Datalog± Multidimensional Ontologies and Data Quality

Leopoldo Bertossi
Carleton University
Ottawa, Canada

In honor of Georg Gottlob on his Celebration Day, Genova, 2016

Motivation: Contexts and Data Quality

A table containing data about the temperatures of patients at a hospital

TempNoon

Patient	Value	Time	Date
Tom Waits	38.5	11:45	Sep/5
Tom Waits	38.2	12:10	Sep/5
Tom Waits	38.1	11:50	Sep/6
Tom Waits	38.0	12:15	Sep/6
Tom Waits	37.9	12:15	Sep/7

Are these quality data?

If not, anything to clean?

What?

We do not know ... It depends ...

Actually, the table is supposed/expected to contain:

"Tom's temperatures taken at noon by a certified nurse with oral thermometer"

Are these quality data? We still do not know ...

We do not have any elements to judge ...

Questions about quality of data make sense in a broader setting

The quality of the data depends on “the context”

A context supporting:

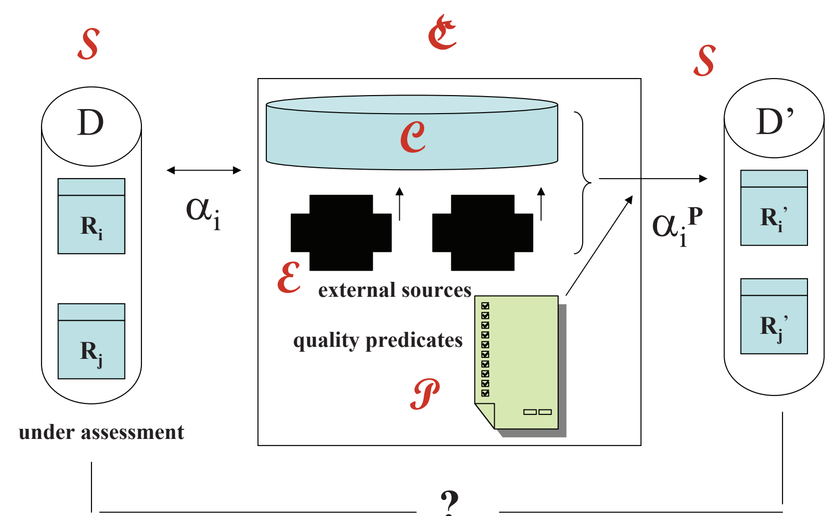
- Making sense of the data
- Providing additional semantics

  In particular, quality related semantics, e.g. quality constraints, rules?
- Providing relationships (mappings) to other, external, additional data
- Assessment of data (quality)
- Data cleaning
- “Clean query answering” with dirty data

Our approach: quality of data can be assessed with contextual knowledge about the production and/or use of data

Our first approach:

(LB et al., VLDB'10 BIRTE WS)

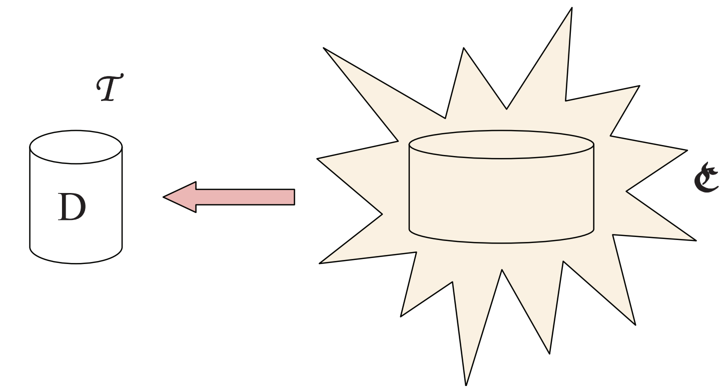


Context $\mathfrak{C}$:

- A relational database, a relational schema, a virtual integration system, a knowledge base, an ontology, ...
- Including quality predicates, data quality rules and constraints, quality criteria and guidelines ...

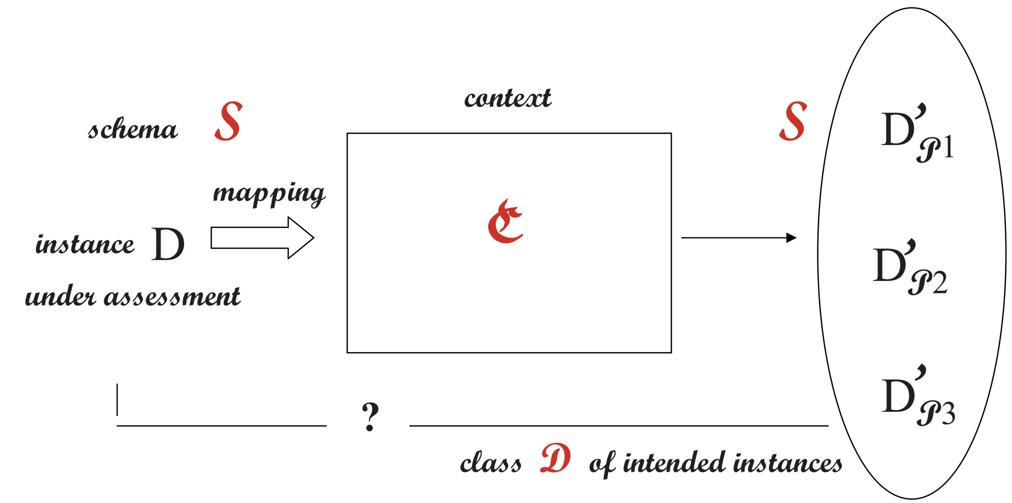
- Instance D under quality assessment (seen) as a footprint of contextual data

D as a footprint of a (broader) contextual instance



Only at $\mathfrak{C}$'s level can D's data be analyzed, assessed, cleaned, ...

- D can be mapped into the context



Quality criteria imposed at contextual level (as above)

Through the context, alternative clean versions of D can be specified, computed, compared (with each other and D), queried, ...

Depending on the mapping and context's ingredients

- D's quality measured by its distance to class $\mathcal{D}$ of its quality versions

  Collection of quality instances reflects "uncertainty" in dirty D

  Ground opened for "quality QA" (certain answers wrt. $\mathcal{D}$)

Multidimensional Contexts and Data Quality

Doctor requires temperatures taken with oral thermometer, and expects data corresponds to requirement

Table has no elements for this assessment

An external context can provide them

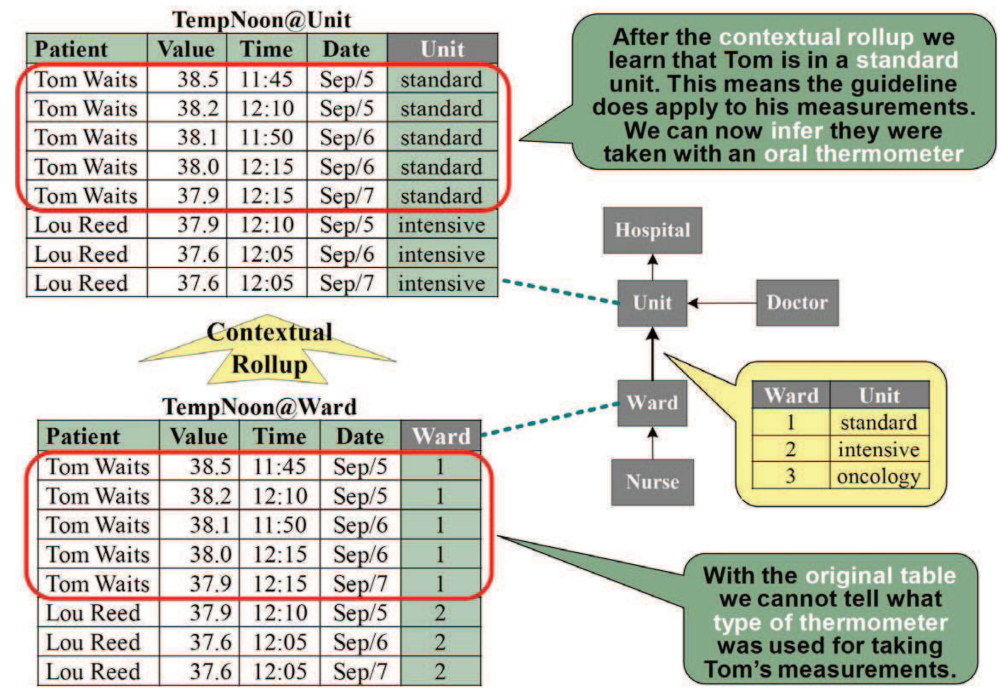
TempNoon@Ward

Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

The context could be a (multi-)dimensional database, or a dimensional ontology

Actually, we may have been missing "dimensions" above, something intrinsically "contextual"

A MD data model/instance



TempNoon@Unit

Patient	Value	Time	Date	Unit
Tom Waits	38.5	11:45	Sep/5	standard
Tom Waits	38.2	12:10	Sep/5	standard
Tom Waits	38.1	11:50	Sep/6	standard
Tom Waits	38.0	12:15	Sep/6	standard
Tom Waits	37.9	12:15	Sep/7	standard
Lou Reed	37.9	12:10	Sep/5	intensive
Lou Reed	37.6	12:05	Sep/6	intensive
Lou Reed	37.6	12:05	Sep/7	intensive

Ward	Unit
1	standard
2	intensive
3	oncology

TempNoon@Ward

Patient	Value	Time	Date	Ward
Tom Waits	38.5	11:45	Sep/5	1
Tom Waits	38.2	12:10	Sep/5	1
Tom Waits	38.1	11:50	Sep/6	1
Tom Waits	38.0	12:15	Sep/6	1
Tom Waits	37.9	12:15	Sep/7	1
Lou Reed	37.9	12:10	Sep/5	2
Lou Reed	37.6	12:05	Sep/6	2
Lou Reed	37.6	12:05	Sep/7	2

A hospital guideline

As a rule or a constraint

"Take patients' temperatures in standard care units with oral thermometers"

Can be taken advantage of through/after upward navigation in the hierarchical, dimensional hospital structure

- Contextual information has a multi-dimensional nature

  Other dimensions could be easily considered, e.g. time

- Enabling contextual, dimensional navigation, rolling-up/drilling-down

  To access and generate missing data at certain levels (as in example above)

- Idea: Embed Hurtado-Mendelzon (HM) MD data model in contexts

  A multidimensional context is generated for dimensional and finer-granularity data quality assessment

- Go beyond: Enrich it with additional, dimension-related data, rules and constraints

  An ontological, multidimensional context!

Ontological Contexts with Dimensions

New ingredients in MD contexts:[1] (AMW'12, RuleML'15)

- A (relational reconstruction of) the HM model
- Categorical relations: Generalize fact tables

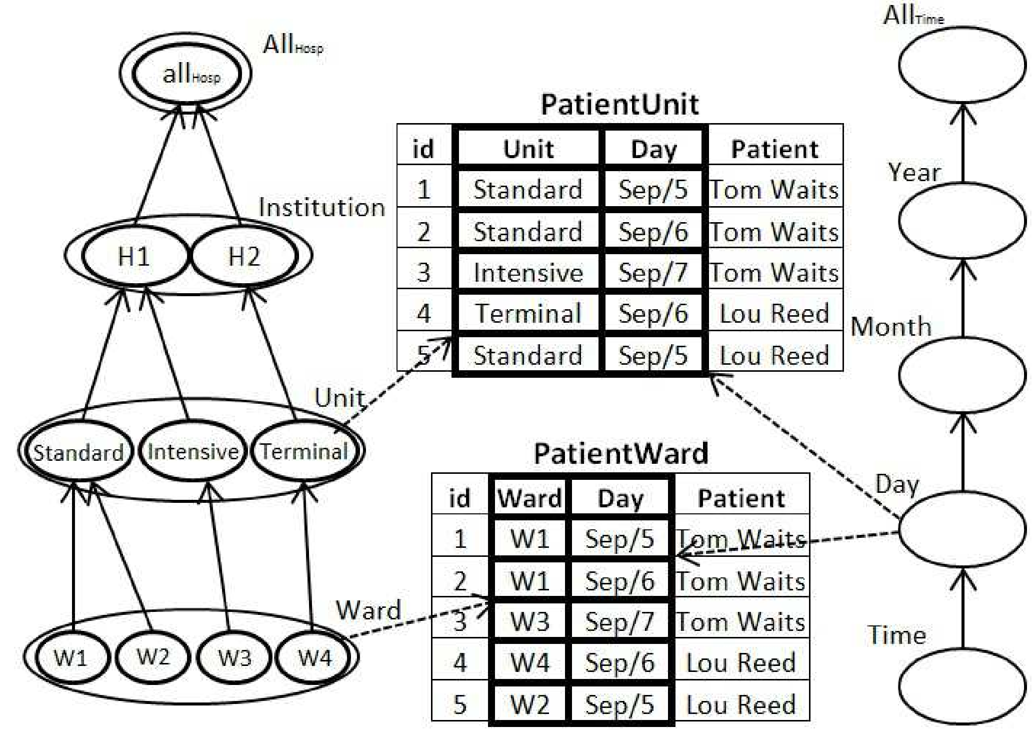
  Not necessarily numerical values, linked to different levels of dimensions, possibly incomplete
- Dimensional rules: generate data where missing, enable navigation
- Dimensional constraints: on (combinations of) categorical relations, involving values from dimension categories

[1] Join work with Mostafa Milani

- *Categories* *Ward* and *Unit* in *Hospital* dimension

- *UnitWard(unit,ward)*: parent/child relation

- *PatientWard*: categorical relation
  Ward and *Day* categorical attributes
  takes values from categories



PatientUnit

id	Unit	Day	Patient
1	Standard	Sep/5	Tom Waits
2	Standard	Sep/6	Tom Waits
3	Intensive	Sep/7	Tom Waits
4	Terminal	Sep/6	Lou Reed
5	Standard	Sep/5	Lou Reed

PatientWard

id	Ward	Day	Patient
1	W1	Sep/5	Tom Waits
2	W1	Sep/6	Tom Waits
3	W3	Sep/7	Tom Waits
4	W4	Sep/6	Lou Reed
5	W2	Sep/5	Lou Reed

- Categorical relations are subject to dimensional constraints

- Need rules for dimensional navigation

What language to express all this? Datalog$\pm$, of course!

(Gottlob et al., ∞)

Datalog$\pm$ MD Ontologies

Dimensional Constraints:

- A referential constraint restricting units in *PatientUnit* to elements in the *Unit* category, as a negative constraint

$$\perp \leftarrow PatientUnit(\boldsymbol{u}, \boldsymbol{d}; p), \neg Unit(u)$$

- *"All thermometers used in a unit are of the same type"* :

$$t = t' \leftarrow Thermometer(\boldsymbol{w}, \boldsymbol{t}; n), Thermometer(\boldsymbol{w'}, \boldsymbol{t'}; n'), UnitWard(u, w), UnitWard(u, w')$$

An EGD

$Thermometer(ward, thermometertype; nurse)$ is categorical relation, t, t' for categorical attributes

- *"No patient in intensive care unit on August /2005"*:

$$\bot \leftarrow PatientWard(\boldsymbol{w}, \boldsymbol{d}; p), UnitWard(\texttt{Intensive}, w),$$
$$MonthDay(\texttt{August}/2005, d)$$

Dimensional Rules:

- Data in *PatientWard* generate data about patients for higher-level categorical relation *PatientUnit*

$$PatientUnit(\boldsymbol{u}, \boldsymbol{d}; p) \leftarrow PatientWard(\boldsymbol{w}, \boldsymbol{d}; p), \ UnitWard(u, w)$$

  To navigate from *PatientWard.Ward* up to *PatientUnit.Unit* via *UnitWard*

  Once at the level of *Unit*, take advantage of guideline (a rule):

  "Temperatures of patients in a standard care unit are taken with oral thermometers"

  Data at *Unit* level that can be used there and at *Ward* level

- Data in categorical relation *WorkingSchedules* generate data in categorical relation *Shifts*

WorkingSchedules

Unit	**Day**	**Nurse**	**Type**
Intensive	Sep/5	Cathy	cert.
Standard	Sep/5	Helen	cert.
Standard	Sep/6	Helen	cert.
Standard	Sep/9	Mark	non-c.

$\searrow$

Shifts

Ward	**Day**	**Nurse**	**Shift**
W4	Sep/5	Cathy	night
W1	Sep/6	Helen	morning
W4	Sep/5	Susan	evening

$$\exists z \; Shifts(\boldsymbol{w}, \boldsymbol{d}; n, z) \leftarrow WorkingSchedules(\boldsymbol{u}, \boldsymbol{d}; n, t), \; UnitWard(u, w)$$

Captures guideline: *"If a nurse works in a unit on a specific day, she has shifts in every ward of that unit on the same day"*

Existential variable z for missing values for the non-categorical *shift* attribute

Rule for downward- navigation and value invention, with join via categorical attribute between categorical and parent-child predicate

Properties of MD Ontologies

- With reasonable and natural conditions, Datalog$\pm$ MD ontologies become weakly-sticky Datalog$\pm$ programs [Cali et al., AIJ'12]

  Important that join variables in TGDs are for categorical attributes (with values among finitely many category members)

- The chase (that enforces TGDs) may not terminate

  Weak-Stickiness guarantees tractability of conjunctive QA: only a "small", initial portion of the chase has to be queried

  Boolean conjunctive QA is tractable for weakly-sticky (WS) Datalog$\pm$ ontologies

- Separability condition on the (good) interaction between TGDs and EGDs becomes application dependent

  If EGDs have categorical head variables (as in page 12), separability holds

  Separability guarantees decidability of conjunctive QA, etc.

We wanted:

(a) A practical QA algorithm for WS Datalog$\pm$

(b) The possibility of optimizing the algorithm

Query Answering on WS MD-Ontologies

- There is a non-deterministic PTIME algorithm for WS Datalog$\pm$

  (Cali et al., AIJ'12)

- Our goal was to develop a practical chase-based QA algorithm

- Apply magic-sets techniques to optimize QA

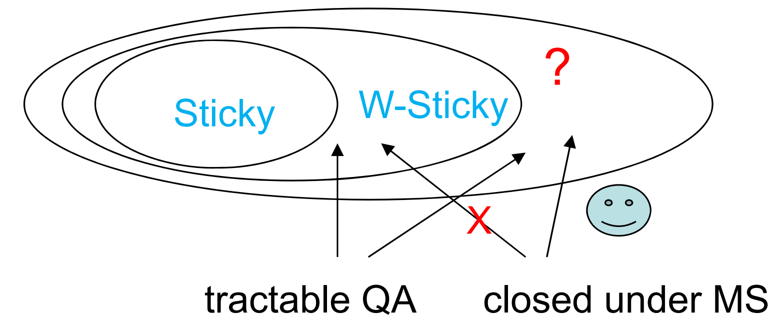
  There is such a technique (MS) available for (a class of) "existential programs" ($\exists$-Datalog) (Alviano et al., Datalog 2.0'12)

- WS Datalog$\pm$ is not closed under MS

Questions:

- A class extending WS Datalog$\pm$, closed under MS, with tractable QA?
- For which a PTIME QA algorithm can be developed?

Going Not-Too-Far Beyond WS MD-Ontologies



- WS Datalog$\pm$ is a syntactic class

  defined by a combination of:

  - The notion of finite-rank position (predicate/attribute) found in weakly-acyclic TGDs (Π_F in data exchange)

  - A variable-marking procedure developed for sticky Datalog$\pm$, to keep track of value propagation via joins

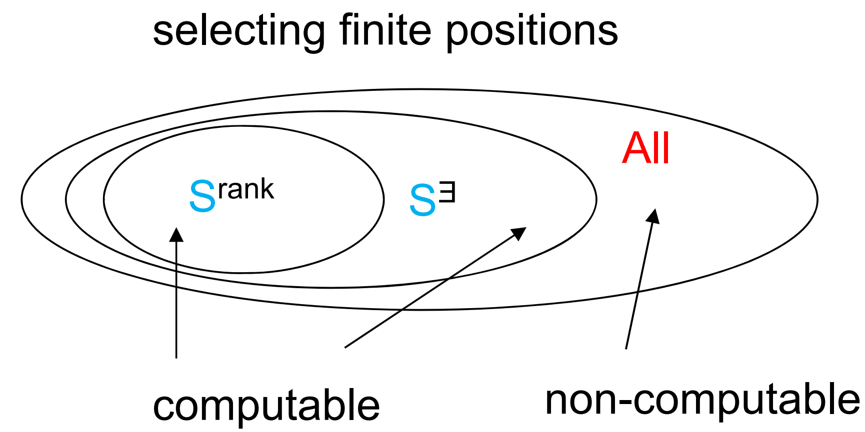
    (A better-behaved, less expressive subclass of WS Datalog$\pm$)

  It captures "finite positions": finitely many nulls in them during the chase

  A "selection function", S^{rank}, of finite positions via finite-rank positions

- But not necessarily *all* finite positions (doing so is undecidable)

We started investigating more general selections functions (AMW'15, RR'16)



- Determining a new, syntactic, computable selection function: $S^{rank} \subseteq S^{\exists}$

  It uses:

  - the *existential-dependency graph* (Krötzsch & Rudolph, IJCAI'11)
  - a marking procedure via join variables in TGDs (neglected by S^{rank})

- We identified and characterized via $S^{\exists}$ the *Joint-Weakly-Sticky* (JWS) class

  A syntactic class with tractable QA that extends WS Datalog$\pm$ and is closed under MS!

Joint-Weak-Stickiness

Set of TGDs Σ:

$$p(\hat{X}, \hat{Y}), u(\hat{Y}) \ \rightarrow \ \exists Z\ p(Y, Z)$$

$$u(X), p(X, \hat{Y}), p(\hat{Y}, \hat{W}) \ \rightarrow \ t(X)$$

Marks body variables that either:

(a) do not appear in heads, e.g. X in the first rule, and Y in the second, or

(b) occur in heads only in positions of marked variables (maybe another rule), e.g. Y in first rule (Y occurs in $p[1]$ in the head, where marked variable Y appears in the body of second rule)

- $\mathcal{S}^{rank}(\Sigma) = \Pi_F(\Sigma) = \{u[1]\}$
- With marked variables as for WS programs
- Σ is WS if marked join variables appear in at least one "finite position"
- Join variable Y appears in $p[1], p[2] \notin \mathcal{S}^{rank}(\Sigma)$ Σ is not WS!
- Σ is JWS: $\mathcal{S}^{\exists}(\Sigma) = \{p[1], p[2], u[1], t[1]\}$

- We proposed a PTIME chase-based QA algorithm for JWS Datalog$\pm$

  For QA a finite initial fragment of the chase is good enough

- The (generic) algorithm takes into account during the chase if a position is finite or not

  As determined by the selection function (which acts as an oracle)

  And behaves accordingly

- As such it can be applied both to WS and JWS, but some finite positions will be missed when applied to WS

QA Algorithm

- Σ :

$$p(\hat{X}, Y) \ \rightarrow \ \exists Z \; p(Y, Z)$$

$$u(\hat{X}), p(\hat{X}, Y), p(Y, \hat{W}) \ \rightarrow \ t(Y)$$

- Σ is JWS: X appears in $\mathcal{S}^{\exists}(\Sigma) = \{u[1]\}$

- Algorithm with $D = \{p(a, b), u(b)\}$ and $\mathcal{Q}\!: \ \exists Y \; t(Y)$

  - Initialize $I := D$, and apply first TGD, creating $p(b, \zeta_1)$
  - First TGD cannot be applied again: $p(\zeta_1, \zeta_2)$ homomorphic to $p(b, \zeta_1)$
  - No applicable rules
  - Resume with frozen ζ_1 (as a constant, relevant for homo tests)
  - As many resumptions as existentials in query (one here)

$D = \{p(a,b), u(b)\}$
$\mathcal{Q}\colon\ \exists Y\ t(Y)$

$$p(\hat{X}, Y) \ \rightarrow\ \exists Z\ p(Y, Z)$$
$$u(\hat{X}), p(\hat{X}, Y), p(Y, \hat{W}) \ \rightarrow\ t(Y)$$

- Algorithm continues
- Apply first and second TGDs, creating $p(\zeta_1, \zeta_2)$ and $t(\zeta_1)$, resp.
- No applicable rules (due to homo test), no more resumptions
- The algorithm stops with instance $I = D \cup \{p(b,\zeta_1), p(\zeta_1,\zeta_2), t(\zeta_1)\}$
- $I \models \mathcal{Q}$, so answer is *true* in $\Sigma \cup D$

Algorithm stops, producing a query-dependant, initial, finite portion of the regular chase, and is good enough to answer the query

Conclusions

- Datalog$^{\pm}$ is an expressive and computationally nice family of existential programs (with constraints)
- Interesting applications in data modeling and the semantic web
- We have used Datalog$^{\pm}$ to create multidimensional ontologies

  They can be seen as logic-based extensions of multidimensional DBs

  They were motivated by data quality concerns

  They are interesting by themselves
- They belong to well-behaved classes of Datalog$^{\pm}$
- We proposed chase-based QA algorithms for (extensions of) WS Datalog$^{\pm}$
- We applied magic-sets techniques
- QA can be used to extract quality data from dirty data (RuleML'15)

Open problems in our setting:

- Sometimes we have to deal with closed predicates, e.g. categories
- Inconsistency tolerance What if constraints are not satisfied?

- Implementation of the QA algorithm and experiments

EXTRA SLIDES

Downward Navigation and Categorical Attributes

TGDs as in page 14 can be used for "deterministic" downward navigation: only values for non-categorical attributes are created, with determinism wrt. the categories involved

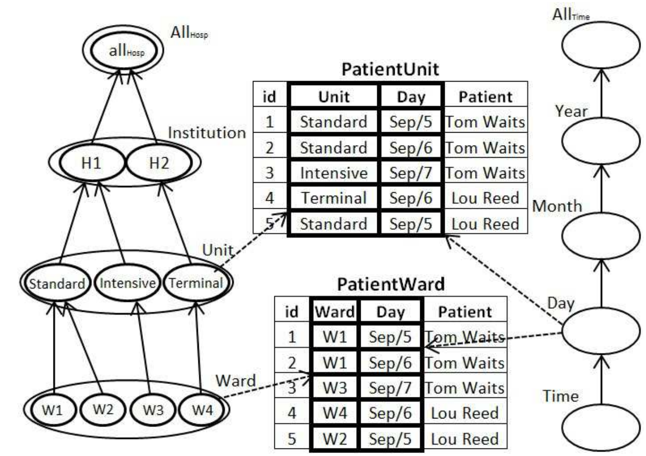
In some applications there may be incomplete data about the categorical attributes

Existential quantifications over categorical variables may be needed

Categorical relation *DischargePatients*, linked to *Institution*, with data about patients leaving the hospital

DischargePatients

Inst.	Day	Patient
H1	Sep/9	Tom Waits
H1	Sep/6	Lou Reed
H2	Oct/5	Elvis Costello



Query on *PatientUnit* about the dates that ‘Elvis Costello’ was in a unit at institution ‘H2’

No answer directly from *PatientUnit* (as derived from *PatientWard*)

If each patient is in a (only one) unit, *DischargePatient* can generate data downwards for *PatientUnit*

Knowledge about the unit (a category value) at the lower level is uncertain:

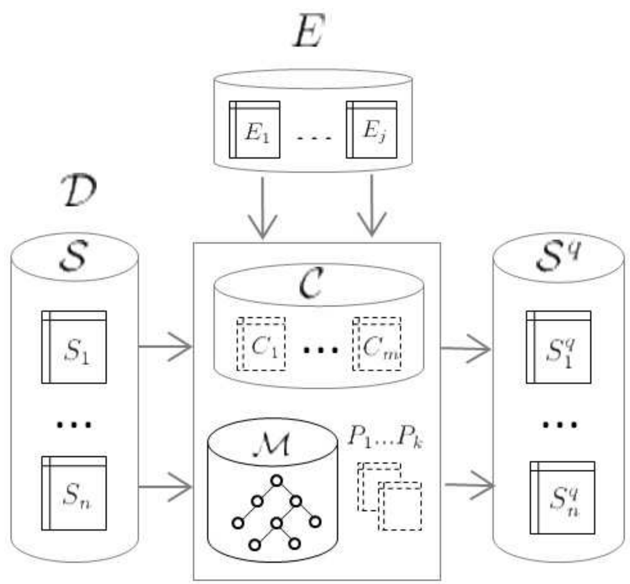
$$\exists u\ InstitutionUnit(i, u), PatientUnit(\boldsymbol{u}, \boldsymbol{d}; p) \ \leftarrow \\ DischargePatients(\boldsymbol{i}, \boldsymbol{d}; p)$$

With rules of this kind, an MD ontology is still weakly-sticky: no infinite loops, only a limited number of new nulls can be generated with the chase

EGDs with only categorical attributes in heads do not guarantee separability anymore, and becomes application dependent

MD Contexts and Quality Query Answering: The Gist

The Datalog$\pm$ MD ontology $\mathcal{M}$ becomes part of the context for data quality assessment



The original instance $\mathcal{D}$ is to be assessed or cleaned through the context

By mapping $\mathcal{D}$ into the contextual schema/instance $\mathcal{C}$

In the context:

- Contextual predicates C_i
- Predicates P_i specifying single quality requirements
- $\mathcal{S}^q$ copy of schema $\mathcal{S}$: S_i^q clean version of original S_i, specified using $\mathcal{C}, \mathcal{P}$ and $\mathcal{M}$

We want quality answers to the query about Tom's temperatures:

$$\mathcal{Q}(t,p,v) \leftarrow Measurements(t,p,v), p = \texttt{Tom Waits}, \\ \texttt{Sep/5-11:45} \leq t \leq \texttt{Sep/5-12:15}.$$

Quality requirements are not captured by this query; we expect:

"Body temperatures of *Tom Waits* for *September 5* around noon taken by a *certified nurse* with a thermometer of *brand B1*"

Table *Measurements* does not contain information about nurses or thermometers

Contextual data must be taken into account, such as categorical relation *PatientUnit* and the guideline

"Temperature measurement for patients in a standard care unit are taken with thermometers of brand B1"

According to the general contextual approach DQA, table (or better predicate) *Measurement* has to be logically connected to the context

As a “footprint” of a “broader” contextual table that is given or built in the context, in this case, one with information about thermometer brands (b) and nurses’ certification status (y):

$$\begin{aligned} Measurement'(t,p,v,y,b) \quad \leftarrow \quad & Measurement^c(t,p,v), \\ & TakenByNurse(t,p,n,y), \\ & TakenWithTherm(t,p,b) \end{aligned}$$

$Measurement^c$ is contextual version of *Measurement* (e.g. the latter mapped into the context)

If we want quality measurements data, we impose the required conditions:

$$\begin{aligned} \mathit{Measurement}^q(t,p,v) \leftarrow \quad & \mathit{Measurement}'(t,p,v,y,b), \\ & y = \mathtt{Certified},\ b = \mathtt{B1} \end{aligned}$$

The auxiliary predicates above:

$$\begin{aligned} \mathit{TakenByNurse}(t,p,n,y) \quad \leftarrow \quad & \mathit{WorkingSchedules}(u,d;n,y), \\ & \mathit{DayTime}(d,t),\ \mathit{PatientUnit}(u,d;p) \\ \mathit{TakenWithTherm}(t,p,b) \quad \leftarrow \quad & \mathit{PatientUnit}(u,d;p), \\ & \mathit{DayTime}(d,t), b = \mathtt{B1}, u = \mathtt{Standard} \end{aligned}$$

(*DayTime* is parent/child relation in *Time* dimension)

The second definition is capturing the guideline above

To obtain quality answers to the original query, we pose to the ontology the new query:

$$\mathcal{Q}^q(t, p, v) \ \leftarrow Measurements(t, p, v)^q, \ p = \texttt{Tom Waits},$$
$$\texttt{Sep}/\texttt{5-11:45} \leq t \leq \texttt{Sep}/\texttt{5-12:15}.$$

Answering it triggers dimensional navigation, when requesting data for categorical relations *PatientUnit* and *WorkingSchedules*

Magic-Sets Rewriting

- We consider a magic-sets rewriting method (MS) for Datalog$^\exists$

  [Alviano et al., Datalog 2.0'12]

  Quite general, and does not bound existential variables

  Nothing like this: $\exists w\ Assist^{f\,b}(v, w) \leftarrow\ Assist^{ff}(u, v)$

- WS not closed under MS, but JWS is

- $AL(\mathcal{S}^{ext})$ can be applied both to a JWS program and its MS rewriting

  Whereas $AL(\mathcal{S}^{rank})$ applied to a WS program's MS rewriting (possibly no longer WS) will be sound, but possibly incomplete

Example: Σ below is WS

$$\begin{aligned}
\sigma_1: &\quad \exists z\ Assist(z,x) \leftarrow Assist(x,y)\\
\sigma_2: &\quad \exists w\ Assist(v,w) \leftarrow Assist(u,v)\\
\sigma_3: &\quad Certified(x') \leftarrow Assist(x',y'), Assist(y',z'), Doctor(y')
\end{aligned}$$

Query $\mathcal{Q}$: $Certified(\texttt{Marie})$?

Adorned program Σ^a:

$$\begin{aligned}
r_1: &\quad \exists z\ Assist^{fb}(z,x) \leftarrow Assist^{bf}(x,y)\\
r_2: &\quad \exists w\ Assist^{bf}(v,w) \leftarrow Assist^{fb}(u,v)\\
r_3: &\quad Certified^{b}(x') \leftarrow Assist^{bf}(x',y'), Assist^{bf}(y',z'), Doctor(y')
\end{aligned}$$

Still WS

The MS rewriting Σ^M:

$$
\begin{aligned}
m_1: &\quad \exists z\ Assist^{fb}(z,x) \leftarrow mg\_Assist^{fb}(x), Assist^{bf}(x,y)\\
m_2: &\quad \exists w\ Assist^{bf}(v,w) \leftarrow mg\_Assist^{bf}(v), Assist^{fb}(u,v)\\
m_3: &\quad Certified^{b}(x') \leftarrow mg\_Certified^{b}(x'), Assist^{bf}(x',y'),\\
&\qquad Assist^{bf}(y',z'), Doctor(y')
\end{aligned}
$$

And the magic rules:

$$
\begin{aligned}
m_4: &\quad mg\_Certified^{b}(\texttt{Marie}).\\
m_5: &\quad mg\_Assist^{bf}(x') \leftarrow mg\_Certified^{b}(x')\\
m_6: &\quad mg\_Assist^{bf}(y') \leftarrow mg\_Certified^{b}(x'), Assist^{bf}(x',y')\\
m_7: &\quad mg\_Assist^{fb}(v) \leftarrow mg\_Assist^{bf}(v)\\
m_8: &\quad mg\_Assist^{bf}(x) \leftarrow mg\_Assist^{fb}(x)
\end{aligned}
$$

Σ^M is not WS!

Σ is JWS since it is WS

Σ^M is also JWS

Example: (EDG and Joint Acyclicity)

Assume a set Σ of tgds (a variable only appears in one rule):

$$\sigma_1 : \quad \exists z\ Assist(x, z) \leftarrow Nurse(x, y), Doctor(x)$$
$$\sigma_2 : \quad \exists w\ Nurse(w, u) \leftarrow Assist(t, u)$$

Π_x^B and Π_x^H are the set of all positions where a variable x occurs in the body and head of a rule

I.e. $\Pi_x^B = \{Nurse[1], Doctor[1]\}$ and $\Pi_x^H = \{Assist[1]\}$

For any $\exists$-variable x, Ω_x is the set of positions in which values invented for x may appear

Ω_x can be computed as the smallest set that:

(1) $\Pi_x^H \subseteq \Omega_x$ and

(2) $\Pi_y^H \subseteq \Omega_x$ for every $\forall$-variable y with $\Pi_y^B \subseteq \Omega_x$

That is, $\Omega_z = \{Assist[2], Nurse[2]\}$ and $\Omega_w = \{Nurse[1]\}$

EDG of Σ has:

(1) $\exists$-variables as its nodes,

(2) There is an edge from x to y if the rule where y occurs contains a $\forall$-variable z in its body with $\Pi_z^B \subseteq \Omega_x$

In this example, EDG of Σ has two nodes: z and w

There is only one edge from z to w

A set of tgds Σ is joint acyclic (JA) if its EDG is acyclic

Σ is JA (because EDG is acyclic)

We now define $\exists$-infinite positions of Σ:

$\Pi^{\exists}_{\infty}(\Sigma) \ := \ \bigcup \Omega_{x_i}$, with x_is variables that appear in a cycle in the EDG

$\Pi^{\exists}_{F}(\Sigma)$ are $\exists$-finite positions (the rest of the positions)

Proposition 1: $\Pi_F(\Sigma) \subseteq \Pi^{\exists}_{F}(\Sigma)$ $(\Pi^{\exists}_{\infty}(\Sigma) \subseteq \Pi_{\infty}(\Sigma))$

In this example:

$\Pi_{\infty}(\Sigma) = \{Assist[1], Assist[2], Nurse[1], Nurse[2]\}$, while

$\Pi^{\exists}_{\infty}(\Sigma) = \emptyset$

Example: (MS) Consider a set Σ of tgds:

$$\sigma_1 : \quad \exists z\ Assist(z,x) \leftarrow Assist(\underline{x},\underline{y})$$

$$\sigma_2 : \quad \exists w\ Assist(v,w) \leftarrow Assist(\underline{u},\underline{v})$$

$$\sigma_3 : \quad Certified(x') \leftarrow Assist(x',\underline{y'}), Assist(\underline{y'},\underline{z'}), Doctor(\underline{y'})$$

$$\Pi_F(\Sigma) = \{Doctor[1]\}$$

$$\Pi_F(\Sigma) = \{Assist[1], Assist[2],\ Certified[1]\}$$

Σ is WS!

y' is repeated and marked but appears in $Doctor[1] \in \Pi_F(\Sigma)$

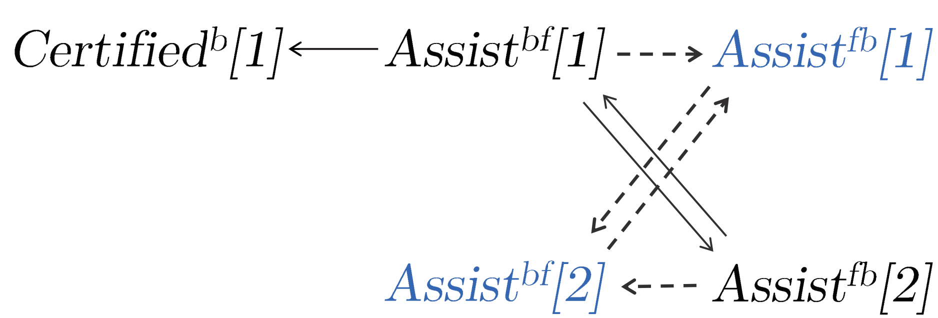
Dashed lines represent special edges

Given a query $\mathcal{Q}: \mathit{Certified}(\texttt{Marie})$ the adorned program Σ^{μ} is:

$$\begin{aligned}
r_1: &\quad \exists z\ \mathit{Assist}^{fb}(z, x) \leftarrow \mathit{Assist}^{bf}(\underline{x}, \underline{y})\\
r_2: &\quad \exists w\ \mathit{Assist}^{bf}(v, w) \leftarrow \mathit{Assist}^{fb}(\underline{u}, \underline{v})\\
r_3: &\quad \mathit{Certified}^{b}(x') \leftarrow \mathit{Assist}^{bf}(x', \underline{y'}), \mathit{Assist}^{bf}(\underline{y'}, \underline{z'}), \mathit{Doctor}(\underline{y'})
\end{aligned}$$

$$\Pi_F(\Sigma^{\mu}) = \{\mathit{Certified}^{b}[1], \mathit{Assist}^{bf}[1], \mathit{Assist}^{fb}[2], \mathit{Doctor}[1]\}$$

$$\Pi_\infty(\Sigma^{\mu}) = \{\mathit{Assist}^{bf}[2], \mathit{Assist}^{fb}[1]\}$$



Σ^{μ} is still WS (y' in r_3 appears in $\mathit{Doctor}[1] \in \Pi_F(\Sigma^{\mu})$)

The MS rewriting Σ^M contains modified rules:

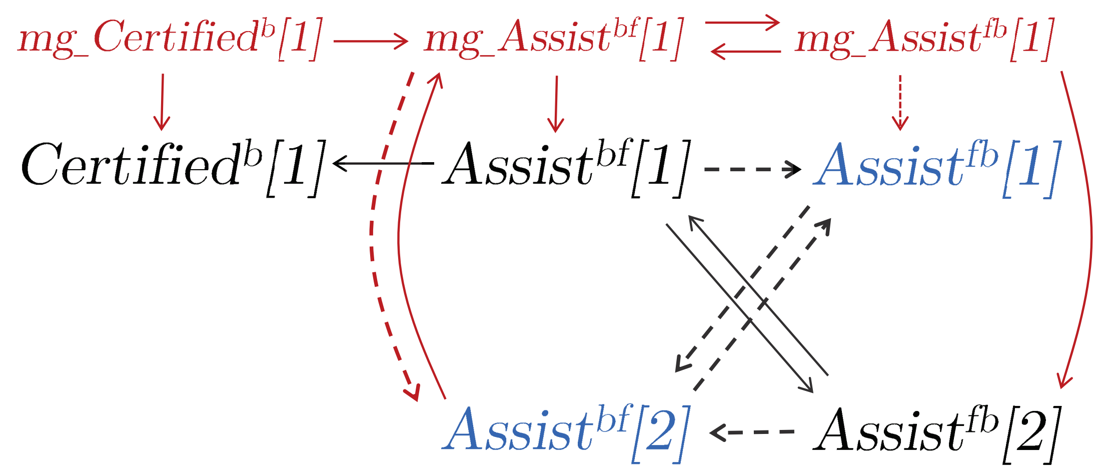
$$
\begin{aligned}
m_1: &\quad \exists z\ Assist^{fb}(z,x) \leftarrow mg\_Assist^{fb}(\underline{x}), Assist^{bf}(\underline{x},\underline{y}) \\
m_2: &\quad \exists w\ Assist^{bf}(v,w) \leftarrow mg\_Assist^{bf}(\underline{v}), Assist^{fb}(\underline{u},\underline{v}) \\
m_3: &\quad Certified^{b}(x') \leftarrow [mg\_Certified^{b}(x'), Assist^{bf}(x',\underline{y'}), \\
&\qquad Assist^{bf}(\underline{y'},\underline{z'}), Doctor(\underline{y'})]
\end{aligned}
$$

And the magic rules:

$$
\begin{aligned}
m_4: &\quad mg\_Certified^{b}(\mathtt{Marie}) \\
m_5: &\quad mg\_Assist^{bf}(x') \leftarrow mg\_Certified^{b}(\underline{x'}) \\
m_6: &\quad mg\_Assist^{bf}(y') \leftarrow mg\_Certified^{b}(\underline{x'}), Assist^{bf}(\underline{x'},\underline{y'}) \\
m_7: &\quad mg\_Assist^{fb}(v) \leftarrow mg\_Assist^{bf}(\underline{v}) \\
m_8: &\quad mg\_Assist^{bf}(x) \leftarrow mg\_Assist^{fb}(\underline{x})
\end{aligned}
$$

$$\Pi_F(\Sigma^M) = \{mg\_Certified^b[1], Doctor[1]\}$$

Σ^M is not WS! Because of repeated variables in m_1,m_2 and m_6



This proves that WS is not closed under MS rewriting

Σ is JWS since it is WS

Now consider the EDG of Σ^M:

Ω_z contains $Assist^{fb}[1]$ and Ω_w has $Assist^{bf}[2]$

Therefore $\Pi^{\exists}_{\infty}(\Sigma)$ contains $Assist^{fb}[1]$ and $Assist^{bf}[2]$

Σ^M is JWS

Example: (The QA algorithm) A WS Σ:

$$\exists z Assist(z,x) \leftarrow Assist(x,y)$$
$$\exists w Nurse(x,w) \leftarrow Doctor(x)$$
$$Certified(z,x) \leftarrow Assist(x,y), Nurse(x,z)$$

$D = \{Doctor(\mathtt{john}), Certified(\mathtt{alice}), Assist(\mathtt{john}, \mathtt{alice})\}$

CQ $\mathcal{Q} : \exists x \exists y (Assist(x,y) \wedge Assist(y, \mathtt{john}))$

We use $\mathcal{S}^{rank}$ and $\Pi_F(\Sigma) = \{Nurse[1], Nurse[2], Doctor[1]\}$

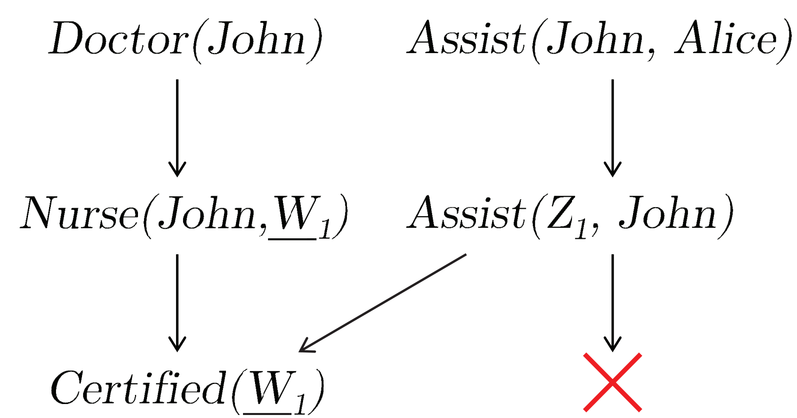
The two phases for QA:

1. *pChase* runs until termination

   However, after a pChase-step the generated nulls appearing in $\Pi_F(\Sigma)$-positions are immediately frozen

   W_1 is frozen (hence underlined)
   immediately, because it appears in $Nurse[2] \in \Pi_F(\Sigma)$

   Z_1 is not frozen, because
   $Assist[1] \in \Pi_\infty(\Sigma)$

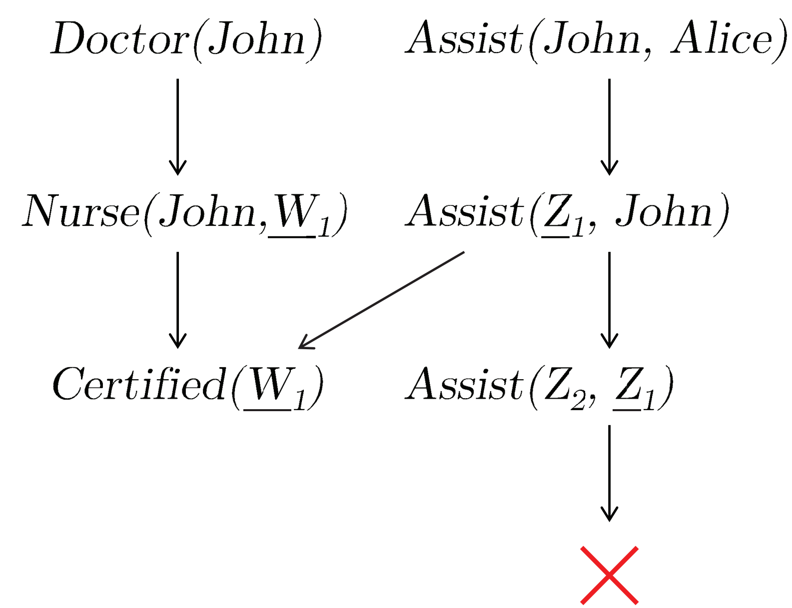


2. *pChase* iteratively resumes for a number of times that depends on the number of distinct ∃-variables that appear in a join in the query (deals with joins in the query)

y is the only ∃-variable that also appears in a join in $\mathcal{Q}$

Therefore, we freeze all nulls (e.g. Z_1), and resume the chase only once

$Assist(Z_2, Z_1)$ is entailed since Z_1 is frozen now!



$\mathcal{Q}$ true after the chase resumption!

It was false without it!

Let us now pose the query:

$$Q': \ \exists x \exists y \exists z \ (Assist(x, y) \wedge Assist(y, z) \wedge Assist(z, \texttt{john}))$$

Now the algorithm runs with two chase resumptions (because of y and z), and returns true!

$$\begin{tikzcd}
Doctor(John) \arrow[d] & Assist(John,\ Alice) \arrow[d] \\
Nurse(John, \underline{W}_1) \arrow[d] & Assist(\underline{Z}_1,\ John) \arrow[d] \arrow[dl] \\
Certified(\underline{W}_1) & Assist(\underline{Z}_2,\ \underline{Z}_1) \arrow[d] \\
 & Assist(Z_3,\ \underline{Z}_2) \arrow[d] \\
 & \times
\end{tikzcd}$$