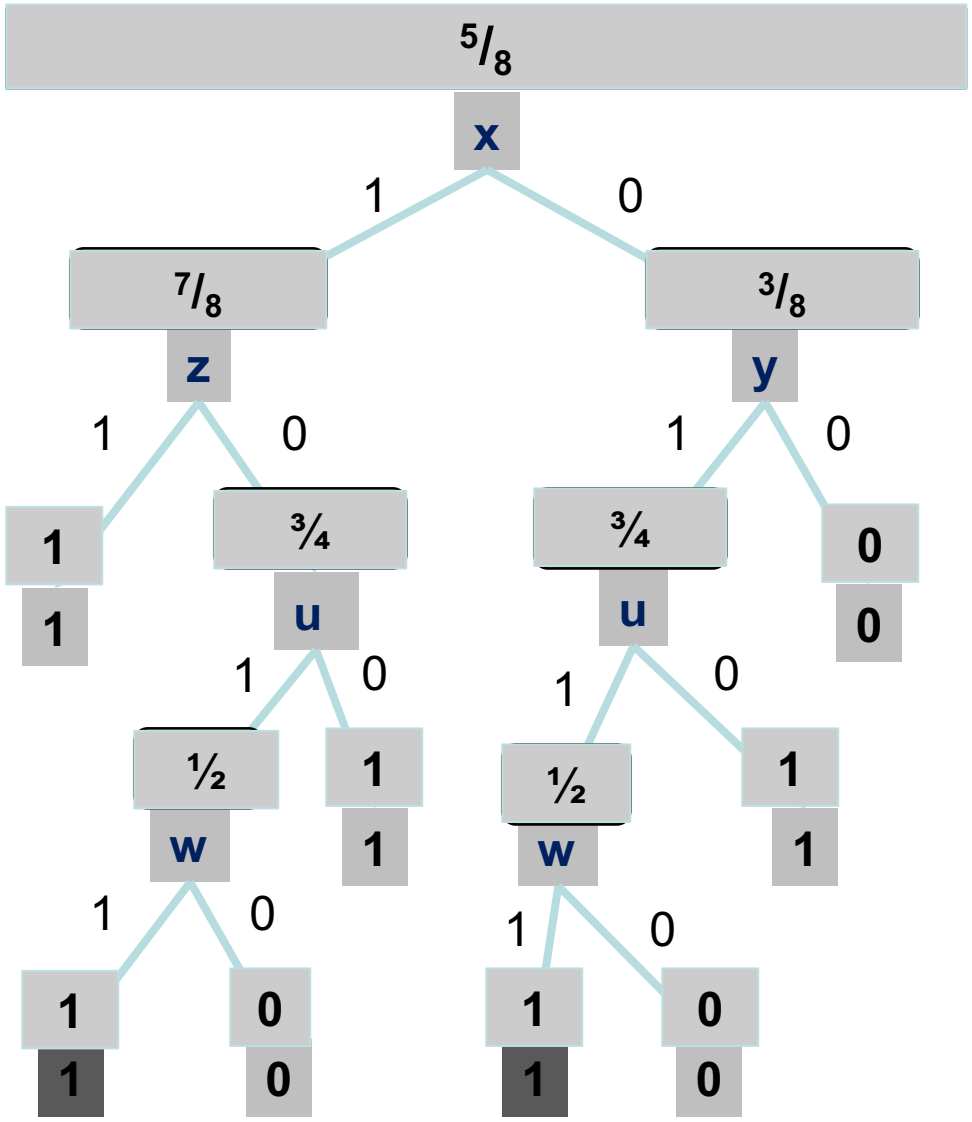
else choose a variable x to branch

return $0.5 * \#DPLL(F|_{x=1}) + 0.5 * \#DPLL(F|_{x=0})$

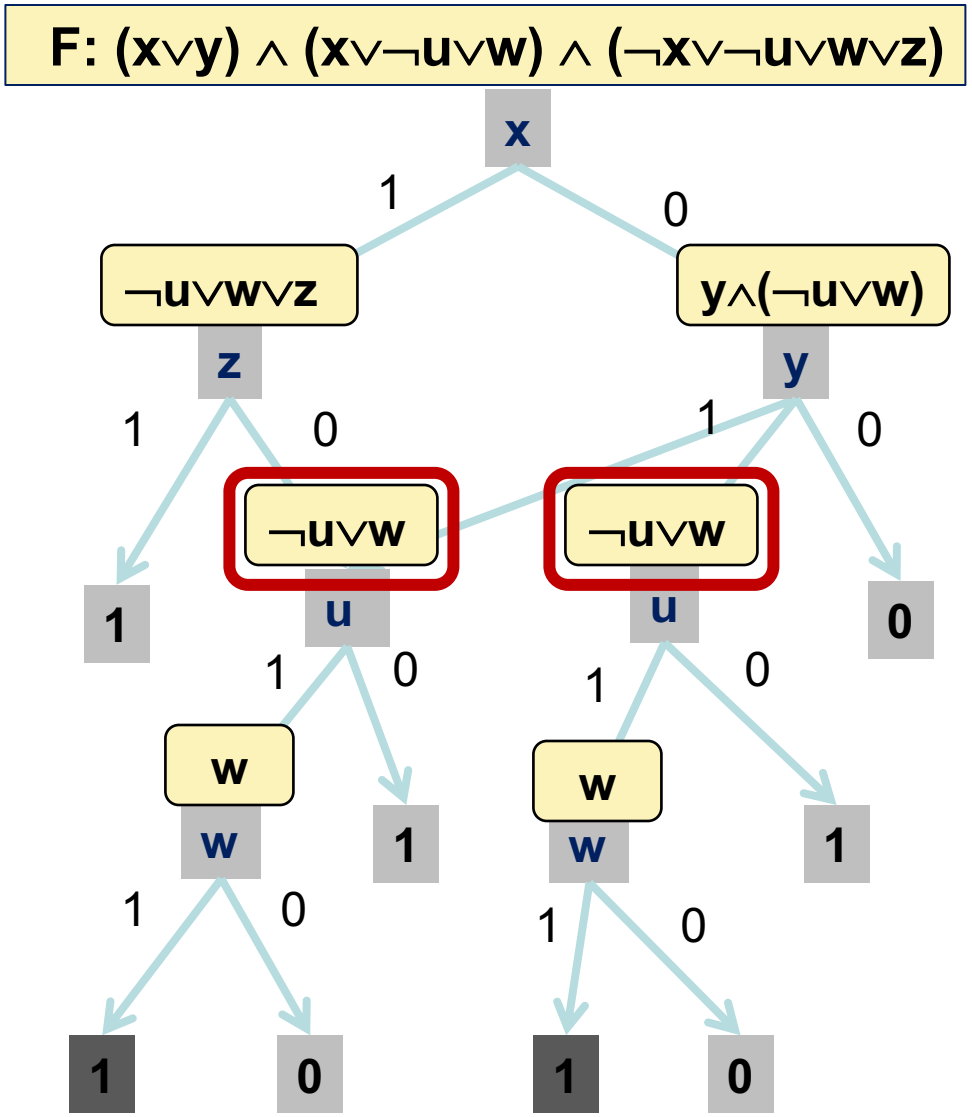
Davis, Martin; Putnam, Hilary. "A Computing Procedure for Quantification Theory". Journal of the ACM. 7 (3): 201–215. **1960**.

Davis, Martin; Logemann, George; Loveland, Donald. "A Machine Program for Theorem Proving". Communications of the ACM. 5 (7): 394–397. **1962**.

DPLL: Beispiel



Caching



Weighted Model Counting (WMC)

- Jedes Literal hat Gewicht $\in$ [0,Max]
  - Gewicht eines Modells = Produkt der Literalgewichte
  - Gewicht einer Formel = Summe der Gewichte der Modelle

Basic Weighted Model Counting (BWMC)

BWMC(F)

if F is empty **then return** 1

if F contains an empty clause **then return** 0

else choose a variable x to branch

return (BWMC($F|_{x=1}$) * weight (x) +

BWMC($F|_{x=0}$) * weight ($\neg$x))

Weighted Model Counting

- Modell = Erfüllende Belegung einer aussagenlogischen Formel Δ
- Zählen der Modelle: #SAT
- Weighted Model Counting (WMC)
  - Gewichte für Variablenzuweisung
  - Modellgewicht = Produkt der Variablengewichte w(·)

$\Delta = (\text{Rain} \Rightarrow \text{Cloudy})$

Rain

w(R)	w(¬R)
1	2

Cloudy

w(C)	w(¬C)
3	5

Rain	Cloudy	Model?	Weight
T	T	Yes	1 * 3 = 3
T	F	No	0
F	T	Yes	2 * 3 = 6
F	F	Yes	2 * 5 = 10
		+	+
		#SAT = 3	**WMC = 19**

Weighted First-Order Model Counting

Modell = Erfüllende Belegung einer aussagenlogischen Formel Δ

$\Delta = \forall d\ (\text{Rain}(d) \Rightarrow \text{Cloudy}(d))$

Days = {Monday **Tuesday**}

Rain

d	w(R(d))	w(¬R(d))
M	1	2
T	4	1

Cloudy

d	w(C(d))	w(¬C(d))
M	3	5
T	6	2

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?	Weight
T	T	T	T	Yes	1 * 3 * 4 * 6 = 72
T	F	T	T	No	0
F	T	T	T	Yes	2 * 3 * 4 * 6 = 144
F	F	T	T	Yes	2 * 5 * 4 * 6 = 240
T	T	T	F	No	0
T	F	T	F	No	0
F	T	T	F	No	0
F	F	T	F	No	0
T	T	F	T	Yes	1 * 3 * 1 * 6 = 18
T	F	F	T	No	0
F	T	F	T	Yes	2 * 3 * 1 * 6 = 36
F	F	F	T	Yes	2 * 5 * 1 * 6 = 60
T	T	F	F	Yes	1 * 3 * 1 * 2 = 6
T	F	F	F	No	0
F	T	F	F	Yes	2 * 3 * 1 * 2 = 12
F	F	F	F	Yes	2 * 5 * 1 * 2 = 20
				+	+
				#SAT = 9	WFOMC = 608

Gogate, V., & Domingos, P. Probabilistic theorem proving. Proc. UAI, **2012**.

Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., & De Raedt, L., Lifted probabilistic inference by first-order knowledge compilation. In Proc.IJCAI-11, pp. 2178-2185, **2011**.

Lifting

Thursday, Wednesday, … ?

Model = solution to first-order logic formula Δ

$\Delta = \forall d\ (\text{Rain}(d) \Rightarrow \text{Cloudy}(d))$

Days = {Monday **Tuesday**}

Rain

d	w(R(d))	w(¬R(d))
M	1	2
T	4	1

Cloudy

d	w(C(d))	w(¬C(d))
M	3	5
T	6	2

Rain(M)	Cloudy(M)	Rain(T)	Cloudy(T)	Model?	Weight
T	T	T	T	Yes	1 * 3 * 4 * 6 = 72
T	F	T	T	No	0
F	T	T	T	Yes	2 * 3 * 4 * 6 = 144
F	F	T	T	Yes	2 * 5 * 4 * 6 = 240
T	T	T	F	No	0
T	F	T	F	No	0
F	T	T	F	No	0
F	F	T	F	No	0
T	T	F	T	Yes	1 * 3 * 1 * 6 = 18
T	F	F	T	No	0
F	T	F	T	Yes	2 * 3 * 1 * 6 = 36
F	F	F	T	Yes	2 * 5 * 1 * 6 = 60
T	T	F	F	Yes	1 * 3 * 1 * 2 = 6
T	F	F	F	No	0
F	T	F	F	Yes	2 * 3 * 1 * 2 = 12
F	F	F	F	Yes	2 * 5 * 1 * 2 = 20
				+	+
				#SAT = 9	WFOMC = 608

[V.d.Broeck'11a, Gogate'11]

Von Wahrscheinlichkeiten zu Gewichten

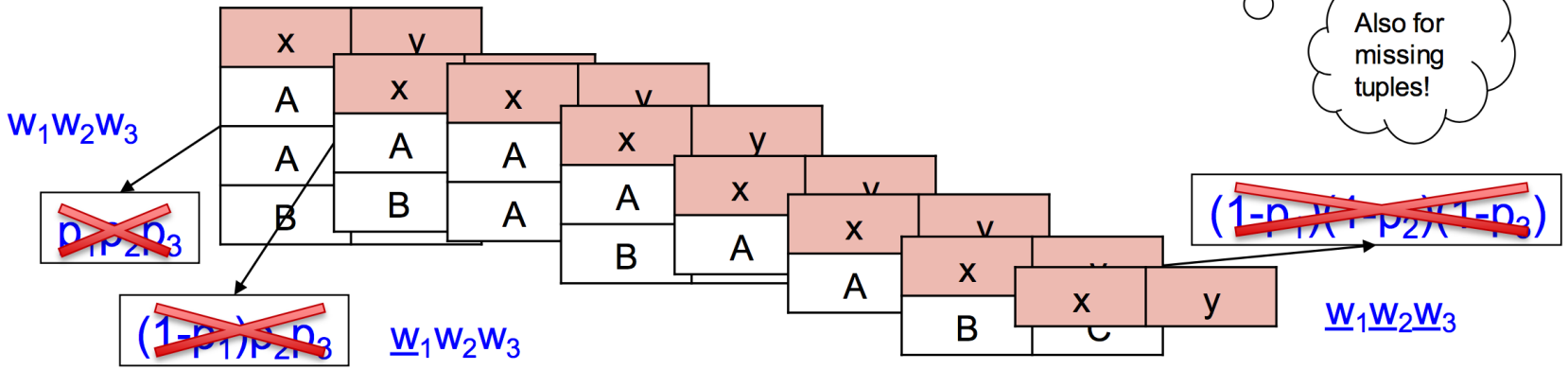
Rechner-arithmetik!

Friend

x	y	P
A	B	~~p_1~~
A	C	~~p_2~~
B	C	~~p_3~~

→

x	y	w(Friend(x,y))	w(¬Friend(x,y))
A	B	$w_1 = p_1$	$\underline{w_1} = 1-p_1$
A	C	$w_2 = p_2$	$\underline{w_2} = 1-p_2$
B	C	$w_3 = p_3$	$\underline{w_3} = 1-p_3$
A	A	$w_4 = 0$	$\underline{w_4} = 1$
A	C	$w_5 = 0$	$\underline{w_5} = 1$
	…	…	



Intensionale Anfrageevaluation

- Anfrage Q + ProbDB D
  => Herkunftsformel (*lineage expression)* F
  - Boolesche Variablen w_1, $\underline{w}_1$, ...
    korrespondierend zu Tupeln t1, t2, ...
  - Die Herkunftsformel F sagt, wann Q wahr ist
- Berechne P(F) mit DPLL-artigem System
- Nutze ggf. WFOMC und Lifting

P. Agrawal, O. Benjelloun, A. Das Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom. Trio: A System for Data, Uncertainty, and Lineage. Proc. 32nd Intl. Conference on Very Large Data Bases, pages 1151-1154, **2006**

J. Widom. Trio: A System for Data, Uncertainty, and Lineage. In C. Aggarwal, editor, Managing and Mining Uncertain Data, Springer, **2009**.

Diskussion

- Einfache Idee: Ersetze p, 1-p durch w, w
- Anfragebeantwortung durch WFOMC
- Für Wahrscheinlichkeitsraum:
  - Dividiere Weltgewicht durch Z = Summe aller Weltgewichte

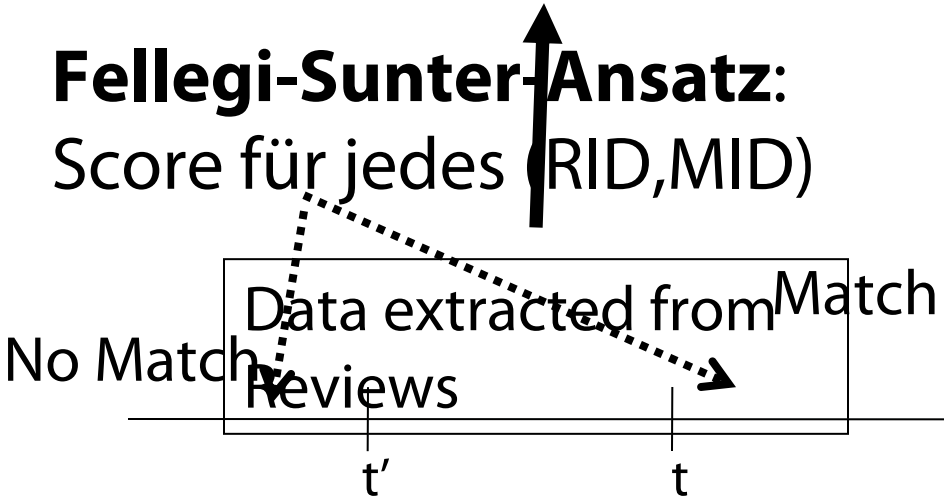
- BUD?
- Verschiedene Formalismen zur Darstellung von Wahrscheinlichkeitsverteilungen

Beispiel

RID	Title
r124	12 Monkeys
r155	Twelve Monkeys
r175	2 Monkey
r194	Monk

MID	Title
m232	12 Monkeys
m143	Monkey Love

Fellegi-Sunter-Ansatz:
Score für jedes (RID,MID)



Clean IMDB Data

New Approach:
Output: (MID,MID) pairs
Convert scores to probabilities

Fellegi, Ivan; Sunter, Alan. "A Theory for Record Linkage" Journal of the American Statistical Association. 64 (328): pp. 1183–1210, **1969**.

Unschärfe ist überall…

RID	Title
r124	12 Monkeys
r155	Twelve Monkeys
r175	**2 Monkey**
r194	Monk

MID	Title
m232	**12 Monkeys**
m143	**Monkey Love**

Fellegi-Sunter-Ansatz:
Score für jedes (RID,MID)

No Match Match

RID	MID	Prob
r175	m232	0.8
r175	m143	0.2

Fellegi, Ivan; Sunter, Alan. "A Theory for Record Linkage" Journal of the American Statistical Association. 64 (328): pp. 1183–1210, **1969**.

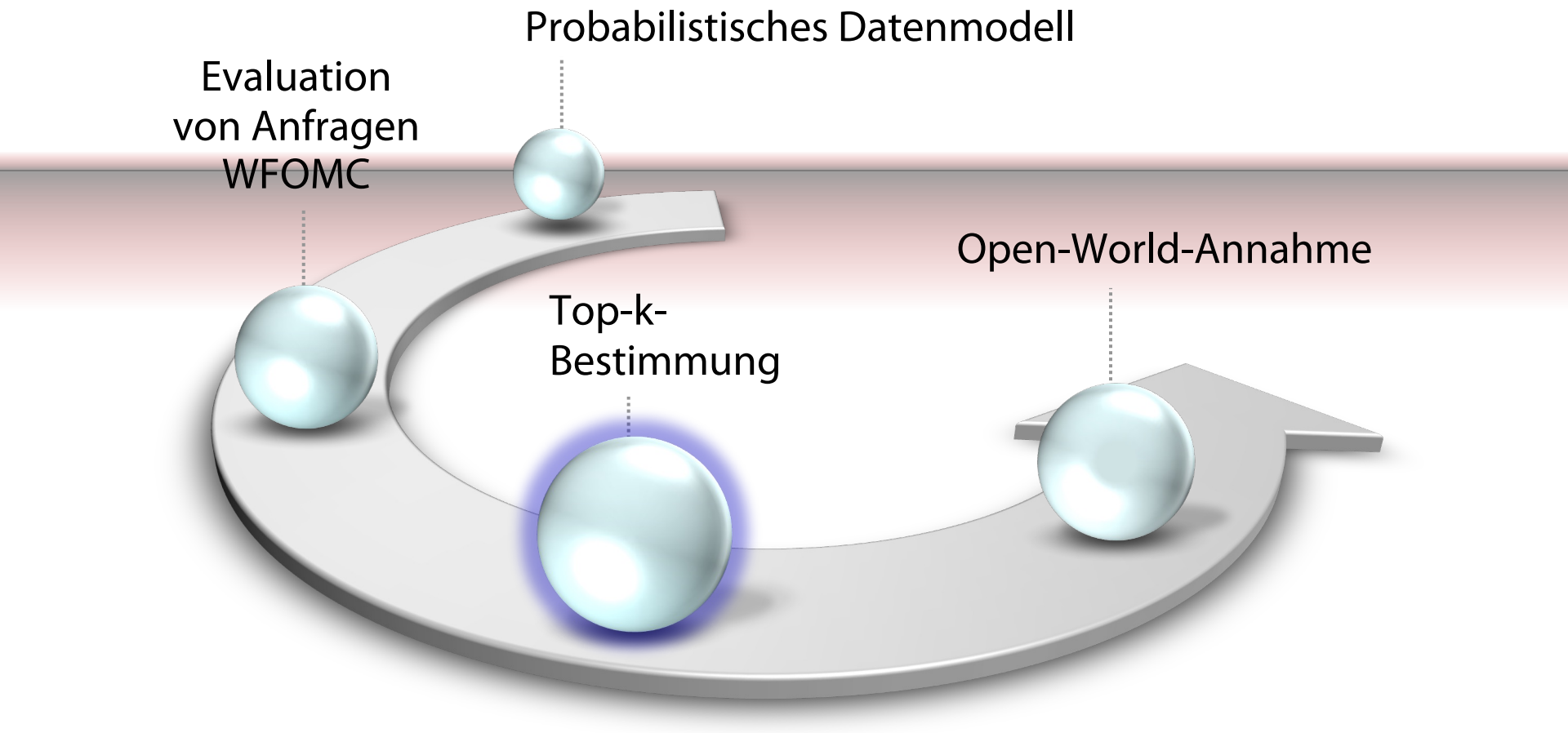
Intensionale Anfrageevaluation

- Boolesche Anfrage Q + ProbDB D
  => Herkunftsformel (*lineage expression)* F
  - Boolesche Variablen e_1, $\underline{e}_1$, ...
    und damit Gewichte w_1, $\underline{w}_1$, ...
    korrespondierend zu Tupeln t_1, t_2, ...
    der extensionalen DB
  - Die Herkunftsformel F sagt, wann Q wahr ist
- Berechne P(F) mit WFOMC

- Für unsichere Anfragen ist das für jedes Ergebnistupel t mit Formel F(t) zu berechnen.

Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Top-k-Anfragebeantwortung

Gegeben: $G = \{t_1, \ldots, t_n\}$ eine Menge von n Objekten mit unbekannten Wahrscheinlichkeiten $p_1, \ldots, p_n$ und eine Zahl $k \leq n$.

Ziel: Finde Menge von k Objekten mit höchsten Wahrscheinlichkeiten, genannt Top-k-Teilmenge von G

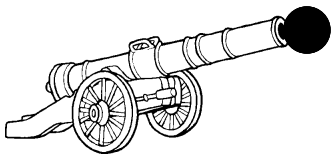
Lösungsidee: Verwende parallele Monte-Carlo-Simulationen, eine für jede Kandidatenantwort (Lineage-Formel in DNF), und approximiere Wahrscheinlichkeiten nur soweit, wie es nötig ist, um die k besten Antworten zu finden

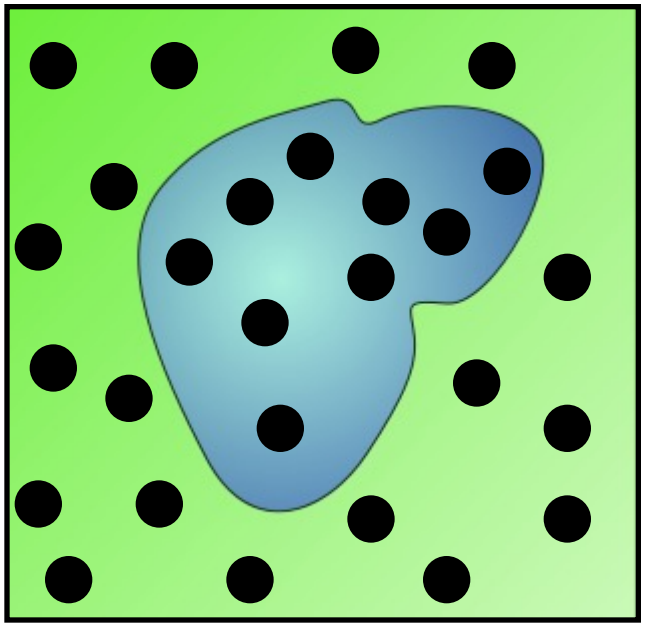
Monte-Carlo-Simulation: Einführung

- Gegeben: Gesamtfläche eines Landstücks (Quadrat)
- Wie kann die Fläche des darauf liegenden Sees bestimmt werden?

$\text{Fläche}_{\text{Land}} = 1000\ m^2$ (Quadrat)
X = Anzahl Kanonenschüsse
N = Einschläge auf grünem Land

$$\frac{\text{Fläche}_{\text{Land}}}{\text{Fläche}_{\text{See}}} = \frac{X}{X-N} \Rightarrow \text{Fläche}_{\text{See}} = \frac{(X-N)\times \text{Fläche}_{\text{Land}}}{X}$$





$\text{Fläche}_{\text{See}} = 1000$

$\text{Fläche}_{\text{See}} = 500$

$\text{Fläche}_{\text{See}} = 333.\overline{3}$

...

$\text{Fläche}_{\text{See}} = 375$

Exkurs

- Gegeben DNF-Formel
- Beispiel: $(A \wedge \neg \mathrm{B} \wedge \mathrm{C}) \vee (\neg A \wedge \mathrm{B} \wedge \mathrm{C}) \vee (\neg \neg A \wedge \mathrm{B} \wedge \neg \mathrm{C})$
- Erfüllbarkeitstest ist einfach:
  - Schaue, ob Muster $\neg \mathrm{X} \wedge \mathrm{X}$ in allen Disjunktionen auftritt
- Wir wollen aber die Anzahl der erfüllenden Belegungen haben!

Function MC-#SAT-Naiv(G)

Wähle N mal zufällig eine mögliche Welt. Man spricht auch von Simulation.

Berechne Wahrheitswert von DNF-Formel G.

Wahrscheinlichkeit $p = P(G)$ approximiert durch Frequenz, mit der G wahr wird

Monte Carlo Simulation: (ε, δ)-Approximation

Function MC-#SAT-Karp-Luby(G)

Fix an order on the disjuncts: $\{t_1, t_2, \ldots, t_m\} = G$

$C := 0$ *// the number of important disjuncts*

repeat // importance sampling

Choose a random disjunct $t_i \in G$

Choose a random truth assignment s.t. $t_i = true$

if forall $j < i$ *it holds that* $t_j = false$ then $C := C + 1$

until *N times*

return C/N

N may need to be very large

$\tilde{p}$:= MC-#SAT-Karp-Luby(G) $\varepsilon = \sqrt{4m \log(2/\delta)/N}$ $a^N = \tilde{p} - \varepsilon$ $b^N = \tilde{p} + \varepsilon$

$$\mathbf{P}(p \in [a^N, b^N]) \quad > \quad 1 - \delta$$

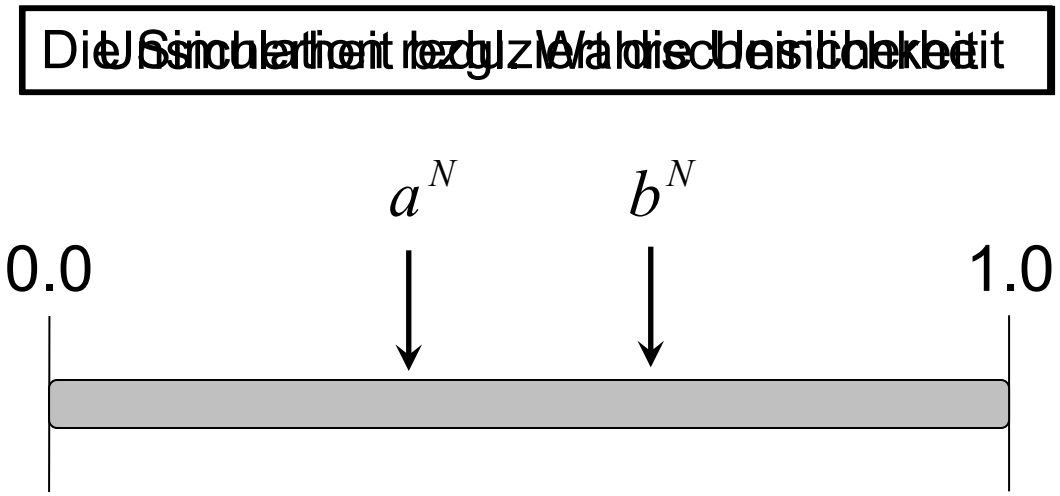
Richard M. Karp, Michael Luby, Monte-Carlo Algorithms for Enumeration and Reliability Problems. STOCS: 56-64, **1983**.

R. M. Karp, M. Luby, and N. Madras. Monte-Carlo approximation algorithms for enumeration problems. J. Algorithms 10, 3 (September 1989), 429-448. **1989**

Karp-Luby-Madras

Nach N Simulationen garantiert:

$$\mathbf{P}(p \in [a^N, b^N]) \quad > \quad 1 - \delta$$ Konfidenzintervall



Anfragebeantwortung

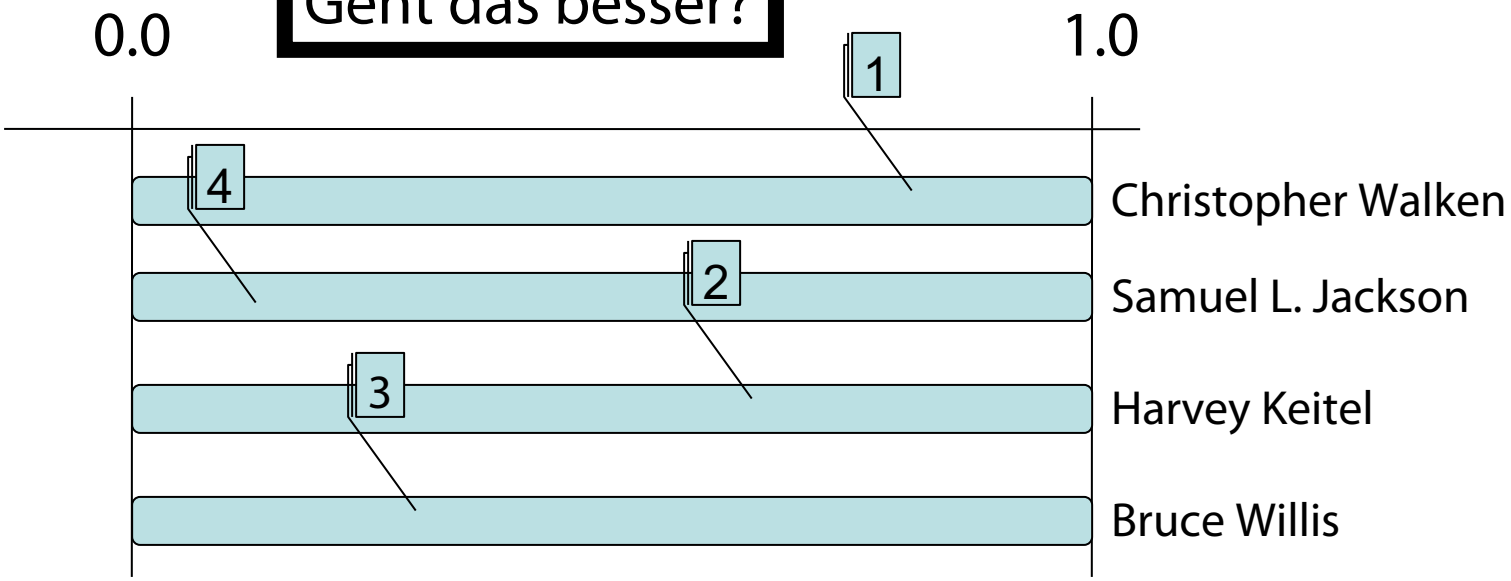
Anfrage: Finde Top-k Regisseure von guten Filmen (Score ≥ 4)
NB: Ranking nach P-Werten

Verfahren: Simuliere für jeden Kandidaten mit Karp-Luby bis ε klein

MID	Director
m232	Christopher Walken
…	…

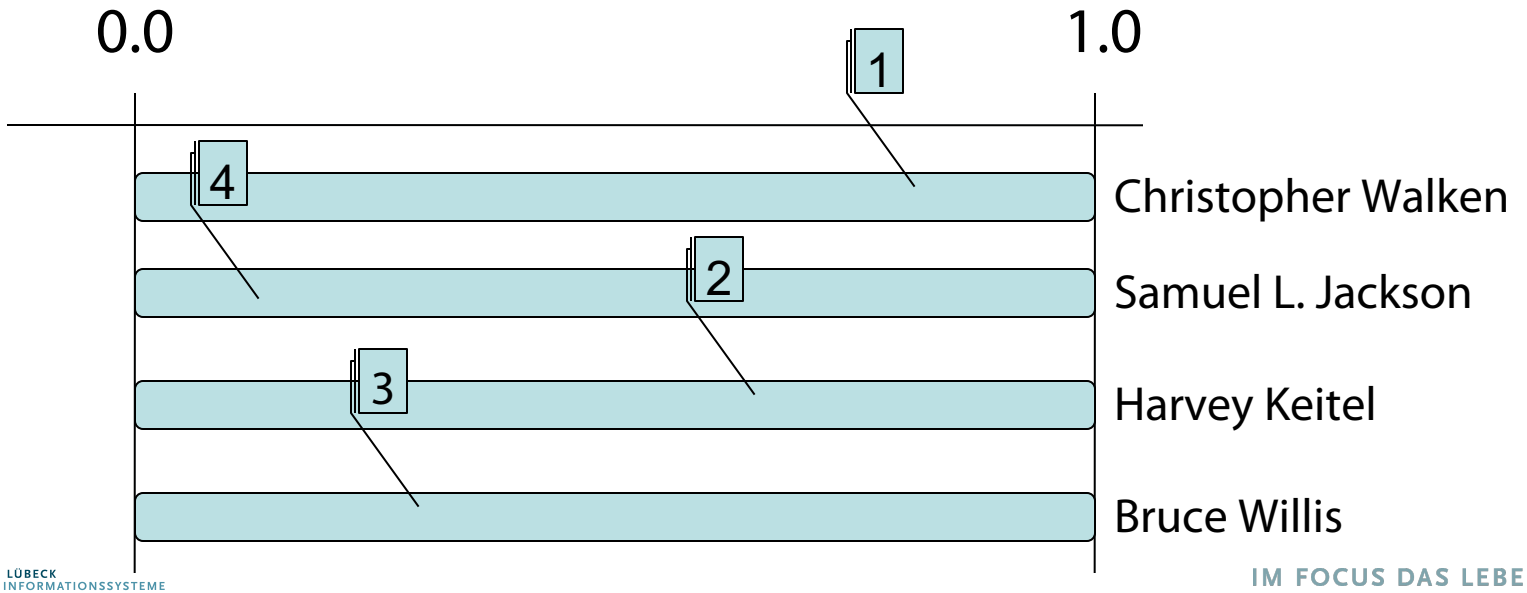
RID	Score
r157	4
…	…

Geht das besser?



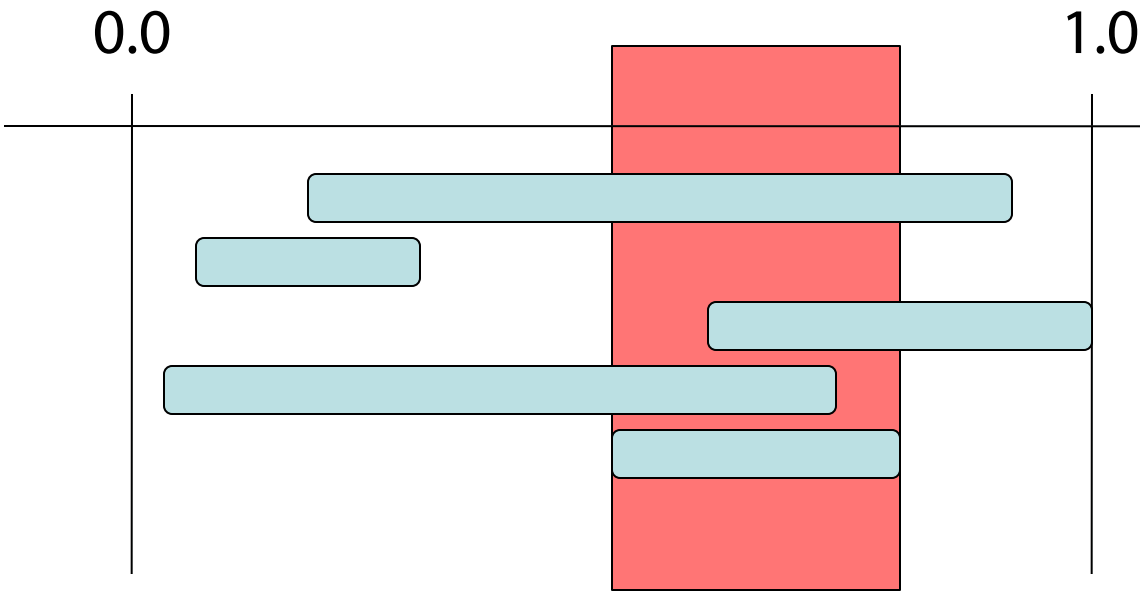
Besseres Verfahren: Multisimulation

- Trenne Top-K mit wenigen Simulationen
  - Betrachte Intervalle im Top-k-Bereich
  - Intervalle ggf. verschränkt
- Vergleich mit imaginärem Verfahren *OPT*
  - *OPT* "weiß," welche Intervalle betrachtet werden müssen
  - Konstruiert aber Beweis



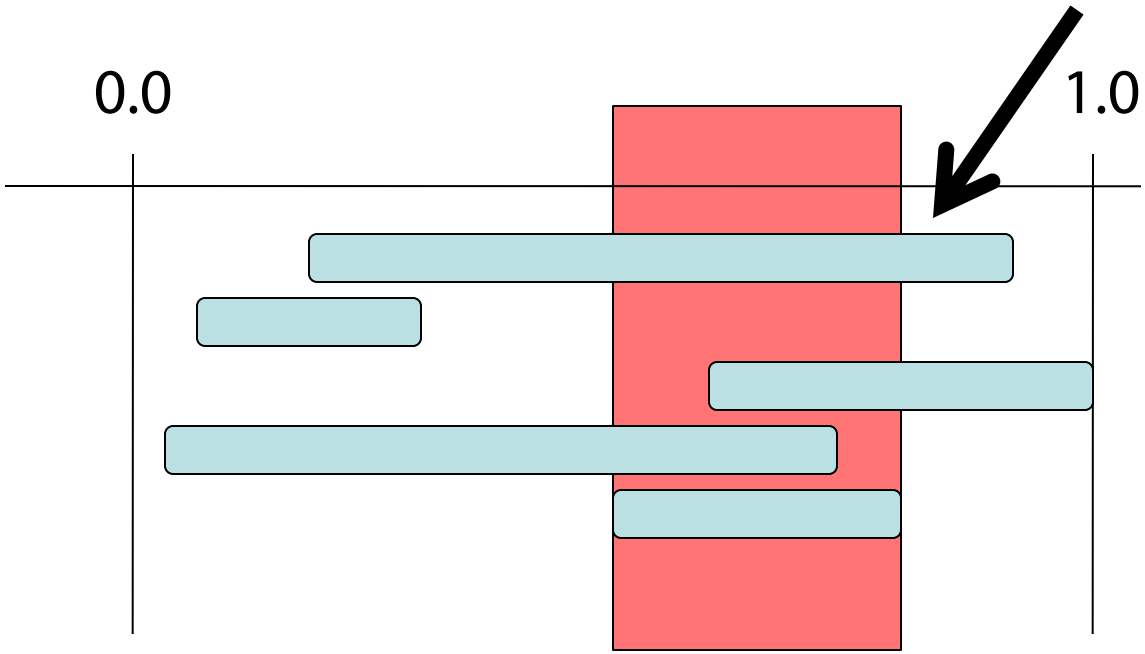
Karp-Luby-Madras im Einsatz: Die kritische Region

- Die kritische Region ist das Intervall
  - (min(k-höchste Min), min(k+1-höchste Max))
  - Für k = 2



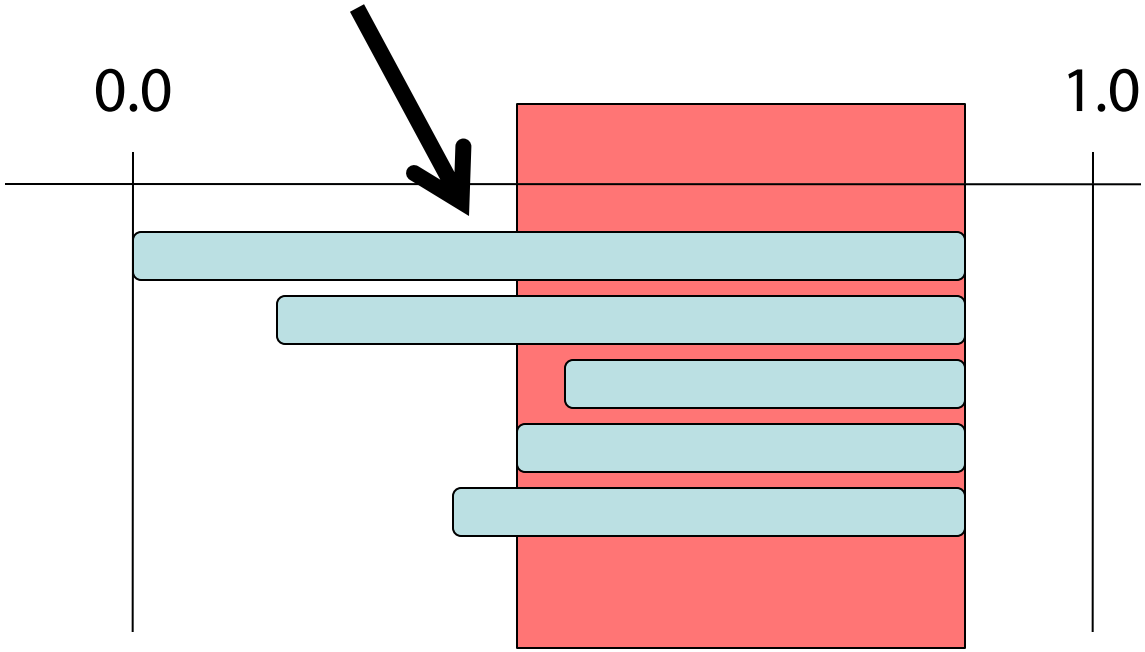
Drei einfache Regeln: Regel 1

- Wähle “Double Crosser”
  - *OPT* muss dieses Intervall auch wählen



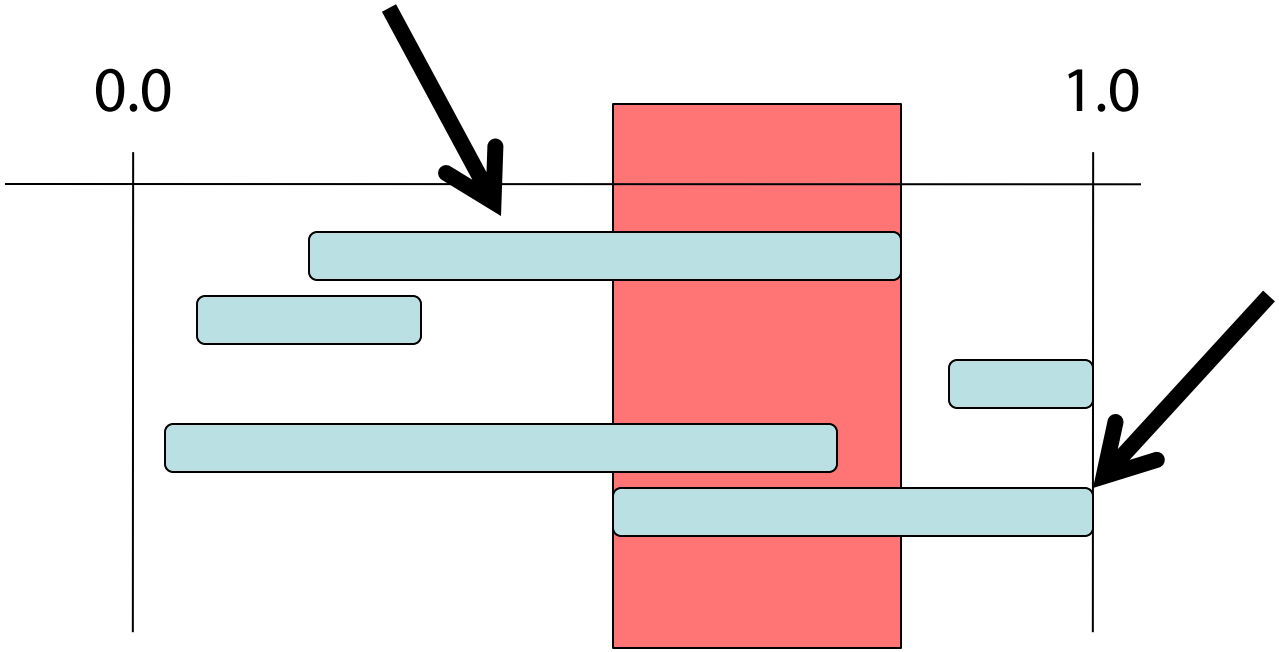
Drei einfache Regeln: Regel 2

- Nur noch "Lower/Upper Crosser"? dann wähle maximales Intervall
  - *OPT* muss das auch machen



Drei einfache Regeln: Regel 3

- Wähle Upper- und Lower-Crosser
  - *OPT* könnte nur ein Intervall wählen



Multisimulation (MS)

function MS_TopK($\{G_1, \ldots, G_n\}$, k):

  $[a_1, b_1] := \ldots := [a_n, b_n] = [0, 1]$

repeat

  $(c, d) := (\min(\text{topk}(a_1, \ldots, a_n)), \min(\text{topk+1}(b_1, \ldots, b_n)))$

  $T := \{G_i \mid d \leq a_i\}$ // top k

  $B := \{G_i \mid b_i \leq c\}$ // no longer to be considered

  Case 1: choose a double crosser to simulate (T, B)

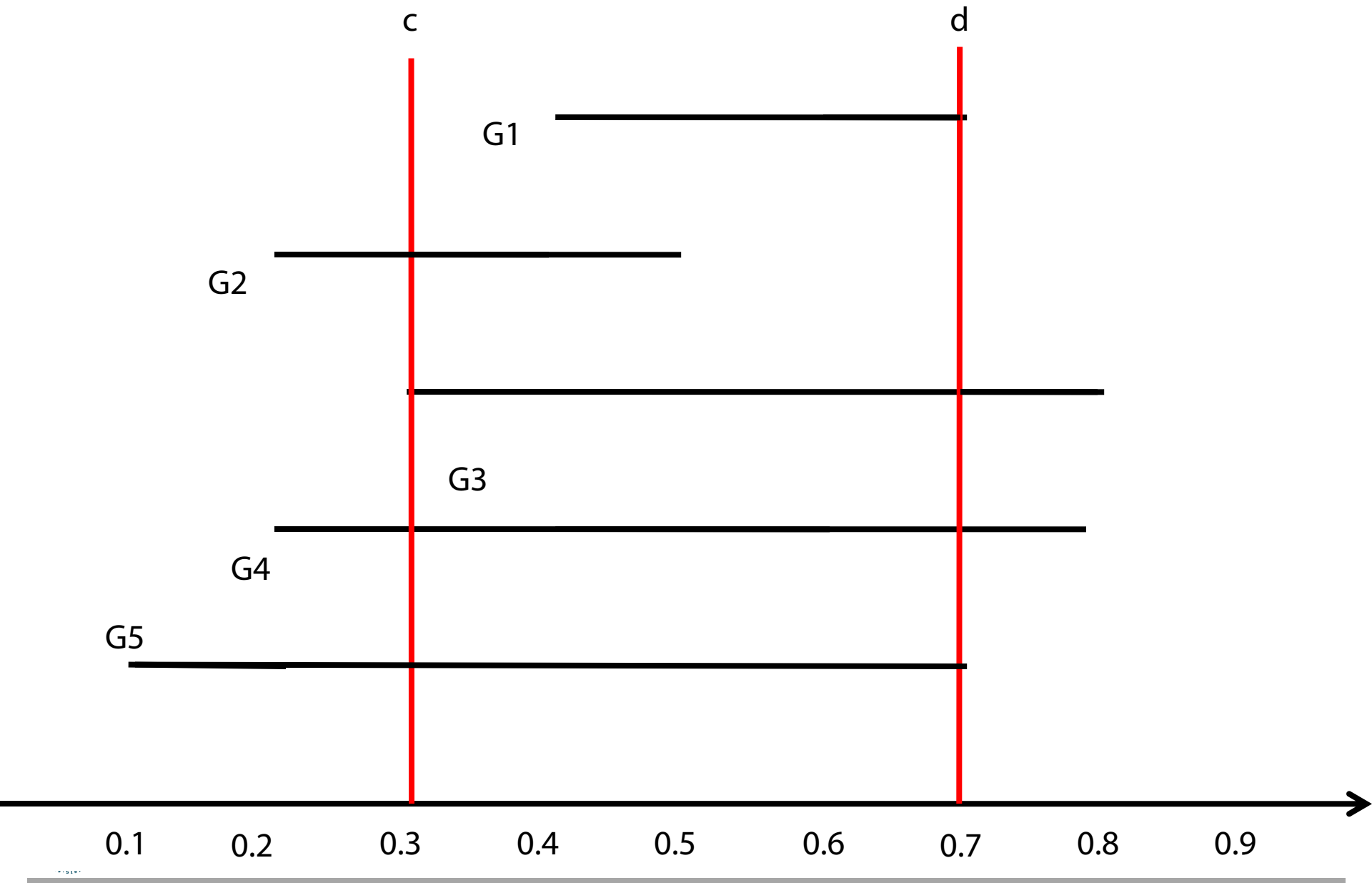
  Case 2: choose upper and lower crosser to simulate (T, B)

  Case 3: choose a maximal crosser to simulate (T, B)

until $c > d$

return T

Beispiel : Wähle Top 2

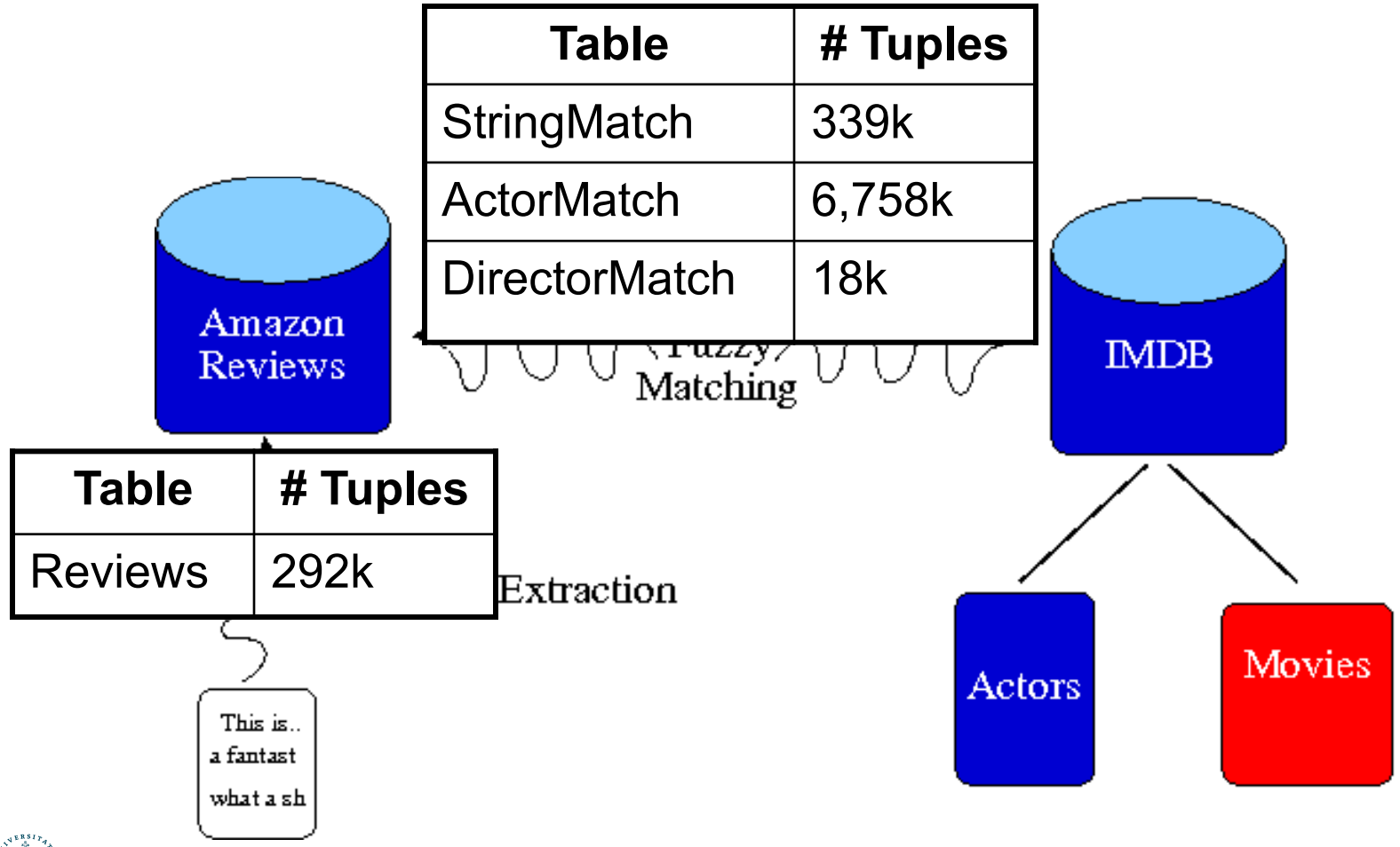


Multisimulation ist 2-approximierend

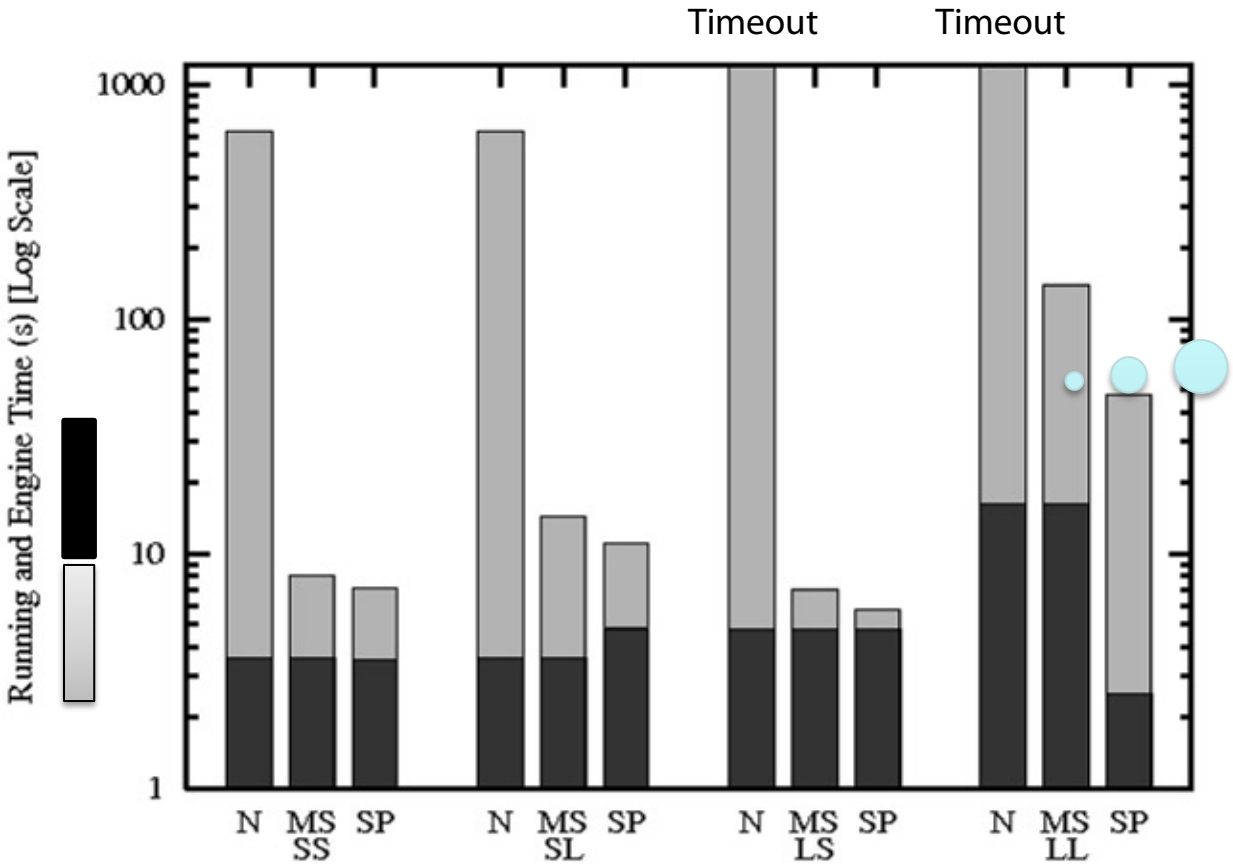
- Theorem [RDS07]: Multisimulation führt höchstens zweimal soviele Simulationen aus wie OPT
  - Und: kein deterministischer Algorithmus kann auf beliebigen Probleminstanzen besser arbeiten
- Varianten
  - Top-k-Menge (gezeigt)
  - Anytime (produziere von 1 bis k)
  - Rang (produziere top-k nach Rang sortiert)
  - Alle (alle Intervalle nach Rang sortiert)

Chris Re, Nilesh Dalvi, Dan Suciu. Efficient Top-k Query Evaluation on Probabilistic Data. Proc. In ICDE, **2007**

Experimente: Unsichere Tupel



Laufzeiten [**N(aiv), MS, and S(afe) P(lan)]**



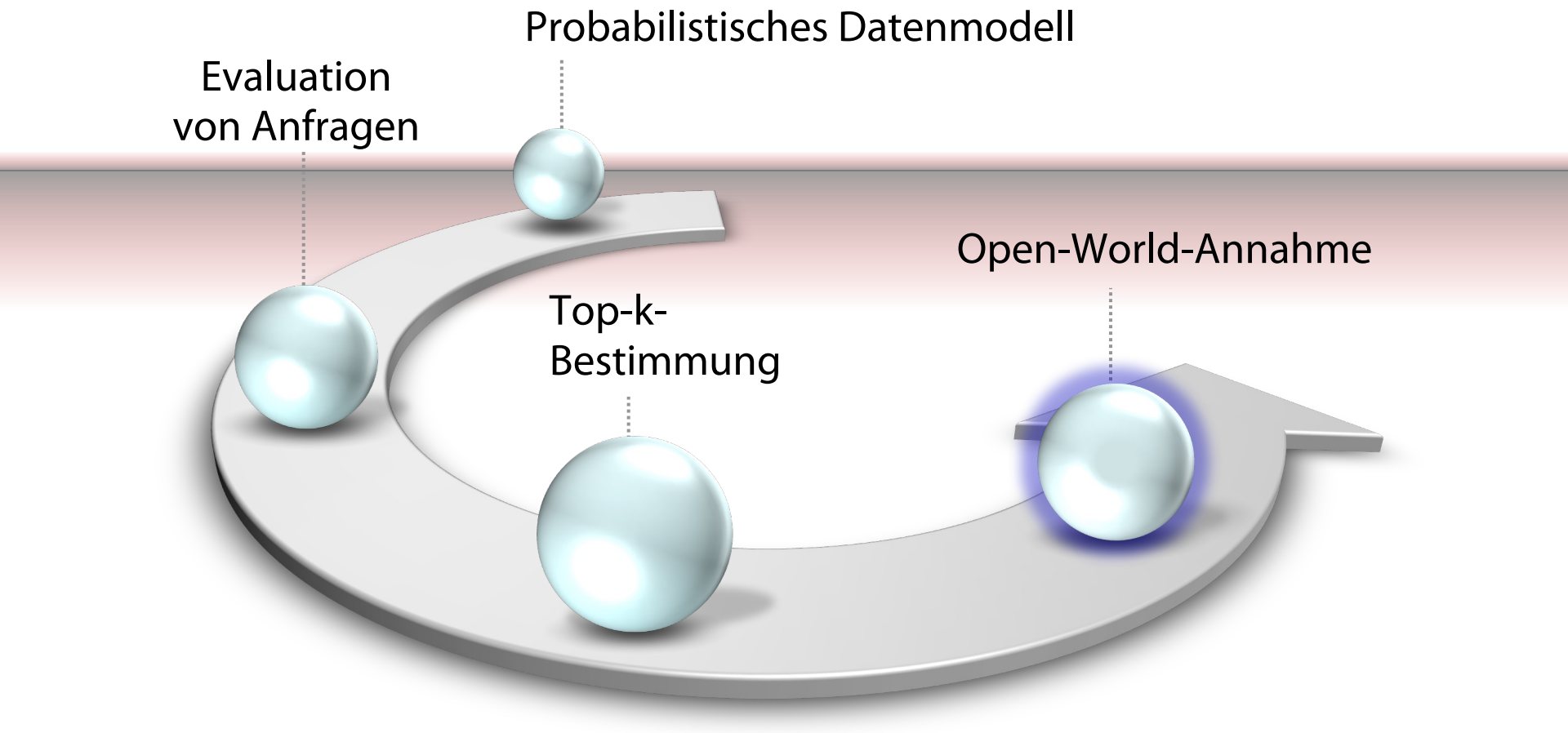
Message:
MS not too
bad compared
to SP

SS=Small # output tuples, small leverage DNF
SL= Small # output tuples, large leverage DNF
LS= Large # output tuples, small leverage DNF
LL= Large # output tuples, large leverage DNF

N(aive)=MC, MS=Multi-Simulation, Safe Plan: siehe vorige Vorlesung

Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Danksagung

Die nachfolgenden Präsentationen sind aus einem Vortrag "Open World Probabilistic Databases" von Ismail Ilkan Ceylan, Adnan Darwiche und Guy Van den Broeck übernommen und wurden etwas modifiziert.

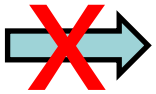
İsmail İlkan Ceylan, Adnan Darwiche, and Guy Van den Broeck. Open-world probabilistic databases. In *Proc. Knowledge Representation and Reasoning* (KR'16), pp. 339-348, **2016**.

CWA vs OWA

Ein Flug taucht nicht in Flüge-Datenbank auf

Der Flug findet nie statt!

In einer Patientenakte ist eine Penicillin-Allergie nicht erwähnt



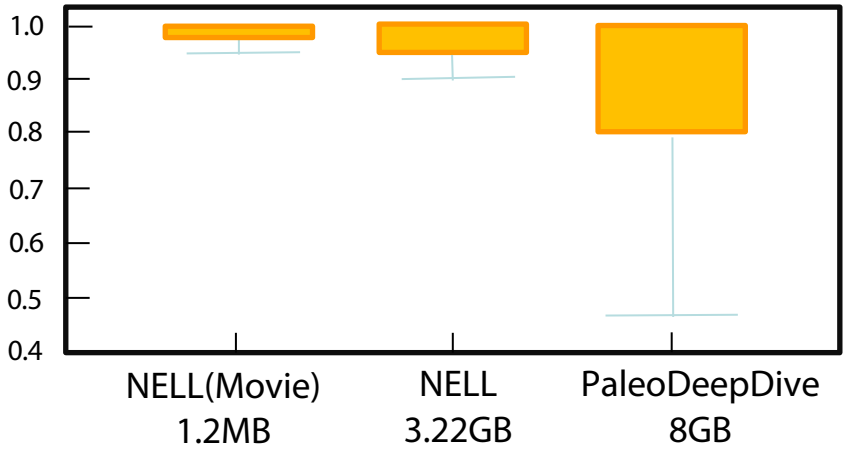
Der Patient leidet nicht an einer Penicillin-Allergie !

Einführung

- Unter der Annahme der geschlossenen Welt (Closed-World Assumption, CWA) wird als falsch angenommen, was nicht als wahr beweisbar ist
- Unter der Annahme der offenen Welt (Open-World Assumption, OWA) wird das, was nicht beweisbar ist, als unbekannt angenommen
- Das klassische PDB-Modell verwendet CWA
  - Fakten nicht in der DB haben Wahrscheinlichkeit 0
- Offene PDBs:
  - Fakten nicht in der DB haben "unbekannte" Wahrscheinlichkeit

Klasssische Anwendung

- Faktenextraktion aus Texten (DeepDive, Nell, Yago)
- Fakten beschrieben mit Sicherheitswerten
  - Deutung: Wie wahrscheinlich ist es, dass Faktum wahr?
- Verteilung der Sicherheitswerte:



- Vervollständigung von fehlenden Einträgen? Unmöglich!

Probleme bei CWA in PDBs: Beispiele

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7

Inmovie		P
w.Smith	ali	0.9
w.Smith	sharktable	0.8
j.james	ali	0.6
arquette	scream	0.7
pitt	mr ms smith	0.5
jolie	mr ms smith	0.7
jolie	sharktable	0.9

$$Q_1(x, y) = \text{Inmovie}(x, z), \text{Inmovie}(y, z), \text{Couple}(x, y)$$

$$Q_2 = \text{Inmovie}(x, z), \text{Inmovie}(y, z), \text{Couple}(x, y)$$

$Q_1(\text{pitt}, \text{jolie})$

Q_2

Erwartet:

$$P\left(Q_1(\text{pitt}, \text{jolie})\right) < P(Q_2)$$

Gefunden:

$$P\left(Q_1(\text{pitt}, \text{jolie})\right) = P(Q_2) = 0.28$$

Probleme bei CWA in PDBs: Beispiele

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7

Inmovie		P
w.Smith	ali	0.9
w.Smith	sharktable	0.8
j.james	ali	0.6
arquette	scream	0.7
pitt	mr ms smith	0.5
jolie	mr ms smith	0.7
jolie	sharktable	0.9

$$Q_1(x, y) = \text{Inmovie}(x, z), \text{Inmovie}(y, z), \text{Couple}(x, y)$$

$Q1(\text{w. smith}, \text{j. james})$

$Q1(\text{thornton}, \text{aniston})$

Erwartet:

$$P\left(Q_1(\text{w. smith}, \text{j. james})\right) > P(Q1(\text{thornton}, \text{aniston}))$$

Gefunden:

$$P\left(Q_1(\text{w. smith}, \text{j. james})\right) = P(Q1(\text{thornton}, \text{aniston})) = 0$$

Probleme bei CWA in PDBs: Beispiele

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7

Inmovie		P
w.Smith	ali	0.9
w.Smith	sharktable	0.8
j.james	ali	0.6
arquette	scream	0.7
pitt	mr ms smith	0.5
jolie	mr ms smith	0.7
jolie	sharktable	0.9

$$Q_1(x, y) = \text{Inmovie}(x, z), \text{Inmovie}(y, z), \text{Couple}(x, y)$$

$Q_1(\text{w. smith}, \text{j. james})$

$\text{Inmovie}(x, y) \wedge \neg\, \text{Inmovie}(x, y)$

$\text{Inmovie}(x, y) \wedge \neg \text{Inmovie}(x, y)$ ist nicht erfüllbar, wird aber gleich bewertet wie $Q_1(\text{w. smith}, \text{j. james})$

OpenPDBs

Inmovie		P
w.Smith	ali	0.9
w.Smith	sharktable	0.8
j.James	ali	0.6
arquette	scream	0.7
pitt	mr ms smith	0.5
jolie	mr ms smith	0.7
jolie	sharktable	0.9

Couple		P
arquette	cox	0.6
pitt	jolie	0.8
thornton	jolie	0.6
pitt	aniston	0.9
kunis	kutcher	0.7

Domain D

cruise
depp
hayek
pitt
j.James
.
.
Mission Impossible
Troy
Kingdom of Heaven

$\lambda \in [0, 1]$

Offene Tupel für $\lambda = 0.3$:

(Inmovie(pitt, Troy), 0.3)

(Inmovie(hayek, Mission Impossible), 0.15)

OpenPDBs

- Eine offene PDB P_λ ist ein Paar $(\mathcal{P}, \lambda)$,
  wobei $\mathcal{P}$ eine probabilistische DB ist und $\lambda \in [0, 1]$
- Für jedes Tupel nicht in $\mathcal{P}$ fügen wir ein Tupel $\langle t : p \rangle$
  hinzu für irgendein $p \in [0, \lambda]$

- P_λ induziert eine Menge von Wahrscheinlichkeitsverteilungen $K_{P\lambda}$
  - Intervall-basierte Wahrscheinlichkeitsangaben für
    offene Tupel

OpenPDBs

- Das Wahrscheinlichkeitsintervall

  einer Booleschen Anfrage Q an P_λ ist

$$K_{P\lambda}(Q) = [\underline{P}_{P\lambda}(Q), \overline{P}_{P\lambda}(Q)]$$

  wobei:

$$\underline{P}_{P\lambda}(Q) = \min_{P \in K_{P\lambda}} P(Q) \quad , \quad \overline{P}_{P\lambda}(Q) = \max_{P \in K_{P\lambda}} P(Q)$$

Beispiele - CWA vs OWA

$$Q_3 = \mathrm{Ac}(\mathrm{patt}), \mathrm{Workedfor}(\mathrm{patt}, \mathrm{hwicke}), \mathrm{Di}(\mathrm{hwicke})$$

$$Q_4 = \mathrm{Ac}(\mathrm{patt}), \mathrm{Workedfor}(\mathrm{patt}, \mathrm{x}), \mathrm{Di}(\mathrm{x})$$

Erwartet:

$$\bar{\mathrm{P}}\,(Q_4) > \bar{\mathrm{P}}(Q_3)$$

NELL Datenbank:

Gefunden:

$$\bar{\mathrm{P}}\,(Q_4) = \bar{\mathrm{P}}(Q_3) = 0$$

In einer offenen Welt:

Gefunden: für $\lambda = 0.3$

$$\bar{\mathrm{P}}\,(Q_4) = 0.82\,, \quad \bar{\mathrm{P}}(Q_3) = 0.51$$

Ac stands for actor, patt for Pattinson, Di for director, hwicke for Hardwicke, Mov for movie, and trainsp for Trainspotting

Beispiele - CWA vs OWA

$$Q_5 = Ac(x), Inmovie(x, trainsp), Mov(trainsp), \neg Di(x)$$

Erwartet:

$$\bar{\mathrm{P}}\,(Q_5) > 0$$

NELL Datenbank:

Found:

$$\bar{\mathrm{P}}\,(Q_5) = 0$$

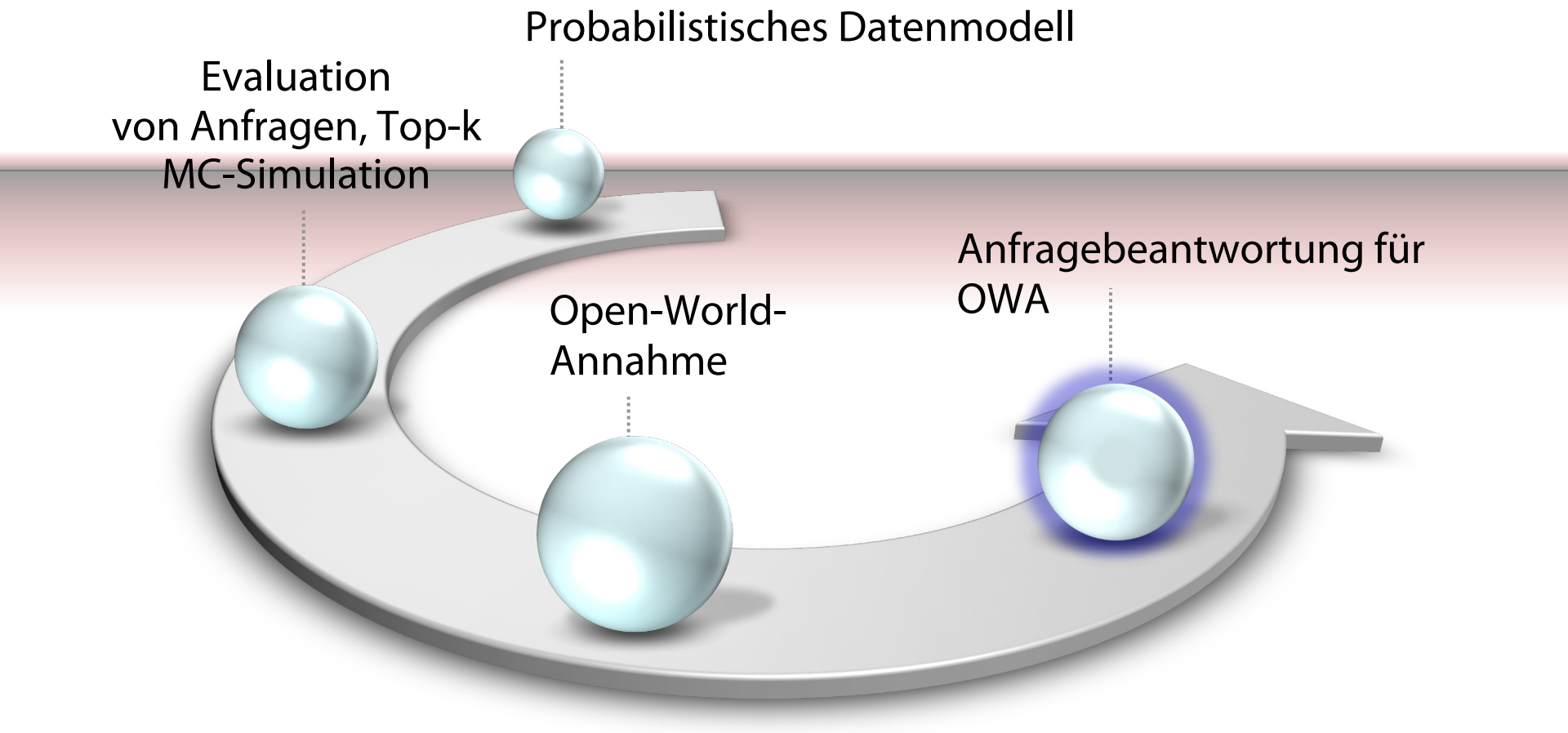
In einer offenen Welt:

Found: for $\lambda = 0.3$

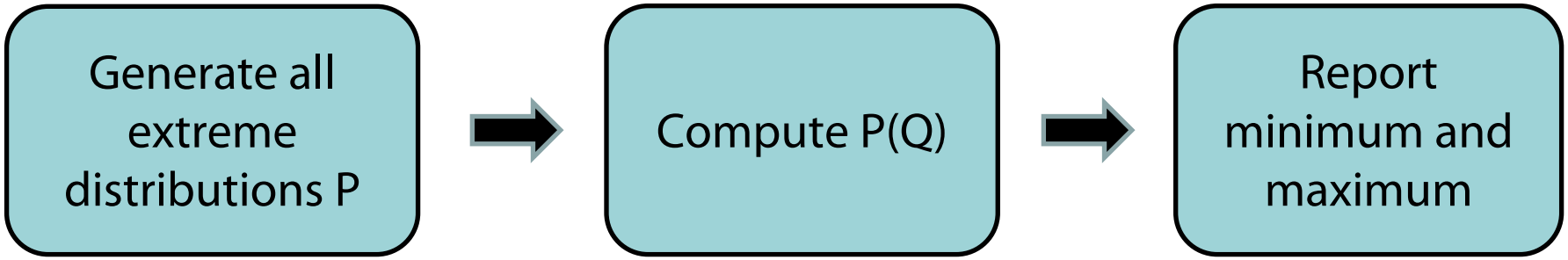
$$\bar{\mathrm{P}}\,(Q_5) = 0.78$$

Non-Standard-Datenbanken und Data Mining

Probabilistische Datenbanken



Naiver Algorithmus



Exponentiell in der Anzahl der Open-World-Tupel

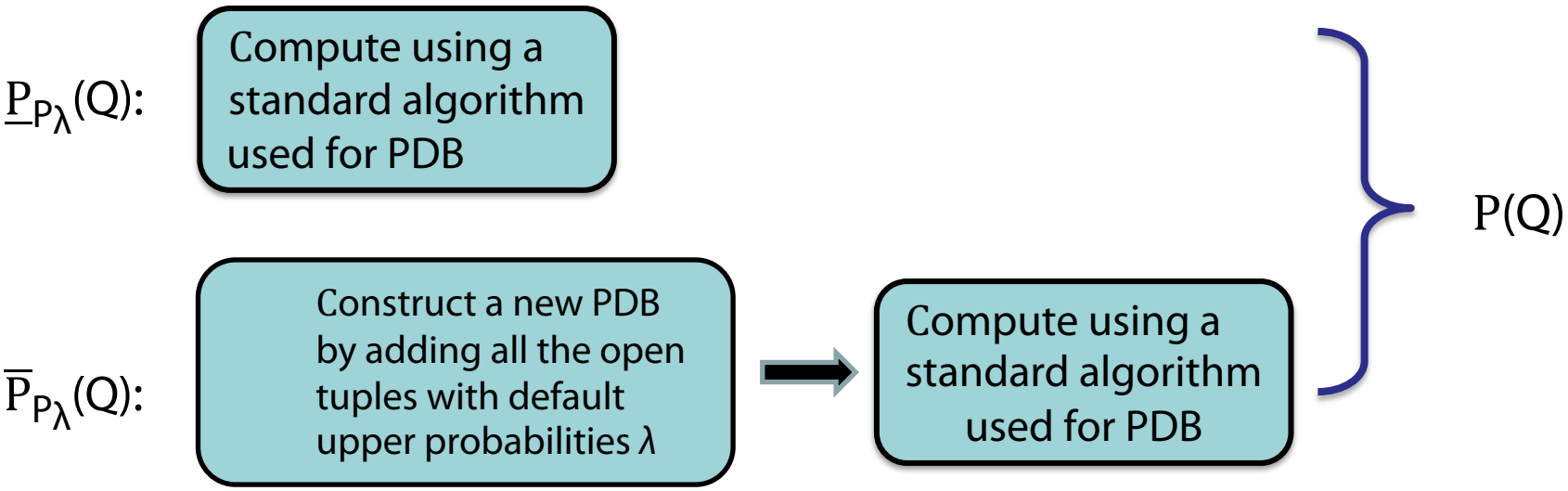
Naiver Algorithmus für UCQs

Beispiel

$$\text{Exist m, a } [\text{Inmovie(a, m)} \text{ AND } \text{Ac(a)}]$$

- Obere Grenze: Jedes Tupel in Inmovie und Ac hat eine maximale Wahrscheinlichkeit
- Untere Grenze: Jedes Tupel in Inmovie und Ac hat eine minimale Wahrscheinlichkeit
- Monotonie für CQs (Intervalle werden nur weiter eingeschänkt) vereinfacht Auswertung

Naiver Algorithmus for UCQs



Bei der Auswertung von $\overline{P}_{P\lambda}(Q)$ wächst die PDB exponentiell in der Domänengröße (n-stellige Relationen)

Neuer Ansatz: $Lift_O^R$ Algorithmus

CNF:
$(R(x) \vee S(y,z)) \wedge (S(x,y) \vee (\mathrm{T}(\mathrm{x}))$

$Lift_O^R$ (Q, P, λ, D) - abbreviated by L(Q, P)

Input: CNF Q , probability tuples Ƥ, threshold λ and domain D

Output: The upper probability $\overline{\mathrm{P}}_{(p,\,\lambda)}(Q)$ over domain D

Step 0: Base of Recursion

if Q is a single ground atom t then

if ⟨t : p⟩ ∈ Ƥ then return p else return λ

Step 1: Rewriting of Query

Convert Q to union of CNFs: $Q_{UCNF} = Q_1 \vee \ldots \vee Q_m$ m = number of disjuncts

Example: $(R(x) \vee S(y,z)) \wedge (S(x,y) \vee (\mathrm{T}(\mathrm{x}))$

➔ $((R(x)) \wedge (S(x,y) \vee (\mathrm{T}(\mathrm{x}))) \vee ((S(y,z)) \wedge (S(x,y) \vee \big(\mathrm{T}(\mathrm{x})\big))$

$Lift_O^R$ Algorithmus

$Q_1 \perp Q_2$ if Q_1, Q_2 doesn't share any relational symbols

Step 2: Decomposable Disjunction
if m > 1 and $Q_{UCNF} = Q_1 \vee Q_2$ where $Q_1 \perp Q_2$ then
- $q_1 \leftarrow L(Q_1, P|_{Q1})$ and $q_2 \leftarrow L(Q_2, P|_{Q2})$
- return $1 - (1 - q1) \cdot (1 - q2)$

Step 3: Inclusion-Exclusion
if m > 1 but Q_{UCNF} has no independent Q_i then
- return $\sum_{s \subseteq \{1,\dots,m\}} (-1)^{|s|+1} \cdot L(\wedge_{i \in s} Q_i, P|_{\wedge i \epsilon s Q_i})$

Step 4: Decomposable Conjunction
if $Q = Q_1 \wedge Q_2$ where $Q_1 \perp Q_2$ then
- return $L(Q_1, P|_{Q1}) \cdot L(Q2, P|_{Q2})$

$\mathcal{P}|_Q$ denotes the subset of $\mathcal{P}$ that talks about the predicates that appear in Q.

$Lift_O^R$ Algorithmus

Step 5: Decomposable Universal Quantifier

if Q has a separator variable x then

- let T be all constants as x-argument in P
- $q_c \leftarrow \prod_{t \in T} L(Q[x/t], PI_{x=t})$
- $q_o \leftarrow L(Q[x/t], \emptyset)$ for some $t \in D \setminus T$
- return $q_c \cdot q_o^{|D \setminus T|}$

A separator is a variable that appears in every atom in Q

An x-argument is an argument that holds a separator variable x

Forall a,m [Inmovie(a,m) OR Ac(a)]

Inmovie	
Pitt	Troy
Butler	300
deCaprio	Inception

Ac
Pitt
Cruise
Hayek

Step 6: Fail

T = {Pitt, Butler, deCaprio, Cruise, Hayek}

$Lift_O^R$ Algorithmus für UCQs

$Lift_O^R$ berechnet die Wahrscheinlichkeiten für UCQs

Daten-Komplexität:

UCQs werden auf OpenPDBs in PTime evaluiert, wenn sie auf PDBs in PTime evaluiert werden und umgekehrt.

Zusammenfassung

- Einfache PDBs erfüllen nicht (mehr) die heutigen Erwartungen (Vervollständigungen nicht möglich)
- OpenPDBs setzen die Annahme der offenen Welt um
- Vorgestellt wurde ein effizienter Algorithmus zur Auswertung von UCQs in OpenPDBs
  - Ob die Semantik mit λ Bestand hat, wird die Zeit zeigen

İsmail İlkan Ceylan, Adnan Darwiche, Guy Van den Broeck, Open-World Probabilistic Databases, In Chitta Baral and James P. Delgrande and Frank Wolter, eds., Proceedings of 15. International Conference on Principles of Knowledge Representation and Reasoning (KR 2016), 339--348, 2016. AAAI Press, **2016**.

Ismail Ilkan Ceylan, Adnan Darwiche and Guy Van den Broeck. Open-World Probabilistic Databases: An Abridged Report, In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI), Sister Conference Best Paper Track, **2017**.