TECHNISCHE UNIVERSITÄT DRESDEN

DATABASE THEORY

Lecture 20: Query Answering Beyond Acyclic TGDs

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 17th July 2018

Beyond finite universal models

So far, we have only considered tgds with finite universal models that are produced by the chase.

Are there any other approaches of deciding entailment?

Yes, several:

- Bounded-treewidth model property
- First-order rewritability

We will start with the former.

Review: Treewidth

Definition 7.2: Consider a graph $G = \langle V, E \rangle$. A tree decomposition of G is a tree structure T where each node of T is a subset of V, such that:

- The union of all nodes of T is V.
- For each edge $(v_1 \to v_2) \in E$, there is a node N in T such that $v_1, v_2 \in N$.
- For every vertex $v \in V$, the set of nodes of T that contain v form a subtree of T; equivalently: if two nodes contain v, then all nodes on the path between them also contain v (connectedness condition).

Definition 7.3: The width of a tree decomposition is the size of its largest bag minus one.
The treewidth of a graph G, denoted $\text{tw}(G)$, is the smallest width of any of its tree decompositions.

The treewidth of a hypergraph is simply the treewidth of its primal graph.

Courcelle's insight

The key to exploiting tree width in reasoning is based on seminal results by Bruno Courcelle:

Theorem 20.1 (Courcelle, 1990): The first-order theory of a class of first-order structures of bounded treewidth is decidable. In particular, first-order entailment with respect to models of bounded treewidth is decidable.

Important remarks:

- Here bounded treewidth means that we consider only first-order structures whose treewidth is bounded by a previously fixed integer bound b.
- Simply requiring unbounded, finite treewidth would not be enough, e.g., all finite models have finite treewidth.
- The fixed bound can still be arbitrarily large, and for entailment can be defined based on the given theory.
- There is no elementary bound on complexity.

Bounded treewidth for tgd query answering

Courcelle highlights a path to new tgd languages over which query entailment is decidable:

Definition 20.2 (Baget et al. 2011): A set Σ of tgds is a bounded-treewidth set (bts) if, for any database instance $\mathcal{I}$, there is a universal model $\mathcal{U}$ of $\mathcal{I} \cup \Sigma$ that has finite treewidth.

Theorem 20.3: BCQ entailment over a bts Σ is decidable.

Proof: Let $\mathcal{I}$ be an instance, $\mathcal{U}$ a universal model of $\mathcal{I} \cup \Sigma$ of treewidth b, and φ a BCQ.

- By universality, $\mathcal{J} \models \varphi$ holds for all models $\mathcal{J}$ of $\mathcal{I} \cup \Sigma$ iff $\mathcal{U} \models \varphi$.
- Again by universality, this holds iff $\mathcal{J} \models \varphi$ for all models $\mathcal{J}$ of $\mathcal{I} \cup \Sigma$ that have treewidth $\leq b$.
- The latter condition can be decided by Courcelle. □

Deciding bts is hard

Theorem 20.4: Whether a set of tgds is bts is undecidable.

Proof: As before, we can use a simple reduction from BCQ entailment:

- Given tgds Σ, an instance $\mathcal{I}$, and a BCQ φ,
- let Π be the set of tgds that contain, for every n-ary predicate p, tgds

$$\varphi \wedge \top(x_1) \wedge \ldots \wedge \top(x_n) \to p(x_1, \ldots, x_n)$$

$$p(x_1, \ldots, x_n) \to \top(x_i) \qquad 1 \leq i \leq n$$

  where $\top$ is a fresh unary predicate and variables x_i do not occur in φ,
- and let Γ be a set of tgds that generates an infinite grid (=unbounded treewidth) starting from any element
- Then $\Sigma \cup \Pi \cup \Gamma \cup \mathcal{I}$ is bts iff $\Sigma, \mathcal{I} \models \varphi$:
  - in the latter case, Π ensures that all models are fully connected cliques, so the critical instance $\mathcal{I}_\star$ is a universal model of treewidth 0;
  - conversely, if φ is not entailed, treewidth will be unbounded due to Γ. □

Decidable cases

As for the finite universal model property, we can look for decidable sufficient conditions.

A prominent case are frontier-guarded tgds:

Definition 20.5: A tgd $\forall \vec{x}, \vec{y}.\varphi[\vec{x}, \vec{y}] \to \exists \vec{z}.\psi[\vec{x}, \vec{z}]$ is frontier-guarded if there is an atom $p(t_1, \ldots, t_n)$ in φ that contains all frontier variables $\vec{x}$.

Theorem 20.6: Every set of frontier-guarded tgds is bts.

Proof (sketch): One can construct a tree decomposition alongside the chase:

- Each bag contains all constants from the initial instance together with all elements that occurred in the instantiation of the head of the applied rule
  $\leadsto$ treewidth bound based on size of instance and rule heads
- The parent node of a given bag is the earliest node containing all frontier elements. □

Examples (1)

Important types of tgds are frontier-guarded:

Example 20.7: All inclusion dependencies are frontier-guarded, for example:

$$\text{Connect}(x, y, z) \to \exists v.\text{Lines}(z, v)$$

However, note that inclusion dependencies may not admit finite universal models. An example is

$$p(x, y) \to \exists v.p(y, v)$$

The special form of frontier-guarded tgds where all body variables occur in a single atom are called guarded tgds.

Therefore, all inclusion dependencies are guarded.

Examples (2)

Frontier-guarded tgds are closely related to Horn description logics.

Example 20.8: The description logic $\mathcal{EL}$ can be captured with tgds of the following basic forms

$$\begin{aligned} A(x) &\to \exists y.R(x,y) \wedge B(y) \\ R(x,y) \wedge B(y) &\to A(x) \\ A_1(x) \wedge A_2(x) &\to B(x) \end{aligned}$$

all of which are frontier guarded.

(Note: query answering for $\mathcal{EL}$ is easier than for arbitrary frontier-guarded rules.)

Reasoning with frontier-guarded tgds

Theorem 20.9 (Baget et al. 2011; Rudolph et al. 2014): BCQ entailment under frontier-guarded tgds is 2ExpTime-complete for combined complexity and P-complete for data complexity.

Remarks:

- The proof is complex and we omit it here.
- Hardness already applies to the case of binary predicates with frontiers of at most one variable.
- No practical implementations are known (some algorithms were proposed, e.g., [Ahmetaj, Ortiz, Simkus, 2018]).

First-Order Rewritable Dependencies

Rewriting Queries

Another approach towards query answering under dependencies:
top-down instead of bottom-up

Query rewriting:

- Start from a given conjunctive query
- Apply rules in a backward fashion to rewrite the query
- Compute results for all queries thus obtained

How to “apply rules in a backward fashion”?

When will this work?

Applying rules backwards

Essentially, we can apply resolution, but with some care:

Given:

- a CQ q (maybe not Boolean) of form $\exists\vec{v}.(\varphi_1 \wedge \varphi_2)$
- a tgd of form $\omega \rightarrow \exists\vec{z}.(\psi_1 \wedge \psi_2)$
- a substitution θ that is the most general unifier of φ_1 and ψ_1 (in particular, $\varphi_1\theta = \psi_1\theta$)

we can create a rewritten CQ:

$$\exists\vec{v}.(\omega\theta \wedge \varphi_2\theta)$$

provided that:

- if x is a variable in q and $z \in \vec{z}$ such that $x\theta = z\theta$, then x does not occur in φ_2
- if x is a free variable in q and $x\theta \notin \vec{z}$

Note: We are free to chose the decomposition into parts $\varphi_1 \wedge \varphi_2$ and $\psi_1 \wedge \psi_2$.

Example

Consider the following tgds:

$$\text{human}(x_1) \rightarrow \exists z_1.\text{hasMother}(x_1, z_1) \wedge \text{human}(z_1) \quad (1)$$
$$\text{student}(x_2) \rightarrow \exists z_2.\text{human}(x_2) \wedge \text{university}(z_2) \wedge \text{enrolledAt}(x_2, z_2) \quad (2)$$
$$\text{professor}(x_3) \rightarrow \text{human}(x_3) \quad (3)$$

and the CQ $\exists v.\text{hasMother}(x, v) \wedge \text{human}(v)$.

We obtain rewritings:

(a) human(x) (query + tgd (1) with $\{x_1 \mapsto x, z_1 \mapsto v\}$)
(b) student(x) ((a) + tgd (2) with $\{x_2 \mapsto x\}$)
(c) professor(x) ((a) + tgd (3) with $\{x_3 \mapsto x\}$)
(d) $\exists v.$hasMother$(x, v) \wedge$ student(v) (query + tgd (2) with $\{x_1 \mapsto v\}$)
(e) $\exists v.$hasMother$(x, v) \wedge$ professor(v) (query + tgd (3) with $\{x_1 \mapsto v\}$)

further rewritings only produce equivalent queries (possibly with renamed variables).

The set of answers entailed by the original CQ under the given tgds is exactly the union of the answers of the rewritings (with the original CQ, but ignoring the tgds).

Correctness

Theorem 20.10: Consider a CQ q and a tgd set Σ. For any database instance $\mathcal{I}$, the answers of q over $\mathcal{I}$ under the dependencies Σ are exactly the answers of the rewritings of q w.r.t. Σ over $\mathcal{I}$.

Intuition: Rewriting is similar to resolution, which is known to be sound and complete for FO reasoning

- The query corresponds to the negation of a clause: $\exists\vec{v}.A_1 \wedge \ldots \wedge A_n$ is expressed as $\forall\vec{v}.\neg A_1 \vee \ldots \vee \neg A_n$ (note: no real $\exists$ here!).
- A tgd $\omega \rightarrow \exists\vec{z}.(\psi_1 \wedge \psi 2)$ entails a weaker tgd $\omega \rightarrow \exists\vec{z}.\psi_1$
- Resolution is usually performed after skolemisation (in our case of $\vec{z}$); we obtain clauses $(\neg\omega \vee H)$ for each atom H in the skolemisation of ψ_1
- We can skip skolemisation/clausification by (a) using hyperresolution, and (b) making sure that existentials are always eliminated completely (as we did)
- Answer variables cannot be bound to existential variables (or skolem terms), as stated in our condition

Resolution remains sound (obvious) and complete (not obvious, needs proof) under these restrictions. □

Example

Why should rewritten queries not contain existential variables from tgd heads?

Consider the tgd:

$$\text{human}(x_1) \rightarrow \exists z_1.\text{hasMother}(x_1, z_1)$$

and the CQ $\exists v.\text{hasMother}(x, v) \wedge \text{professor}(v)$.

By just unifying and replacing parts of the CQ, we can create:

- $\exists v.$human$(x) \wedge$ professor(v) (rewriting with $\{x_1 \mapsto x, z_1 \mapsto v\}$)

The original query asks for children of professors, the rewritten one returns all humans as long as there is one professor in the database.

$\rightsquigarrow$ Left over existential variables incorrectly loose their connection to other atoms

Using query rewriting to answer CQs

Query rewriting can be used as a semi-decision procedure:

- Iteratively compute rewritings of a given query
- Optimisation: eliminate redundant rewritings (=queries contained in previously computed rewritings)
- Gather all answers of queries computed so far (over the database)

This process may not terminate, but it does in the following case:

Definition 20.11: A set Σ of tgds is first-order rewritable if, for any BCQ q, there is a first-order query q_Σ such that, for all database instances $\mathcal{I}$, $\mathcal{I}, \Sigma \models q$ iff $\mathcal{I} \models q_\Sigma$.

Known results: (without proofs)

- By this definition, query answering is in AC^0 for data complexity (like for FO)
- If it exists, the first-order rewriting q_Σ can always be a UCQ, and it can be obtained as the union of all rewritings as computed above
- The condition is very strong: rewriting might also work for some queries but not all

First-order rewritable tgds

Not surprisingly, first-order rewritability is undecidable (easy to show using similar proof techniques as before)

However, there are some known classes of FO-rewritable tgd languages:

- Non-recursive (predicate-acyclic) sets of tgds
- All linear tgds (with single-atom bodies)
  - and especially all inclusion dependencies
- Several variants of the description logic DL-Lite (which can be written in tgds)
- Some more elaborate cases, such as sticky tgds

The latter three are interesting since they do not generally enjoy a finite model property (and the chase might not terminate)

Rewriting algorithms have been implemented and used especially for DL-Lite

(advantage: FO queries can be answered by RDBMS; challenge: rewriting leads to large numbers of large CQs)

Summary and Outlook

Universal models of bounded treewidth allow for deciding BCQ entailment, since they generalise finite universal models

Frontier-guarded tgds are an easy-to-recognise class of tgds with this property

First-order rewritable sets of tgds allow for deciding BCQ entailment, since they can be finitely represented by unions of CQs, obtained through rewriting

Several easy-to-recognise syntactic classes with this property exist

Next topics:

- Summary
- Outlook