A general Datalog-based framework for tractable query answering over ontologies

Cali, Gottlob and Lukasiewicz

Main Ideas of the paper

December 2014

Goal

Since datalog is not as expressive as Description Logic (DL) ... :

- What are the main modifications of datalog, required for ontological query-answering ?
- Are there versions of datalog that encompass the DL-Lite family and that share the favourable data complexity bounds for query-answering with DL-Lite ?

Guarded Datalog$^{\pm}$

Extension

Existentially qualified variables in rule heads.
Expressive enough to model ontologies

Tuple generating dependencies (TGD)

- Given a relational database schema $\mathcal{R}$, a TGD σ is a first-order formula of the form
$$\forall \mathbf{X} \forall \mathbf{Y}\ \phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \psi(\mathbf{X}, \mathbf{Z})$$
where $\phi(\mathbf{X}, \mathbf{Y})$ and $\psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$(without nulls).
- σ is satisfied in a database D for $\mathcal{R}$ iff whenever there exists a homomorphism h that maps the atoms of $\phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\psi(\mathbf{X}, \mathbf{Z})$ to atoms of D.
- We usually omit the universal quantifiers. All sets of TGD are finite here.

Example

Database D

employee(jo), manager(jo), directs(jo,finance), supervises(jo,ada), employee(ada), works_in(ada,finance)

Some constraints encoded as TGD

1. $manager(M) \rightarrow employee(M)$ Every manager is an employee
2. $manager(M) \rightarrow \exists P\ directs(M, P)$ Every manager directs at least one department
3. $employee(E), directs(E, P) \rightarrow \exists E'\ manager(E), supervises(E, E'), works\_in(E', P)$
   Every employee who directs a department is a manger, and supervises at least another employee who works in the same department
4. $employee(E), supervises(E, E'), manager(E') \rightarrow manager(E)$
   Every employee supervising a manager is a manager

Satisfaction

Above TGD satisfy D but do not satisfy $D' = D \cup \{manager(ada)\}$ (2nd TGD)

Query answering under TGD

Let D be a db for $\mathcal{R}$ and let Σ be a set of TGD on $\mathcal{R}$.

Models

$mods(D, \Sigma)$ (the set of models of D and Σ) is the set of all (possibly infinite) databases B s.t. :

1. $D \subseteq B$
2. every $\sigma \in \Sigma$ is satisfied in B

Evaluation of CQ (conjunctive queries)

$ans(Q, D, \Sigma)$ (the set of answers for a CQ Q to D and Σ) is the set of all tuples $\mathbf{a}$ s.t. $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$

Evaluation of BCQ (boolean conjunctive queries)

The answer of a BCQ Q to D and Σ is YES (i.e, $D \cup \Sigma \models Q$) iff $ans(Q, D, \Sigma) \neq \emptyset$

Complexity

Query answering under general TGD is undecidable

Guarded TGD

- We should restrict rule syntax for achieving decidability
- Rule bodies of TGD are guarded : in each rule body of a TGD there must exist an atom, called guard, in which all non-existentially quantified variables of the rule occur as argument.

$$P(X), R(X, Y), Q(Y) \rightarrow \exists Z \;\; R(Y, Z)$$

Expression power

More expressive than DL-Lite, and so we are going to make more restrictions...

TGD Chase

Definition

Procedure for repairing a db relative to a set of dependencies, so that the result of the chase satisfies the dependencies.

TGD Chase rule

We consider :

- A db D over a relation schema $\mathcal{R}$
- A TGD σ on $\mathcal{R}$ of the form $\forall \mathbf{Y}\ \phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\ \psi(\mathbf{X}, \mathbf{Z})$

Then σ is applicable to D if there exists a homomorphism h that maps the atoms of $\phi(\mathbf{X}, \mathbf{Y})$ to atoms of D.

Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows :
for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$;
for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a *fresh* null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced.

The application of σ on D adds to D the atom $h_1((X, Z))$ if not already in D (which is possible when Z is empty).

Chase algorithm

An exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which leads as result to a (possibly infinite) chase for D and Σ.

Equality-generating dependencies (EGD)

EGD σ

- A first-order formula of the form

$$\forall \mathbf{X}\ \phi(\mathbf{X}) \rightarrow \mathbf{X_i} = \mathbf{X_j}$$

  where the body $\phi(\mathbf{X})$ is a (not necessarily guarded) conjunction of atoms (without nulls), and X_i and X_j are variables from $\mathbf{X}$.
- The head $X_i = X_j$ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $\phi(\mathbf{X}) \subseteq D$, it holds that $h(X_i) = h(X_j)$.
- We usually omit the universal quantifiers in EGDs, and all sets of EGDs are finite here.

Example

Datalog$_0^\pm$

Datalog$_0^\pm$ can be called a DL

- Strictly more expressive than any of the DL-Lite family
- Linear TGD alone can express relationships such as $manager(X) \rightarrow manages(X, X)$ that are not expressible in any DL of DL-Lite family.

Characteristics

1. Linear TGD
2. Negative constraints
3. Non-conflicting keys

Characteristics (1)

Linear TGD

Only singleton body atom

$$\forall \mathbf{X} \forall \mathbf{Y} \phi(\mathbf{XY}) \rightarrow \exists \mathbf{Z}\ \psi(\mathbf{X}, \mathbf{Z})$$

Negative constraints

Body is a conjunction of atoms (without nulls) not necessarily guarded

$$\forall \mathbf{X}\ \phi(\mathbf{X}) \rightarrow \perp$$

Also written as

$$\forall \mathbf{X} \phi'(\mathbf{X}) \rightarrow \neg \mathbf{p}(\mathbf{X})$$

where ϕ' is obtained from ϕ by removing the atom $p(\mathbf{X})$.

Characteristics (2)

Non conflicting keys

Let K be a key.
Let σ be a TGD $\forall \mathbf{X} \forall \mathbf{Y} \phi(\mathbf{XY}) \rightarrow \exists \mathbf{Z}\ \mathbf{r}(\mathbf{X}, \mathbf{Z})$
K is non conflicting with σ iff either :

- the relational predicate on which K is defined is different from r, or
- the position of K in r are not a proper subset of the **X**-position in r in the head of σ and
- every variable in **Z** appears only once in the head of σ

Example

Four keys k_1, k_2, k_3, k_4 defined by the key attribute set $K_1 = \{r[1], r[2]\}$, $K_2 = \{r[1], r[3]\}$, $K_3 = \{r[3]\}$ and $K_4 = \{r[1]\}\}$.
TGD $\sigma = p(X, Y) \rightarrow \exists Z\ r(X, Y, Z)$
Then, the head predicate of σ is r and the set of positions in r with universally quantified variables is $\mathbf{H} = \{r[1], r[2]\}$.
All keys but K_4 are NC with σ, since only $K_4 \subseteq \mathbf{H}$
Every atom added in a chase by applying σ would have a fresh null in some position in K_1, K_2, and K_3 and thus never firing the keys k_1, k_2, k_3

DL-LITE

Elementary ingredients

Let $A \in \mathbf{C}$ be an atomic concept name, $P \in \mathbf{C}$ be an atomic role and P^- the inverse of P. In the abstract syntax :

$$
\begin{array}{lll}
B ::= A \mid \exists R & \qquad\qquad & R ::= P \mid P^- \\
C ::= B \mid \neg B & & E ::= R \mid \neg R
\end{array}
$$

B denotes a *basic concept*, *i.e.*, an atomic concept or a concept of the form $\exists R$ and R denotes a *basic role*, *i.e.*, a role that is either an atomic role or the inverse of an atomic role. Finally, C denotes a (general) concept, which can be a basic concept or its negation, whereas E denotes a *(general) role*, which can be a basic role or its negation.

A knowledge base ($\mathcal{K} = (\mathcal{T}, \mathcal{A})$) has two components : a TBox $\mathcal{T}$, used to represent intentional knowledge, and a ABox $\mathcal{A}$, used to represent extensional knowledge.

DL-Lite Family

DL-Lite Family

- DL-LITE$_{\mathcal{F}}$: no role inclusion
- DL-LITE$_{\mathcal{R}}$: no functionality constraints
- DL-LITE$_{\mathcal{A}}$: no functionality constraints on roles involved in role inclusions

`http://webdam.inria.fr/Jorge/files/slquery-onto.pdf`

Reduction to datalog$_0^\pm$

Elementary Ingredients

- Data value
- Data type
- Atomic concept
- Abstract role
- Attribute
- Individual
- Role attribute

Translation of elementary ingredients

- Every data value v has a constant $\tau(v) = c \in \Delta$ s.t. the $\tau(\mathbf{V_d})$'s for all datatypes $d \in \mathbf{D}$ are pairwise disjoint.
- Every data type $d \in \mathbf{D}$ has under τ a predicate $\tau(d) = p_d$ along with the constraint $p_d(X) \wedge p'_d(X) \to \bot$ for all pairwise distinct $d, d' \in \mathbf{D}$.
- Every atomic concept $A \in \mathbf{A}$ has a unary predicate $\tau(A) = p_A \in \mathcal{R}$.
- Every abstract role $P \in \mathbf{R_A}$ has a binary predicate $\tau(P) = p_p \in \mathcal{R}$.
- Every attribute $U \in \mathbf{R_D}$ has a binary predicate $\tau(U) = p_U \in \mathcal{R}$.
- Every individual $i \in \mathbf{I}$ has a constant $\tau(i) = c_i \in \Delta \setminus \bigcup_{d \in \mathbf{D}} \tau(\mathbf{V_d})$. $\Longleftarrow$ distinction between data values and individuals ! ! ! necessary ?

Translation of axioms (concept inclusion)

Every concept inclusion axiom $B \sqsubseteq C$ is translated to the TGD or constraint $\tau(B \sqsubseteq C) = \tau'(B) \rightarrow \tau''(C)$ where

- $\tau'(B)$ is defined as
  1. $p_A(X)$ if B is of the form A,
  2. $p_P(X, Y)$ if B is of the form $\exists P$,
  3. $p_P(Y, X)$ if B is of the form $\exists P^-$,
  4. $p_U(X, Y)$ if B is of the form $\delta(U)$, $\Leftarrow$ concept attribute
  5. $p_{U_R}(X, Y, Y')$ if B is of the form $\exists \delta(U_R)$, $\Leftarrow$ role attribute
  6. $p_{U_R}(Y, X, Y')$ if B is of the form $\exists \delta(U_R)^-$, $\Leftarrow$ role attribute
- $\tau''(C)$ is defined as
  1. $p_A(X)$ if C is of the form A,
  2. $\exists Z\ p_P(X, Z)$ if C is of the form $\exists P$,
  3. $\exists Z\ p_P(Z, X)$ if C is of the form $\exists P^-$,
  4. $\exists Z\ p_U(X, Z)$ if C is of the form $\delta(U)$,
  5. $\neg p_A(X)$ if C is of the form $\neg A$,
  6. $\neg p(X, Y')$ if C is of the form $\neg \exists P$,
  7. $\neg p(Y', X)$ if C is of the form $\neg \exists P$,
  8. $\neg p_U(X, Y')$ if C is of the form $\neg \delta(U)$,
  9. $\exists Z\ p_P(X, Z) \wedge p_A(Z)$ if C is of the form $\exists P.A$, $\Leftarrow$ See !
  10. $\exists Z\ p_P(Z, X) \wedge p_A(Z)$ if C is of the form $\exists P^-.A$. $\Leftarrow$ See !
  11. $\exists Z, Z'\ p_{U_R}(X, Z, Z')$ if C is of the form $\exists \delta(U_R)$, $\Leftarrow$ role attribute
  12. $\exists Z, Z'\ p_{U_R}(Z, X, Z')$ if C is of the form $\exists \delta(U_R)^-$, $\Leftarrow$ role attribute
  13. $\neg p_{U_R}(X, Z, Z')$ if C is of the form $\neg \exists \delta(U_R)$, $\Leftarrow$ role attribute
  14. $\neg p_{U_R}(Z, X, Z')$ if C is of the form $\neg \exists \delta(U_R)^-$, $\Leftarrow$ role attribute

Translation of axioms (functionality)

Functionality axioms

- Axiom (func P) is translated to the EGD $p_P(X, Y) \wedge p_P(X, Y') \rightarrow Y = Y'$
- Axiom (func P^-) is translated to the EGD $p_P(X, Y) \wedge p_P(X', Y) \rightarrow X = X'$
- Axiom (func U) is translated to the EGD $p_U(X, Y) \wedge p_U(X, Y') \rightarrow Y = Y'$
- Axiom (func U_R) is translated to the EGD $p_{U_R}(X, Y, Z) \wedge p_{U_R}(X, Y, Z') \rightarrow Z = Z'$ $\Leftarrow$ role attribute

Translation of axioms (membership)

Membership axioms

- Concept membership $A(a)$ is translated to $p_A(c_a)$,
- Role membership $P(a, b)$ is translated to $p_P(c_a, c_b)$,
- Attribute membership $U(a, v)$ is translated to $p_U(c_a, c_v)$,
- Role attribute membership $U_R(a, b, c)$ is translated to $p_{U_R}(c_a, c_b, c_c)$.

Translation of axioms (role inclusion)

Role inclusion

Every role inclusion axiom $Q \sqsubseteq R$ is translated to the TGD or constraint $\tau(Q \sqsubseteq R) = \tau'(Q) \rightarrow \tau''(R)$ where

- $\tau'(Q)$ is defined as :
  - $p_P(X, Y)$ if Q is of the form P,
  - $p_P(Y, X)$ if Q is of the form P^-,
  - $p_{U_R}(X, Y, Y')$ if Q is of the form $\delta(U_R)$, $\Leftarrow$ role attribute
  - $p_{U_R}(Y, X, Y')$ if Q is of the form $\delta(U_R)^-$ $\Leftarrow$ role attribute
- $\tau''(R)$ is defined as :
  - $p_P(X, Y)$ if R is of the form P,
  - $p_P(Y, X)$ if R is of the form P^-,
  - $\neg p_P(X, Y)$ if R is of the form $\neg P$,
  - $\neg p_P(Y, X)$ if R is of the form $\neg P^-$,
  - $\exists Z\ p_{U_R}(X, Y, Y')$ if R is of the form $\delta(U_R)$, $\Leftarrow$ role attribute
  - $\exists Z\ p_{U_R}(Y, X, Y')$ if R is of the form $\delta(U_R)^-$ $\Leftarrow$ role attribute
  - $\neg\exists Z\ p_{U_R}(X, Y, Y')$ if R is of the form $\neg\delta(U_R)$, $\Leftarrow$ role attribute
  - $\neg\exists Z\ p_{U_R}(Y, X, Y')$ if R is of the form $\neg\delta(U_R)^-$ $\Leftarrow$ role attribute

Translation of axioms (attribute inclusion)

Attribute inclusion axiom

- $U \sqsubseteq U'$ is translated to the TGD $p_U(X, Y) \rightarrow p_{U'}(X, Y)$,
- $U \sqsubseteq \neg U'$ is translated to the TGD $p_U(X, Y) \rightarrow \neg p_{U'}(X, Y)$
- $U_R \sqsubseteq U'_R$ is translated to the TGD $p_{U_R}(X, Y) \rightarrow p_{U'_R}(X, Y)$,
- $U_R \sqsubseteq \neg U'_R$ is translated to the TGD $p_{U_R}(X, Y) \rightarrow \neg p_{U'_R}(X, Y)$

Translation of axioms (datatype inclusion)

Datatype inclusion axiom

- Every datatype inclusion axiom $\rho(U) \sqsubseteq d$ is translated to the TGD $p_U(X, Y) \rightarrow p_d(X, Y)$,
- datatype $p_U(X, Y) \sqsubseteq \top_D$ can be safely ignored.
- Every datatype inclusion axiom $E \sqsubseteq F$ is translated to $\tau'(E) \rightarrow \tau''(F)$ where
  - $\tau'(E)$ is defined as
    1. $p_d(X)$ if E is of the form d,
    2. $p_{U_R}(Y, Y', X)$ if E is of the form $\rho(U_R)$
  - and $\tau'(F)$ is defined as
    1. $p_d(X)$ if E is of the form d,
    2. $p_{U_R}(Z, Z', X)$ if E is of the form $\rho(U_R)$,
    3. $\neg p_d(X)$ if E is of the form $\neg d$,
    4. $\neg p_{U_R}(Z, Z', X)$ if E is of the form $\neg\rho(U_R)$,