Towards declarative comparabilities: application to functional dependencies.

BDA 2021

Lhouari Nourine [1], Jean-Marc Petit [2], and *Simon Vilmin* [1].

[1]LIMOS, CNRS, Université Clermont Auvergne, [2]LIRIS, INSA Lyon, Université Claude Bernard Lyon 1

October 2021

Introduction

r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100

A: triglyceride level (mmol/L) B: sex

C: waist size (cm)

"biased" functional dependencies

$$A \longrightarrow BC$$

$$C \longrightarrow AB$$

Cannot find "real" dependencies

$$BC \longrightarrow A$$

In short

- Deciding that $x = y$ is a tough problem:
  - depends on the context, types, units, ...
  - measuring similarity may not be expressive enough: what makes two values more or less similar?
  - equality impacts dependencies inference
- Implicit in topics such as:
  - Query answering [Libkin, 2016]
  - Inconsistent databases
- In fact:
  - only *domain experts* know about equality
  - but *programmers* have to implement it

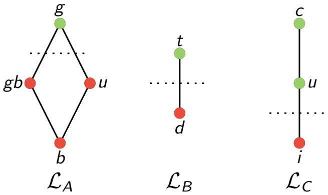
Highlights

- Declarative framework which:
  - extends the relation schema;
  - allows multiple definition of equality
- With a focus on:
  - functional dependencies,
  - prototypical implementation.

Framework in a nutshell

- Deciding equality is about:
  - *comparing* values
  - *interpreting* comparabilities with *true* or *false* ;

r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	null
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100



interpretations

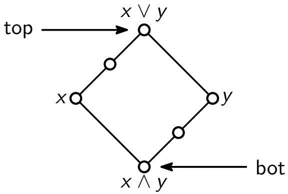
	$t_1 = t_4$	$BC \rightarrow A$
g_0	✗	✗
g_1	✓	✓
g_2	✗	✗

Some related works

- Approach of Fuzzy logic [Goguen, 1967]
- Truth values as lattices, e.g. Kleene's 3-valued logic [Libkin, 2016, Bolc, Borowik, 2013]
- Use of some (different) similarity functions [Caruccio et al., 2015, Baixeries et al, 2018, Bertossi et al., 2013]
- dealing with inconsistent data [Bertossi, 2011]

Appetizers: few notations

- See e.g. [Day, 1992], [Demetrovics et al., 1992]
- R a set of *attributes* (or relation schema),
- A *functional dependency* is an expression of the form $X \rightarrow Y$ where $X, Y \subseteq \mathrm{R}$
  - $Z \subseteq \mathrm{R}$ satisfies $X \rightarrow Y$ if $X \subseteq Z$ implies $Y \subseteq Z$.
  - if Z_1 and Z_2 satisfy $X \rightarrow Y$, so does $Z_1 \cap Z_2$ (closure system).
  - *closure* $Z^+ = \{A \in \mathrm{R} \mid Z \rightarrow A \text{ holds}\}$.
- *Lattice* $\mathcal{L}$:
  - partially ordered set
  - each pair $x, y \in \mathcal{L}$ has a *least upper bound* $x \vee y$ and a *greatest lower bound* $x \wedge y$

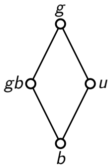


Starting point: attribute context

- A *truth lattice* $\mathcal{L}_A$ for an attribute A:
  - set of *abstract values* ordered as a lattice,
  - models a similarity-scale for pairs of attribute values;
- A *comparability function* f_A
  - maps pairs of attribute values to an abstract value,
  - subsumes equality (but for `null`), i.e. $f_A(x, x)$ equals the top of $\mathcal{L}_A$ if $x \neq \texttt{null}$.
- The pair $\{f_A, \mathcal{L}_A\}$ is the *attribute context*.
- Combining attribute contexts we obtain the *schema context* $\{f_{\mathrm{R}}, \mathcal{L}_{\mathrm{R}}\}$
  - $\mathcal{L}_{\mathrm{R}}$: collection of all possible *abstract tuples*, i.e, the product of abstract lattices, ordered component wise.
  - $f_{\mathrm{R}}(t_i, t_j)$ component-wise comparison of the tuples t_i, t_j,
  - $f_{\mathrm{R}}(r) = \{f_{\mathrm{R}}(t_i, t_j) \mid t_i, t_j \in r\}$ set of abstract tuples associated to r

Running example

$$f_A(x,y) = \begin{cases} \textit{good} & \text{if } x = y \text{ or } x, y \in [0,2[\\ \textit{good or bad} & \text{if } x, y \in [2,5[, x \neq y \\ \textit{unknown} & \text{if } (x,y) \text{ or } (y,x) \in [0,2[\times[2,5[\\ \textit{bad} & \text{otherwise.} \end{cases}$$

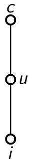


$\mathcal{L}_A$

$$f_B(x,y) = \begin{cases} \textit{true} & \text{if } x = y \\ \textit{different} & \text{otherwise.} \end{cases}$$

$\mathcal{L}_B$

$$f_C(x,y) = \begin{cases} \textit{correct } (c) & \text{if } x = y \neq \texttt{null} \text{ or } 70 \leq x, y \leq 80 \\ \textit{unknown } (u) & \text{if } x = \texttt{null} \text{ or } y = \texttt{null} \\ \textit{incorrect } (i) & \text{otherwise.} \end{cases}$$



$\mathcal{L}_C$

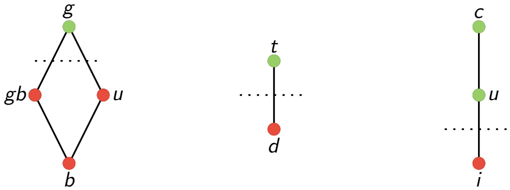
Interpretations

- An *attribute context interpretation* $h_A \colon \mathcal{L}_A \to \{0, 1\}$:
  - semantic for equality on A
  - *surjective*, differentiates equal and not equal: 1 to the greatest abstract value, 0 to the least one
  - *increasing*: a truth value cannot be considered as less equal than any of its predecessors
- The *schema interpretation* g:
  - point-wise evaluation of attributes interpretations,
  - maps each abstract tuple of $\mathcal{L}_{\mathrm{R}}$ to a binary word (equivalently a subset of R).

Running example

r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100

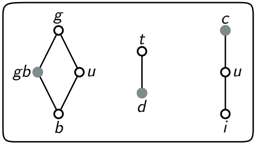
$g_2(f_R(\ldots))$	A	B	C
t_1, t_1	1	1	1
t_1, t_2	1	1	1
t_1, t_3	0	0	1
t_1, t_4	1	1	1
t_1, t_5	0	1	0



$\mathcal{L}_A$ $\mathcal{L}_B$ $\mathcal{L}_C$

Pipeline

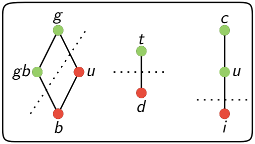
r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100



comparability

$f_R(r)$	A	B	C
t_1, t_1	g	t	c
t_1, t_2	g	t	u
t_1, t_3	u	d	c
t_1, t_4	u	t	c
…		…	

g_0



interpretation

$g_0(f_R(r))$	A	B	C
t_1	1	1	1
t_2	1	1	1
t_3	0	0	1
t_4	0	1	1
t_5	0	1	0

…

What about FDs ?

- Semantic of functional dependency $X \rightarrow Y$ smoothly adapted to this framework
- **Intuition**: when two tuples are "equal" on X, they must be on Y too
- Formally, for any tuples t_i and t_j, $X \subseteq g(f_R(t_i, t_j))$ implies $Y \subseteq g(f_R(t_i, t_j))$
- **Problem**:
  - all the knowledge depends on the choice of g, not uniquely on r
  - what if no semantic for equality is given ?
- **Idea**: *abstract tuples* define some *"abstract knowledge"* !

Abstract lattice, abstract FDs

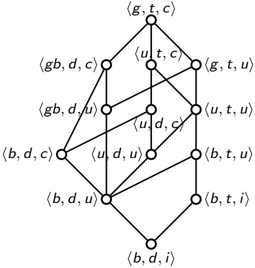
r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100

f_R →

	A	B	C
t_1, t_1	g	t	c
t_1, t_2	g	t	u
...	...	...	...
t_4, t_5	b	t	i

abstract tuples of $f_R(r)$

$\bigwedge f_R(r)$



abstract lattice $\mathcal{L}_r$

closure →

$\langle g, t, i \rangle \rightarrow \langle gb, d, c \rangle$

$\langle b, t, c \rangle \rightarrow \langle u, t, c \rangle$

$\langle gb, d, i \rangle \rightarrow \langle b, d, u \rangle$

...

abstract FDs

Abstract lattice, abstract FDs

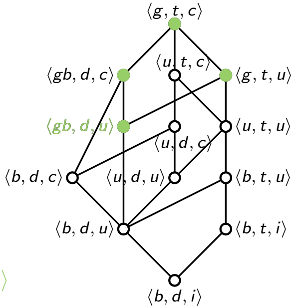
- Basically same intuition as classical functional dependencies (and agree sets [Beeri et al., 1984]) !
- *Abstract lattice* $\mathcal{L}_r$ associated to r:
  - start from $f_{\mathrm{R}}(r) = \{f_R(t_i, t_j) \mid t_i, t_j \in r\}$
  - close by the $\wedge$ operation of $\mathcal{L}_{\mathrm{R}}$.
- *Abstract functional dependency*:
  - expression $x \rightarrow y$ based on abstract tuples
  - *"Whenever the similarity of two tuples is above x, it is also above y"*
  - Abstract FDs are well-behaved (Armstrong axioms, fix-point closure, ...) [Day, 1992]
- Represent abstract knowledge associated to r:
  - r *satisfies* $x \rightarrow y$ if $x \leq f_R(t_i, t_j)$ implies $y \leq f_R(t_i, t_j)$ for any tuples t_i, t_j of r.
  - r satisfies $x \rightarrow y$ if and only if $\mathcal{L}_r$ satisfies $x \rightarrow y$.
  - no need of equality semantic.

Running example

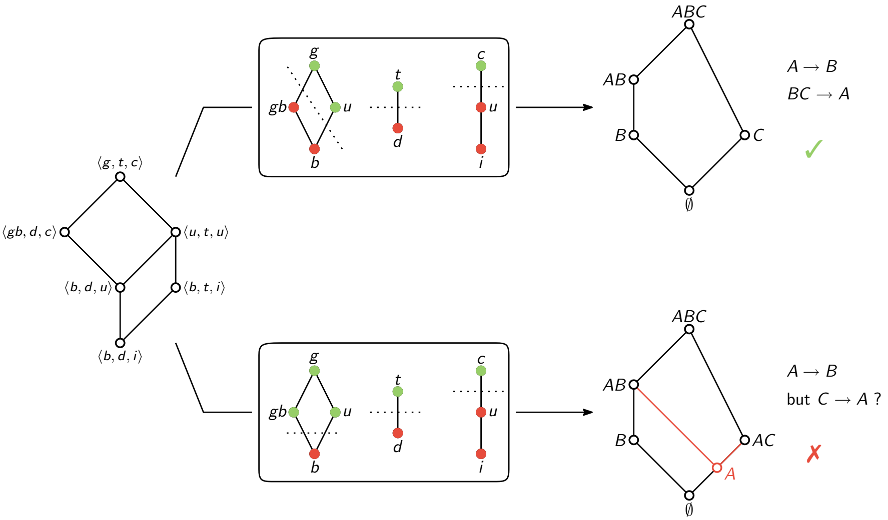
- *"A good or bad level of triglyceride should entail a correct waist size"*, modelled by $\langle gb, d, i\rangle \rightarrow \langle gb, d, c\rangle$.
- *fails* because of `null` value
- must be corrected to $\langle gb, d, i\rangle \rightarrow \langle gb, d, u\rangle$ to take `null` into account.

r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	`null`
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100

closure properties entails $\langle gb, d, i\rangle \rightarrow \langle gb, d, u\rangle$



Interpreting abstract knowledge



Realities

- **Problem**: when applied to the abstract knowledge of r, a semantic for equality lays the ground for functional dependencies ... *or NOT !!!*
- **Question**: what kind of interpretation guarantees that, the interpretation of any possible abstract knowledge (i.e. any possible abstract lattice) gives a sound semantic for classical FDs (i.e. a closure system) ?
- **Answer**: lattice *homomorphisms* !

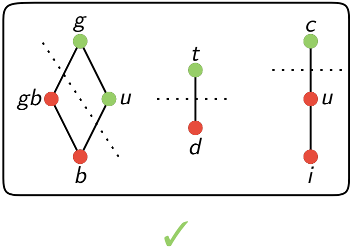
> **Theorem [Nourine et al. 2021]:** Let $\mathcal{C}_R$ be a schema context with at least 3 attributes, and let g be a schema interpretation. Then, $g(\mathcal{L})$ is a closure system for any $\wedge$-sublattice $\mathcal{L}$ if and only if g is a $\wedge$-homomorphism.

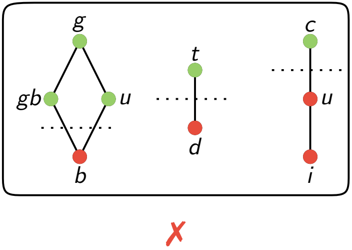
- We call such interpretations *realities*

Realities, quite simply

- Hints to understand realities:
  - *"true and true should be true"*
  - in each truth lattice, the family of abstract values set to 1 has a *unique minimal element*
  - A reality is represented by an *abstract tuple* !

reality represented by $\langle u, t, c \rangle$





Abstract FDs, FDs and realities

- **Thought**: a reality g interprets $\mathcal{L}_r$ in a suitable way for functional dependencies. Somehow, g "*realizes*" a part of the abstract knowledge of r

Proposition [Nourine et al. 2021]: If $X \rightarrow Y$ is a valid FD in $g(\mathcal{L}_r)$, for a given reality g, there exists an abstract FD $x \rightarrow y$ such that $g(x) = X$, $g(y) = Y$ and $x \rightarrow y$ is a valid abstract FD of $\mathcal{L}_r$.

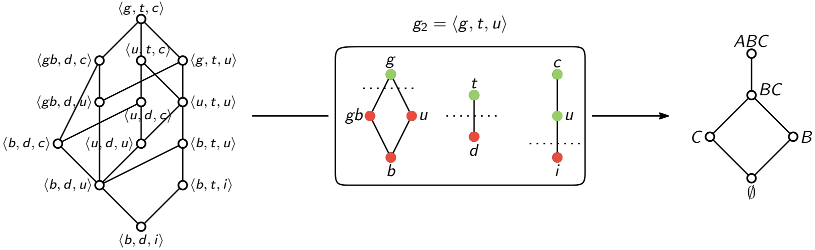
Proposition [Nourine et al. 2021]: If $x \rightarrow y$ is a valid abstract FD in $\mathcal{L}_r$, there exists a reality g such that $g(x) \rightarrow g(y)$ is a valid functional dependency of $g(\mathcal{L}_r)$.

Possible, Certain FDs

- Numerous possible meanings of equality
- Sometimes, an FD $X \to Y$ may hold, sometimes not ...
- Recover knowledge (FDs) from biased data
- Thinking about *query answering* [Libkin, 2016] leads to the natural questions:
  - *Possible FD*: is there a reality in which $X \to Y$ holds ?
  - *Certain FD*: is it true that $X \to Y$ holds in any reality ?

Running example

	$AB \rightarrow A$	$C \rightarrow AB$	$BC \rightarrow A$
g_0	✓	✗	✗
g_1	✓	✗	✓
g_2	✓	✗	✗



Results in brief

Problem - Possible Functional Dependency (PFD)

- **Input:** a relation r over a schema context $\mathcal{C}_\mathrm{R}$ (given), a FD $X \rightarrow Y$.
- **Output:** *Yes* if there exists a reality g in which $X \rightarrow Y$ is valid, *No* otherwise.

Problem - Certain Functional Dependency (CFD)

- **Input:** a relation r over a schema context $\mathcal{C}_\mathrm{R}$ (given), a FD $X \rightarrow Y$.
- **Output:** *Yes* if $X \rightarrow Y$ holds in every reality, *No* otherwise.

Theorem [Nourine et al. 2021]: PFD and CFD can be solved in polynomial time.

Experiment Time !!!

with Martin Benito-Rodriguez and Gabriel Eychene

Aim

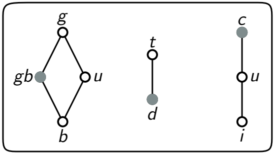
- Declarative approach for schema contexts (DDL)
- Use realities in a query of the form:

```
SELECT * FROM r
WHERE r.A = 1.4 AND r.B = 'F' AND r.c = 74
USING REALITY gb, t, u;
```

Answering the query

```
SELECT * FROM r
WHERE r.A = 1.4 AND r.B = 'F' AND r.c = 74
USING REALITY gb, t, u;
```

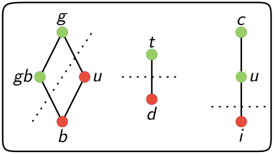
r	A	B	C
t_1	1.4	F	73
t_2	1.5	F	null
t_3	3.2	M	72
t_4	3.5	F	76
t_5	40	F	100



$f_R(t_i, t)$	A	B	C
t_1	g	t	c
t_2	g	t	u
t_3	u	d	c
t_4	u	t	c
t_5	b	t	i

$t = \langle 1.4, F, 74 \rangle$

$g' = \langle gb, t, u \rangle$



$g'(f_R(t_i, t))$	A	B	C
t_1	1	1	1
t_2	1	1	1
t_3	0	0	1
t_4	0	1	1
t_5	0	1	0

$t = \langle 1.4, F, 74 \rangle$

	A	B	C
t_1	1.4	F	73
t_2	1.5	F	null

Answer

Prototype

- Implementation using SQL and PLSQL on PostGreSQL.
- Comparabilities and interpretations as functions

```
CREATE OR REPLACE FUNCTION f_B(x char, y char)
RETURNS bit AS $$ <code> $$ LANGUAGE sql;

CREATE OR REPLACE FUNCTION h_B(b bit)
RETURNS boolean AS $$ <code> $$ LANGUAGE sql;
```

- Experiment on SQLiteOnline with :
  - Our toy example
  - IRIS Dataset: without null, 4-valued abstract lattices, comparabilities based on the difference between two values

Results

```
SELECT *
FROM <table>
WHERE
```

	Classic SQL		Framework	
	# tuples	run. time	# tuples	run. time
Toy example	/5		/5	
`level = 5`	0	0.035ms	0	0.045ms
`level = 1.4`	1	0.055ms	2	0.075ms
`r.level = s.level`	1	0.180ms	9	0.350ms
IRIS dataset	/150		/150	
`sepal_l = 0`	0	0.065ms	0	0.150ms
`sepal_l = 5`	10	0.060ms	83	0.170ms
`r.sepal_l = s.sepal_l`	900	0.700ms	12820	25.000ms

Conclusion

- Problem:
  - Deciding equality is a hard task
  - left to programmer, meant to domain experts.
- Introduction of a framework:
  - based on *comparabilities* and *interpretations* (and *realities*)
  - provide numerous semantics for equality, easy to declare.
- Highlights on:
  - abstract functional dependencies, no need of hypothesis on equality
  - connections between realities and (abstract/possible/certain) FDs
  - Prototypical implementation
- Further research:
  - Relational algebra ? Covers of functional dependencies ?
  - Implementation and experiments with real data ?

Thank you for your attention !

References

- J. Baixeries, V. Codocedo, M. Kaytoue, A. Napoli
  Characterizing approximate-matching dependencies in formal concept analysis with pattern structures
  Discrete Applied Mathematics, 249 :18-27, 2018.
- C. Beeri, M. Dowd, R. Fagin, R. Statman.
  On the structure of Armstrong relations for functional dependencies.
  Journal of the ACM, 31 :30-46, 1984.
- L. Bertossi,
  Database repairing and consistent query answering
  Morgan & Claypool Publishers, 3 :1-121, 2011.
- L. Bertossi, S. Kolahi, L. Laksmhmanan.
  Data cleaning and query answering with matching dependencies and matching functions.
  Theory of Computing Systems, 52 :441-482, 2013.
- L. Bolc, P. Borowik
  Many-valued logics 1: theoretical foundations.
  Springer Science & Business Media, 2013.
- L. Caruccio, V. Deufemia, G. Polese
  Relaxed functional dependencies—a survey of approaches
  IEEE Transactions on knowledge and data engineering, 28 :147-165, 2015.

References

- A. Day
  The Lattice Theory of Functional Dependencies and Normal Decompositions.
  Int. J. Algebra Comput., 2 :409-432, 1992.

- J. Demetrovics, L. Libkin, I. Muchnik
  Functional dependencies in relational databases: A lattice point of view
  Discrete Applied Mathematics, 40 :155-185, 1992.

- L. Libkin
  SQL's three-valued logic and certain answers.
  Algebra Universalis, 24 :60-73, 1987.

- J. Goguen
  L-fuzzy sets.
  Journal of mathematical analysis and applications, 18 :145-174, 1967.

- L. Nourine, J.-M Petit, S. Vilmin
  Towards declarative comparabilities: application to functional dependencies.
  arXiv preprint arXiv:1909.12656, 2021.