Connecting Databases to Ontologies: A Data Quality Perspective.

Horacio Tellez, Jef Wijsen

Département d'Informatique
University of Mons, Belgium

DBDBD 2019,'s-Hertogenbosch The Netherlands, 11 December 2019

Introduction

Description Logics and Databases

Descriptive Logics are useful to databases:

- inconsistency detection,
- knowledge representation,
- enhanced query answering,
- ...

Introduction

Description Logics and Databases

Descriptive Logics are useful to databases:

- inconsistency detection,
- knowledge representation,
- enhanced query answering,
- ...

Ontology-Based Database Access (OBDA) allows accessing a database by using an ontological vocabulary, expressed in some DL.

DLs

What are DLs and ontologies ?

- a knowledge representation formalism;
- adapted to a variety of situations;
- Description Logics is the language used to write ontologies;

DLs

What are DLs and ontologies ?

- a knowledge representation formalism;
- adapted to a variety of situations;
- Description Logics is the language used to write ontologies;

Why?

- a well developed and robust theory for reasoning;
- nice complexity properties;
- well adapted to the database world;

DLs

An easy to understand syntax with multitude of constructors:

function	symbol	interpretation
true	$\top$	$\top^{\mathcal{I}} := \Delta$
false	$\bot$	$\bot^{\mathcal{I}} := \emptyset$
unary relation (concept)	A	$A^{\mathcal{I}} \subseteq \Delta$
binary relation (role)	r	$r^{\mathcal{I}} \subseteq \Delta \times \Delta$
and	$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
or	$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
negation	$\neg C$	$\Delta \backslash C^{\mathcal{I}}$
inverse role	r^{-}	$(a, b) \in (r^{-})^{\mathcal{I}} \Leftrightarrow (b, a) \in r^{\mathcal{I}}$
'exists' quantifier	$\exists r.C$	$\{x \mid \exists y \in C^{\mathcal{I}} \wedge (x, y) \in r^{\mathcal{I}}\}$
'for all' quantifier	$\forall r.C$	$\{x \mid \forall y\ (x, y) \in r^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\}$
functionality	$func(r)$	$(x, y) \in r^{\mathcal{I}} \wedge (x, z) \in r^{\mathcal{I}} \Rightarrow y = z$
transitivity	$trans(r)$	$(x, y) \in r^{\mathcal{I}} \wedge (y, z) \in r^{\mathcal{I}} \Rightarrow (x, z) \in r^{\mathcal{I}}$
numeric restriction	$(\leq n)r.C$	$\{x \mid \text{card}(\{y \mid (x, y) \in r^{\mathcal{I}}\}) \leq n\}$
constants	$\{a\}$	$a^{\mathcal{I}} \in \Delta$
...	...	...

DLs

1. $\mathcal{EL}$: $\langle \exists \mid \sqcap \mid \top \rangle$;
2. *DL- Lite* : $\langle \exists \mid \neg \mid \top \rangle$
3. $\mathcal{FL}$: $\langle \forall \mid \sqcup \mid \top \rangle$;
4. $\mathcal{ALC}$: $\langle \exists \mid \forall \mid \sqcap \mid \sqcup \mid \neg \rangle$
5. $\mathcal{S}$: $\mathcal{ALC}+$ *trans* ;
6. $\mathcal{SHOIQ}$: $\mathcal{S}+$ hierarchy $+$ inverse $+$ numeric restriction $+$ constants.

A bigger number mean a bigger (more or less) expressivity of the current logic.

Ontology

An ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is composed of two parts:

TBox $\mathcal{T}$

Store our knowledge about the studied phenomena.
$\{\text{Person} \sqsubseteq \neg\text{Book}, \mathit{motherof} \subseteq \mathit{parentof}\}$

ABox $\mathcal{A}$

Store our knowledge about individuals.
$\{\text{Book}(\text{Romeo and Juliet}), \text{Person}(\text{Romeo})\}$
It behaves itself *almost* as a database.

Ontology

An ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ is composed of two parts:

TBox $\mathcal{T}$

Store our knowledge about the studied phenomena.
$\{\text{Person} \sqsubseteq \neg\text{Book}, \textit{motherof} \subseteq \textit{parentof}\}$

ABox $\mathcal{A}$

Store our knowledge about individuals.
$\{\text{Book}(\text{Romeo and Juliet}), \text{Person}(\text{Romeo})\}$
It behaves itself *almost* as a database.

Open world assumption

A fact α is only false for an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$ if it is incompatible:

$$\langle \mathcal{T}, \mathcal{A} \rangle \wedge \alpha \models \bot.$$

Not so hidden difficulty

DLs possess powerful tools for reasoning:

- inference;
- conflicts detection;
- low complexity;

Not so hidden difficulty

DLs possess powerful tools for reasoning:

- inference;
- conflicts detection;
- low complexity;

ABoxes are not regular databases. All relation are at most of binary arity in DLs.

OBDA

Specification

An OBDA specification is a triple $(\Sigma, \mathcal{M}, \mathcal{T})$ where

- Σ is a set of constraints over a (fixed) database schema;
- $\mathcal{M}$ is a set of mapping rules, defining how database facts map to ABox assertions; and
- $\mathcal{T}$ is a TBox in some DL.

OBDA

Specification

An OBDA specification is a triple $(\Sigma, \mathcal{M}, \mathcal{T})$ where

- Σ is a set of constraints over a (fixed) database schema;
- $\mathcal{M}$ is a set of mapping rules, defining how database facts map to ABox assertions; and
- $\mathcal{T}$ is a TBox in some DL.

OBDA Semantics

- Given a database instance **db**, we write $\mathcal{M}(\mathbf{db})$ for the ABox generated from **db** by applying the rules in $\mathcal{M}$.
- $(\mathcal{T}, \mathcal{M}(\mathbf{db}))$ is a knowledge base, which can be queried, checked for consistency...
- Often, but not always, **db** is assumed to be consistent w.r.t. Σ.

Design Questions

- What is a "good" mapping language? ☜ Focus of this paper.
  Criteria: expressiveness, complexity, user-friendliness...
- Can we reconcile closed-world semantics of databases with open-world semantics of DLs?
- How to deal with inconsistent databases?
- ...

Common Formalism for Mapping Rules

Mapping

$$\forall \vec{x} \left(\varphi(\vec{x}) \to \psi(\vec{x}) \right)$$

- the body φ is a conjunction of atoms over the database schema;
- the head ψ is an atom over the vocabulary (concept names and role names) of the ontology.

Common Formalism for Mapping Rules

Mapping

$$\forall \vec{x}\,(\varphi(\vec{x}) \rightarrow \psi(\vec{x}))$$

- the body φ is a conjunction of atoms over the database schema;
- the head ψ is an atom over the vocabulary (concept names and role names) of the ontology.

Questions

- Can we add negation to the body without running into undecidability of important reasoning problems?
- Can we have a variable-free formalism alike DLs?
- Is it natural to have the same syntax for concept-generating and role-generating rules?

Our Choice for Rule Bodies: Semijoin Algebra

Semijoin Algebra

- Subset of the relational algebra

Our Choice for Rule Bodies: Semijoin Algebra

Semijoin Algebra

- Subset of the relational algebra
- Operators:
  - selection σ, projection π, attribute renaming δ, union $\cup$, difference $-$;

Our Choice for Rule Bodies: Semijoin Algebra

Semijoin Algebra

- Subset of the relational algebra
- Operators:
  - selection σ, projection π, attribute renaming δ, union $\cup$, difference $-$;
  - the join operator is replaced with the (left) semijoin operator $\ltimes$:

    $R \ltimes S$ returns the tuples in R that join with some tuple in S.

Semijoin Algebra

Syntax and semantics

- single relation: $t \in R^{\mathbf{db}}$;
- union: $t \in (E_1 \cup E_2)^{\mathbf{db}} \Leftrightarrow t \in E^{\mathbf{db}} \vee t \in E^{\mathbf{db}}$
- difference: $t \in (E_1 - E_2)^{\mathbf{db}} \Leftrightarrow t \in E_1^{\mathbf{db}} \wedge \neg t \in E_2^{\mathbf{db}}$;
- selection
  - value-based: $t \in \sigma_{A=c} E \Leftrightarrow t \in E^{\mathbf{db}} \wedge t[A] = c$;
  - attribute-based: $t \in \sigma_{A=B} E \Leftrightarrow t \in E^{\mathbf{db}} \wedge t[A] = t[B]$;
- projection: $t \in (\pi_X E)^{\mathbf{db}} \Leftrightarrow t \in E^{\mathbf{db}} \wedge sort(t) = X$;
- attribute renaming: $t \in (\delta_{A \to B} E)^{\mathbf{db}} \Leftrightarrow$
  $\exists t' \in E^{\mathbf{db}},\ \forall C \in sort(E) \setminus \{A\},\ t[C] = t'[C] \wedge t'[A] = t[B]$;
- semijoin: $t \in (E_1 \ltimes E_2)^{\mathbf{db}} \Leftrightarrow$
  $t \in E_1^{\mathbf{db}} \wedge \exists t' \in E_2^{\mathbf{db}}, \forall C \in sort(E_1) \cap sort(E_2),\ t[C] = t'[C]$.

Entity-Expressions and Concept-generating rules

Entity-Expression (EE)

Every expression in the semijoin algebra is an EE.

We use the term Database-to-ABox Dependency (DAD) for rules mapping database facts to ABox assertions.

Entity-Expressions and Concept-generating rules

Entity-Expression (EE)

Every expression in the semijoin algebra is an EE.

We use the term Database-to-ABox Dependency (DAD) for rules mapping database facts to ABox assertions.

CDAD: Concept-generating rule

If E is an EE and C is a concept name, then $E : C$ is a CDAD.

Informal meaning: if t is a tuple in E, then add the concept assertion $t : C$.

Relationship-Expressions and Role-generating rules

Relationship-Expression (RE)

- every EE is an RE;
- if E_1, E_2 are EEs, then $E_1 \bowtie E_2$ is an RE.

Relationship-Expressions and Role-generating rules

Relationship-Expression (RE)

- every EE is an RE;
- if E_1, E_2 are EEs, then $E_1 \bowtie E_2$ is an RE.

RDAD: Role-generating rules

If E_1, E_2 are EEs and E is an RE, then $[E_1, E_2, E] : r$ is a RDAD.

Informal meaning: if t_1 and t_2 are tuples in, respectively, E_1 and E_2 such that t_1 and t_2 occur together in E, then add the role assertion $(t_1, t_2) : r$.

DADs

- $(R \cup S \cup T)$: KnownData;
- $(\sigma_{B=C} S)$: $\perp$;
- $((\pi_C R \cup \pi_C S) - \pi_C T)$:GoodC;
- $(R \ltimes T)$: InterestingR;
- $[\pi_{A,B} R, \pi_{C,D} S, R \bowtie S]$: $RtoS$

Creating Individual Names

- We assume a one-one correspondence between database tuples (with attribute names) and DL individual names.
- Attribute names allow distinguishing between individuals that are value-wise the same: for example, on the DL side,

$$\{\mathit{Firstname} : \text{Paris}, \mathit{Lastname} : \text{Hilton}\}$$

and

$$\{\mathit{City} : \text{Paris}, \mathit{Hotel} : \text{Hilton}\}$$

are treated as distinct (and atomic) individual names.

Creating Individual Names

- We assume a one-one correspondence between database tuples (with attribute names) and DL individual names.
- Attribute names allow distinguishing between individuals that are value-wise the same: for example, on the DL side,

$$\{\mathit{Firstname} : \text{Paris}, \mathit{Lastname} : \text{Hilton}\}$$
and
$$\{\mathit{City} : \text{Paris}, \mathit{Hotel} : \text{Hilton}\}$$

are treated as distinct (and atomic) individual names.

Possible encoding

$\{\mathit{City}, \mathit{Firstname}, \mathit{Hotel}, \mathit{Lastname}\} \longrightarrow (x_1, x_2, x_3, x_4)$:

- $\{\mathit{Firstname} : \text{Paris}, \mathit{Lastname} : \text{Hilton}\} \longrightarrow (\varepsilon, \text{Paris}, \varepsilon, \text{Hilton})$;
- $\{\mathit{City} : \text{Paris}, \mathit{Hotel} : \text{Hilton}\} \longrightarrow (\text{Paris}, \varepsilon, \text{Hotel}, \varepsilon)$.

$\mathcal{M}$ + **db**

db =

R	A	B	C
	a	b	c
	d	e	f

S	B	C	D
	b	c	e
	b	b	m

T	C	E	F
	l	e	d
	f	g	g

$\mathcal{M}$ =

- $(R \cup S \cup T)$:KnownData; $(\sigma_{B=C} S)$:$\perp$;
- $((\pi_C R \cup \pi_C S) - \pi_C T)$:GoodC; $(R \ltimes T)$:InterestingR;
- $[\pi_{A,B} R, \pi_{C,D} S, R \bowtie S]$: $RtoS$;

$\mathcal{M}$(**db**)

KnownData $= \{(a, b, c, \varepsilon, \varepsilon, \varepsilon), (d, e, f, \varepsilon, \varepsilon, \varepsilon), (\varepsilon, b, c, e, \varepsilon, \varepsilon),$
$(\varepsilon, b, b, m, \varepsilon, \varepsilon), (\varepsilon, \varepsilon, l, \varepsilon, e, d), (\varepsilon, \varepsilon, f, \varepsilon, g, g)\}$
GoodC $= \{(\varepsilon, \varepsilon, c, \varepsilon, \varepsilon, \varepsilon), (\varepsilon, \varepsilon, b, \varepsilon, \varepsilon, \varepsilon)\}$
$\perp = \{(\varepsilon, b, b, \varepsilon, \varepsilon, \varepsilon)\}$
InterestingR $= \{(d, e, f, \varepsilon, \varepsilon, \varepsilon)\}$
$RtoS = \{((a, b, \varepsilon, \varepsilon, \varepsilon, \varepsilon), (\varepsilon, \varepsilon, c, e, \varepsilon, \varepsilon))\}$

Discussion

- Intuitive syntax without first-order variables;
- negation allowed in the body of rules;

Discussion

- Intuitive syntax without first-order variables;
- negation allowed in the body of rules;
- as for expressive power,
  - the semijoin algebra is (almost) the guarded fragment of first-order logic;
  - our mapping rules are incomparable with CQ (because joins are disallowed in CDADs).

Usage

- Ontology-based data access (OBDA)
- Data quality: ☜ Focus of this paper.
  - Detect dirty data by confronting the data in the database to the ontological "ground truth."
  - Discover conflicts between database constraints and the ontological TBox.
  - Infer missing database constraints from the ontological TBox.
  - ...

Data Quality Questions

Satisfiability

Given $(\Sigma, \mathcal{M}, \mathcal{T})$, is there a database **db** such that $\mathbf{db} \models \Sigma$ and the knowledge base $(\mathcal{T}, \mathcal{M}(\mathbf{db}))$ is consistent?

Data Quality Questions

Satisfiability

Given $(\Sigma, \mathcal{M}, \mathcal{T})$, is there a database **db** such that $\mathbf{db} \models \Sigma$ and the knowledge base $(\mathcal{T}, \mathcal{M}(\mathbf{db}))$ is consistent?

Non-Protection

Given $(\Sigma, \mathcal{M}, \mathcal{T})$, is there a database **db** such that $\mathbf{db} \models \Sigma$ but the knowledge base $(\mathcal{T}, \mathcal{M}(\mathbf{db}))$ is inconsistent?

Informally, a "yes"-answer means that the ontological TBox contains some knowledge not present in the database constraints.

Result

Theorem

Both problems *Satisfiability* and *Non-Protection* are decidable in EXPTIME if their input OBDA specifications $(\Sigma, \mathcal{M}, \mathcal{T})$ are restricted in the following way :

- Σ can be expressed in the guarded fragment;
- $\mathcal{T}$ can be expressed in the guarded fragment:
    - *DL- Lite* family;
    - $\mathcal{EL}$ family;
    - $\mathcal{ALC}$;
    - ...
- all Relationship-Expressions in $\mathcal{M}$ are join-free.

Contributions

Our framework

- Mapping rules with rule bodies in the semijoin algebra (which can be embedded in the *guarded fragment*);
- syntax without first-order variables;
- distinction between concept-generating and role-generating rules;
- only link between databases and ontologies is $\mathcal{M}$;
- some important reasoning problems are decidable.

Future work

- Complexity of the satisfiability problem for Relationship-Expressions with joins;
- database repairing in the case that the database is inconsistent;
- database repairs with both closed-world and open-world assumptions;
- consistent query answering;
- ...

Thanks!