Probabilistic Databases for Dealing with Missing Values Governed by Missingness Mechanisms

Leopoldo Bertossi

Joint work with: **Farouk Toumani**
(U. Clermont-Ferrand, France)

Incomplete Data: A Data Quality Dimension

- Incompleteness of data is one of the main data quality concerns

  So as inconsistency of data w.r.t. integrity constraints (ICs)

- Data completion is one of the most important, common and costly data cleaning activities

- Most commonly, data imputation methods are applied
  That is, filling in for missing data

  There are multiple data imputation techniques; some more *ad hoc* than others

- In this talk incompleteness refers to missing attribute values

  Not to missing tuples or open-world DBs

Missing Values and Consistency

	A	B	C
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

- We want to do imputation

  But consistently!

- Consistently w.r.t. what?

  A specific form of external knowledge

- Giving rise to a set of preferred possible worlds!

  Very much in the spirit of DB repairs

- That knowledge comes as known Missingness Mechanisms

  MMs tell us under what conditions MVs may appear

- They were introduced by Donald Rubin

  Famous for Causality in Observational Studies

- For example, a MM could tell us

  "With high probability a MV appears in C when B takes value 1"

	A	B	C
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

- We assume MMs are represented as Bayesian Networks

  Becoming Missingness Graphs (MGs)

- They are well-known and studied

  They convey a causal semantics

  They can be easily combined with data management

- Some typical MMs have been identified and used in practice

  We will represent them as MGs (coming)

DBs with Missing Values

- A^o, B^o always observed variables (no MVs)

  $C^\star$ is the observed version of underlying variable C^m that may have MVs

- There is some true but unknown MV-free DB D underneath

	A^o	B^o	$C^\star$
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

"observed" DB $D^\star$

- For each attribute C^m that may exhibit MVs via $C^\star$, introduce a missingness indicator variable $\mathbb{I}^C$

  Deterministically or with probability 1:

  $C^\star = \text{na}$ when $\mathbb{I}^C = 1$ $\qquad$ $C^\star = C^m$ when $\mathbb{I}^C = 0$

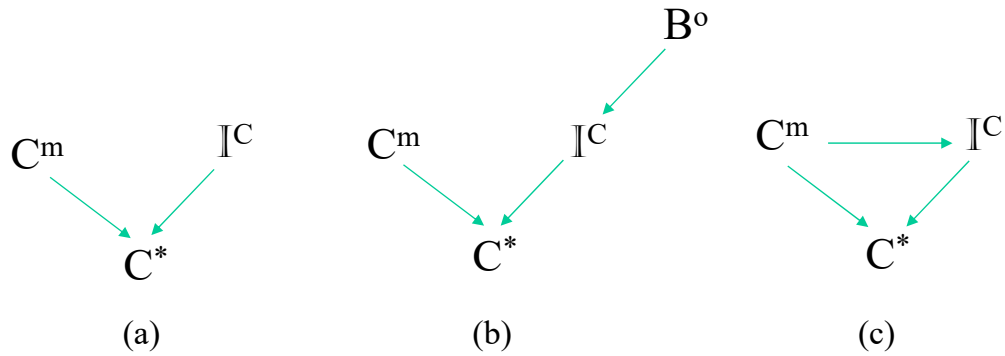
- Values for $C^\star$, including MVs, depend on C^m and possibly other attributes via $\mathbb{I}^C$

- Occurrence of MVs governed by *Missingness Mechanisms*

  Dependencies upon the same or other variables in this regard

- MMs represented by Causal or BNs

  (K. Mohan & J. Pearl)



(a) MVs for C^m (shown in $C^\star$) depend on RV $\mathbb{I}^C$

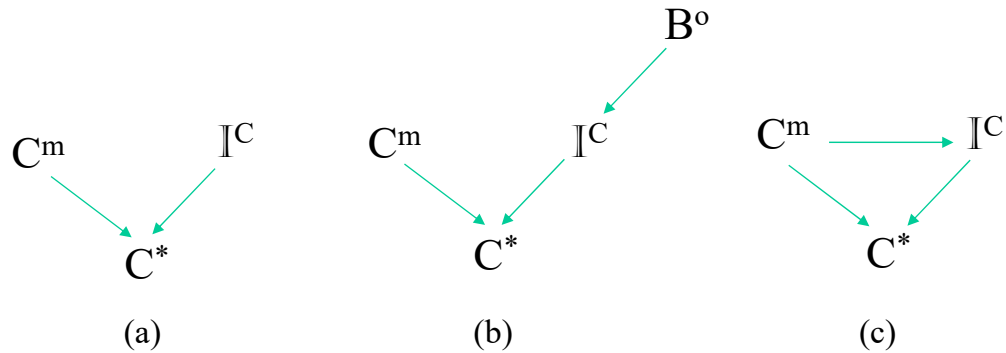
Value of $\mathbb{I}^C$ obtained, e.g., flipping a coin

MG represents a "Missing Completely at Random" (MCAR) MM for C^m

(b) MVs for C^m depend on fully observed variable B^o

For example, flip a coin only when B^o takes value 1

This is a case of "Missing at Random" (MAR) for C^m



(c) A case of "Missing Not Completely at Random" (MNCAR)

Complex dependencies that determine occurrence of MVs

Common particular case: Self-Censorship

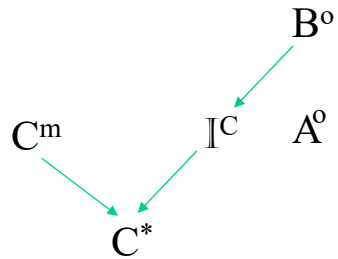
A variable determines its own MVs

For example, a salary is shown as a MV when the salary is high (with high probability)

- MMs can be used to do classic imputation
- BNs as MGs determine a joint probability distribution $P^{\mathcal{M}}$ over tuples

DB Representation and QA

- Back to our example:



	A^o	B^o	$C^\star$
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

"observed" DB $D^\star$

- Values for $C^\star$, including MVs, depend on C^m and B^o via $\mathbb{I}^C$ (MAR)
- Want to query "underlying" DB D, but we only have $D^\star$
- Many possible D's ...
  - Determined by what?
  - Which one?
  - With what properties?
  - Computed how?

- Instead of usual imputation, build a Probabilistic DB
- Each underlying value for C^m has a probability of being the missing one

  As determined by MG $\mathcal{M}$ and other information in the tuple

	A^o	B^o	$C^\star$
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

$Dom(C^m) = \{0, 1, 2\}$

- We want:

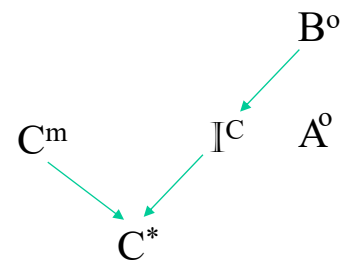
$$p_3^1 := P^{\mathcal{M}}(C^m = 0 \mid A^o = a, B^o = 1, C^\star = \text{na})$$
$$p_3^2 := P^{\mathcal{M}}(C^m = 1 \mid A^o = a, B^o = 1, C^\star = \text{na})$$
$$p_3^3 := P^{\mathcal{M}}(C^m = 2 \mid A^o = a, B^o = 1, C^\star = \text{na})$$

	A^o	B^o	$C^\star$
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

- Assume: (distributions in $\mathcal{M}$)

B^o	$\mathbb{I}^C$	$P(\mathbb{I}^C \mid B^o)$
0	0	1
0	1	0
1	0	1/4
1	1	3/4

$$P^{\mathcal{M}}(C^m = 0) := \tfrac{1}{2}$$
$$P^{\mathcal{M}}(C^m = 1) = P^{\mathcal{M}}(C^m = 2) := \tfrac{1}{4}$$



- Here: $p_3^1 := P^{\mathcal{M}}(C^m = 0 \mid A^o = a, B^o = 1, C^\star = \text{na})$
  $= P^{\mathcal{M}}(C^m = 0) = \frac{1}{2}$

	A^o	B^o	$C^\star$
τ_1	a	0	0
τ_2	a	0	0
τ_3	a	1	na
τ_4	a	1	0
τ_5	a	1	na
τ_6	a	1	1
τ_7	a	1	na
τ_8	a	1	2

- A Block-Independent PDB (BID) D^p
- $\mathcal{W}$: collection of possible worlds, picking one tuple per block

  Each with a probability $P^{\mathcal{W}}(W)$, product of the chosen tuples' probabilities

	A^o	B^o	C^m	p^{BID}
τ_1	a	0	0	1
τ_2	a	0	0	1
$\mathcal{B}(\tau_3)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_4	a	1	0	1
$\mathcal{B}(\tau_5)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_6	a	1	1	1
$\mathcal{B}(\tau_7)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_8	a	1	2	1

- MV-free subinstances of D^p
- A possible world with probability $(\frac{1}{2})^3$:

$W_1 := W^{[000]} :=$

	A^o	B^o	C^m
τ_1	a	0	0
τ_2	a	0	0
τ_3^1	a	1	0
τ_4	a	1	0
τ_5^1	a	1	0
τ_6	a	1	1
τ_7^1	a	1	0
τ_8	a	1	2

A most-probable world

World	Notation	$\mathbf{P}^{\mathcal{W}}(\mathbf{W})$
$\mathbf{W_1}$	$\mathbf{W^{[000]}}$	$(\frac{1}{2})^3 = \mathbf{0.125}$
W_2	$W^{[001]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
W_3	$W^{[002]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
W_4	$W^{[010]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
W_5	$W^{[011]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
$\mathbf{W_6}$	$\mathbf{W^{[012]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_7	$W^{[020]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
$\mathbf{W_8}$	$\mathbf{W^{[021]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_9	$W^{[022]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{10}	$W^{[100]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
W_{11}	$W^{[101]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
$\mathbf{W_{12}}$	$\mathbf{W^{[102]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{13}	$W^{[110]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{14}	$W^{[111]}$	$(\frac{1}{4})^3 = 0.015$
W_{15}	$W^{[112]}$	$(\frac{1}{4})^3 = 0.015$
$\mathbf{W_{16}}$	$\mathbf{W^{[120]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{17}	$W^{[121]}$	$(\frac{1}{4})^3 = 0.015$
W_{18}	$W^{[122]}$	$(\frac{1}{4})^3 = 0.015$
W_{19}	$W^{[200]}$	$(\frac{1}{2})^2 \times \frac{1}{4} = 0.06$
$\mathbf{W_{20}}$	$\mathbf{W^{[201]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{21}	$W^{[202]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
$\mathbf{W_{22}}$	$\mathbf{W^{[210]}}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{23}	$W^{[211]}$	$(\frac{1}{4})^3 = 0.015$
W_{24}	$W^{[212]}$	$(\frac{1}{4})^3 = 0.015$
W_{25}	$W^{[220]}$	$(\frac{1}{4})^2 \times \frac{1}{2} = 0.03$
W_{26}	$W^{[221]}$	$(\frac{1}{4})^3 = 0.015$
W_{27}	$W^{[222]}$	$(\frac{1}{4})^3 = 0.015$

- We can now define and do QA as usual on PDBs

- For a BCQ $\mathcal{Q}$: $P^{\mathcal{W}}(\mathcal{Q}) := \sum_{\substack{W \in \mathcal{W} \\ W \models \mathcal{Q}}} P^{\mathcal{W}}(W)$
- For a scalar, numerical aggregate query $\mathcal{Q}$:

$$\mathcal{Q}[D^P] := \sum_{W \in \mathcal{W}} \mathcal{Q}[W] \times P^{\mathcal{W}}(W) = \mathbb{E}^{P^{\mathcal{W}}}(\mathcal{Q})$$

- Coming back to this ...
- However:

  There are MAR MGs and BCQs for which QA is #P-hard in data-complexity

  QA on TIDs (tuple-independent PDBs) can be reduced to our case

Classes of Matching Possible Worlds

- Can we do something better?
- Imputation may lead to duplicate tuples (except for the tids)

  We adopt a bag-semantics
- Some worlds are "matching", i.e. identical as multisets

	A^o	B^o	C^m
τ_1	a	0	0
τ_2	a	0	0
τ_3^1	**a**	**1**	**0**
τ_4	a	1	0
τ_5^3	**a**	**1**	**2**
τ_6	a	1	1
τ_7^1	a	1	0
τ_8	a	1	2

	A^o	B^o	C^m
τ_1	a	0	0
τ_2	a	0	0
τ_3^3	**a**	**1**	**2**
τ_4	a	1	0
τ_5^1	**a**	**1**	**0**
τ_6	a	1	1
τ_7^1	a	1	0
τ_8	a	1	2

matching worlds

$$P^{\mathcal{W}}(W^{[020]}) = 1 \times 1 \times \tfrac{1}{2} \times 1 \times \tfrac{1}{4} \times 1 \times \tfrac{1}{2} \times 1.$$

$$P^{\mathcal{W}}(W^{[200]}) = 1 \times 1 \times \tfrac{1}{4} \times 1 \times \tfrac{1}{2} \times 1 \times \tfrac{1}{2} \times 1.$$

- Group them in classes of matching worlds
- Probability of a class is the sum of the worlds' probabilities
- All worlds in a class return the same query answer

Most-Compliant Classes

- Some worlds, so as their classes, may be more "compliant" with the MG than others ...

Classes C	Worlds	$P^{\mathcal{C}}(C)$	$KLD(C)$
C_1	$W^{[002]}, W^{[020]}, W^{[200]}$	0.188	0.431
C_2	$W^{[011]}, W^{[101]}, W^{[110]}$	0.094	0.518
C_3	$W^{[012]}, W^{[021]}, W^{[102]}, W^{[120]}, W^{[201]}, W^{[210]}$	0.188	0.452
C_4	$W^{[111]}$	0.016	0.711
C_5	$W^{[001]}, W^{[010]}, W^{[100]}$	0.188	0.431
C_6	$W^{[022]}, W^{[202]}, W^{[220]}$	0.094	0.711
C_7	$W^{[112]}, W^{[121]}, W^{[211]}$	0.047	0.929
C_8	$W^{[000]}$	0.125	0.431
C_9	$W^{[122]}, W^{[212]}, W^{[221]}$	0.047	0.972
C_{10}	$W^{[222]}$	0.016	1.111

- Empirical distribution of members in a class:

  For $t \in T$, the set of different tuples:

$$P^E_W(t) \;:=\; mult^W(t)/||W|| \quad \text{(multiset-counting)}$$

- Kullback-Leibler Divergence compared with $\mathcal{M}$-induced distribution:

$$Div_{KL}(P^E_W \parallel P^{\mathcal{M}}) \;:=\; \sum_{t\in T} P^E_W(t)\ \ln \frac{P^E_W(t)}{P^{\mathcal{M}}(t)}$$

- Here, one most compliant class (there may be several)

Algorithmic and Complexity Results

- Our results depend on our class-based semantics

  Based on an underlying bag-semantics (we have also developed a set-semantics)

- Several hardness results, among them: arXiv:2604.06520
  - Computation of Most-Probable Classes
  - Computation of Most-Compliant Classes
- However, we can efficiently compute a most-compliant class
  - By reduction to computing a "minimum-cost flow" in bipartite graphs
  - Actually, a most probable in that class
- Also a Pareto-optimal class w.r.t. the probability and compliance dimensions (w/ Mourad Baiou; to appear)

- Results apply to a broad family of compliance measures, including KLD
- Once we have such a class, we can compute an MC-world, its probability, and query it
- Our algorithms and implementation are considerably more efficient and accurate than all common baseline, agnostic imputation methods (w/ Ali Boukria; to appear)

  Ours uses additional knowledge in the form of a MG

What Without a Bayesian MG?

- We may have the observed DB $D^\star$, with MVs

  Plus a qualitative MG, i.e. essential a Causal Network

  Then, no joint distribution

- To build the BID, all we need are the conditional probabilities

  $$p_3^1 := P^{\mathcal{M}}(C^m = 0 \mid A^o = a, B^o = 1, C^\star = \text{na})$$ (as on Page 9)

- Can we use $D^\star$?

- Depending on the MM that the MG represents, it may be possible to recover those probabilities (Mohan and Pearl)

- Such is the case for MCAR and MAR MMs

  There are computable estimators that converge in probability

  With MNCAR sometimes yes, sometimes not

Back to the BID

- What about QA directly on our BID?
- Our BID is not a tuple-independent PDB (TID)

  Explicit QA on all possible worlds is inefficient

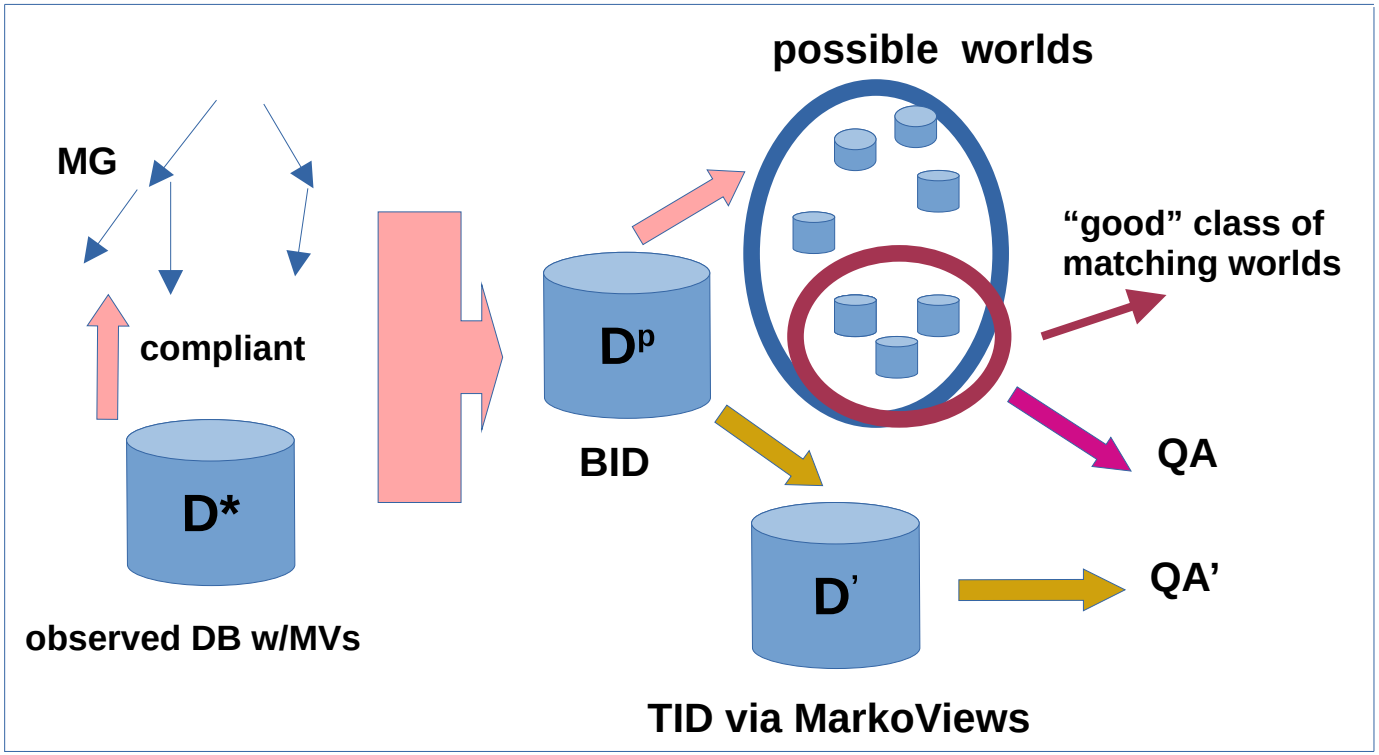
	A^o	B^o	C^m	p^{BID}
τ_1	a	0	0	1
τ_2	a	0	0	1
$\mathcal{B}(\tau_3)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_4	a	1	0	1
$\mathcal{B}(\tau_5)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_6	a	1	1	1
$\mathcal{B}(\tau_7)$	a	1	0	1/2
	a	1	1	1/4
	a	1	2	1/4
τ_8	a	1	2	1

	A^o	B^o	C^m	p^{W_1}
τ_1	a	0	0	1
τ_2	a	0	0	1
τ_3^1	a	1	0	1/2
τ_4	a	1	0	1
τ_5^1	a	1	0	1/2
τ_6	a	1	1	1
τ_7^1	a	1	0	1/2
τ_8	a	1	2	1

a possible world and
a TID, among 27

- For TIDs there are available methods and implementations
- Can we transform the BID into a single TID?

- Use "Marko-Views" (Kumar Jha & Suciu, 2012)



- Impose on the BID constraints à la Markov-Logic Networks

(P. Domingos et al.)

- Requesting: C_1: "at most one tuple per block"
  C_2: "at least one tuple per block"

- Marko-Views are violation views:

$$V_1(b, t, ...)[w_1] \longleftarrow R'(b, t, \ldots), R'(b, t', \ldots), t \neq t'.$$
$$V_2(b)[w_2] \longleftarrow \textit{not } Aux.$$
$$Aux \longleftarrow R'(b, t, \ldots). \quad R' \text{ contains block-ids}$$

- w_1, w_2: numerical weights, penalizing constraint violation

  Equivalently, for having tuples in the views

  Soft and hard constraints can be imposed

- BID's probabilities $p(\tau)$ become tuples' weights
- View-weights are modified: $w^{V_i} := \frac{1-w_i}{w_i}$.
- Probabilities for view-tuples $\tau \in V_i$: $p^{V_i}(\tau) := \frac{w^{V_i}}{1+w^{V_i}}$
- New tuples' probabilities: For $\tau \in D^p$, $p^{tid}(\tau) := \frac{p(\tau)}{1+p(\tau)}$
  (from weights)

  E.g. $p^{tid}(\tau_3^1) = \frac{0.5}{1+0.5} = 1/3$
- DB tuples become independent

- We impose hard constraints: $V_1(b, t, \ldots)[0] \longleftarrow \ldots$ $V_2(b)[0] \longleftarrow \ldots$
- Violation-view weights: ∞ (highest penalty)
- Probabilities of view-tuples become: $p^{V_i}(\tau) = 1$
- We have a new PDB, now a TID, with D^p (with new probabilities), plus two views
- A query $\mathcal{Q}$ posed to the BID has to be rewritten to be posed to the TID (below)
- There are techniques for QA on TIDs

Back to QA

- Observed DB $D^\star$ is queried via the resulting TID D^{TID}
- QA done with ProvSQL (P. Senellart et al.)
- Rewriting original query to BID into two queries to TID
- Probability of a BCQ $\mathcal{Q}$:

$$P(\mathcal{Q}) \;:=\; P^{TID}(\mathcal{Q} \mid \neg U) = \frac{P^{TID}(\mathcal{Q} \wedge \neg U)}{P^{TID}(\neg U)} = \frac{P^{TID}(\mathcal{Q} \vee U) - P^{TID}(U)}{1 - P^{TID}(U)}$$

- U is essentially the lineage (provenance) of the views, that should be unsatisfiable

  In terms of the tuples in D^p that make them true

$$\begin{aligned} U \;&:=\; U_{C_1} \vee U_{C_2} \\ &:=\; \text{disjunction of conjunction of tuples in } D^p \text{ leading to tuples in } V_i \\ &:=\; \text{disjunction of conjunction of tuples violating constraints } C_i \end{aligned}$$

- In our example:

$$U_{C_1} = (\tau_3^1 \wedge \tau_3^2) \vee (\tau_3^1 \wedge \tau_3^3) \vee \cdots \vee (\tau_3^1 \wedge \tau_3^2 \wedge \tau_3^3) \vee \cdots \vee (\tau_7^1 \wedge \tau_7^2 \wedge \tau_7^3)$$

$$U_{C_1} = (\neg\tau_3^1 \wedge \neg\tau_3^2 \wedge \neg\tau_3^3) \vee \cdots \vee (\neg\tau_7^1 \wedge \neg\tau_7^2 \wedge \neg\tau_7^3) \vee \neg\tau_1 \vee \neg\tau_2 \vee \neg\tau_4 \vee \neg\tau_6 \vee \neg\tau_8$$

bid	tid	A^o	B^o	C^m	p^{tid}
β_1	τ_1	a	0	0	p_1^{tid}
β_2	τ_2	a	0	0	p_2^{tid}
β_3	τ_3^1	a	1	0	1/3
β_3	τ_3^2	a	1	1	p_{32}^{tid}
β_3	τ_3^3	a	1	2	p_{33}^{tid}
β_4	τ_4	a	1	0	p_4^{tid}
β_5	τ_5^1	a	1	0	p_{51}^{tid}
β_5	τ_5^2	a	1	1	p_{52}^{tid}
β_5	τ_5^3	a	1	2	p_{53}^{tid}
β_6	τ_6	a	1	1	p_6^{tid}
β_7	τ_7^1	a	1	0	p_{71}^{tid}
β_7	τ_7^2	a	1	1	p_{72}^{tid}
β_7	τ_7^3	a	1	2	p_{73}^{tid}
β_8	τ_8	a	1	2	p_8^{tid}

- Lineages of $\mathcal{Q}$ and $\mathcal{Q} \vee U$ are in DNF
- ProvSQL uses knowledge compilation on these lineages

  Eventually compiling them into d-DNNF-Boolean Circuits

  On them probabilities are computed bottom-up
- Tuples' probabilities in the right-most column are used assuming independence

Final Remarks

- Our approach to dealing with MV can be seen as a generalization of traditional imputation techniques

  Instead of a single "clean" instance, we appeal to a class of probability-weighted clean instances

  Making for a more principled and uncertainty-aware data semantics and QA

- Our approach can be understood as a form of collective imputation by means of probability distributions over possible values

  Distributions that are related to each other through an underlying joint distribution determined by a MG

  Those "cell-level" distributions can be seen as footprints of the joint distribution

- We have experimented with ProvSQL
  (w/ Zahra Mousavi & Mohamad Alipour)
- Lineages for MarkoViews may become very large; a limitation
- ProvSQL has a new "repair-key" functionality

  Used to repair "violations" of "Block-Id Keys"

  Keeping one tuple per block (no need to specify this)
- An alternative to MarkoViews for key-constraints

  Scales up better than MarkoView-approach, but still limited
- Knowledge Compilation techniques become crucial for QA
- MarkoViews method can be applied with more general ICs

  Imposing ICs on PDBs
- Comparisons with DB repairs! (w/ Felipe Azua & Nina Pardal)