Slide 0

Answering Queries Using Views over Description Logics Knowledge Bases

D. Calvanese, G. De Giacomo, M. Lenzerini

Dipartimento di Informatica e Sistemistica

Università di Roma "La Sapienza"

17th Nat. Conf. on Artificial Intelligence (AAAI 2000)
July 30 – August 3, 2000 — Austin

Slide 1

Answering queries using views

Basic Idea:

- we want to compute the answer to a query over a database
- raw data in the database is not accessible
- we have information about a set of views over the database
- we may know that the database satisfies certain constraints

Slide 2

Answering queries using views – Example

Inaccessible database relations: R and S

Constraint: $\forall x \forall y\, (R(x, y) \rightarrow S(x, y))$

View: $V(x, y) \leftarrow R(x, z) \wedge R(z, y)$

Slide 2

Answering queries using views – Example

Inaccessible database relations: R and S

Constraint: $\forall x \forall y\, (R(x, y) \rightarrow S(x, y))$

View: $V(x, y) \leftarrow R(x, z) \wedge R(z, y)$

- If we know about the database that (a, a) satisfies V,
  we can conclude that a satisfies $Q(x) \leftarrow S(x, y)$

Answering queries using views – Example

Inaccessible database relations: R and S

Constraint: $\forall x \forall y\,(R(x,y) \rightarrow S(x,y))$

View: $V(x,y) \leftarrow R(x,z) \wedge R(z,y)$

Slide 2

- If we know about the database that (a,a) satisfies V, we can conclude that a satisfies $Q(x) \leftarrow S(x,y)$
- If we know that (a,a) is the only tuple satisfying V, we can conclude that a satisfies $Q(x) \leftarrow R(x,x)$

Answering queries using views – Example

Inaccessible database relations: R and S

Constraint: $\forall x \forall y\,(R(x,y) \rightarrow S(x,y))$

View: $V(x,y) \leftarrow R(x,z) \wedge R(z,y)$

Slide 2

- If we know about the database that (a,a) satisfies V, we can conclude that a satisfies $Q(x) \leftarrow S(x,y)$
- If we know that (a,a) is the only tuple satisfying V, we can conclude that a satisfies $Q(x) \leftarrow R(x,x)$

Significance: query optimization, query answering with incomplete information, data warehousing, data integration, etc.

Outline

Slide 3

- The problem of answering queries using views
- The framework
- Previous results
- $\mathcal{DLR}$: the language for expressing the KB and the queries
- Answering queries using views in $\mathcal{DLR}$
- Conclusions

Answering queries using views – Framework

Slide 4

We have the following information:

- constraints on the database in terms of a Description Logic knowledge base
- information on a set of views: for each view V
  - a definition $def(V)$ in terms of a query over the database
  - information about the extension of V in terms of a set $ext(V)$ of tuples of individuals
  - a specification of which assumption to adopt for V
- a specification of which assumption to adopt for the database domain

Slide 4

Answering queries using views – Framework

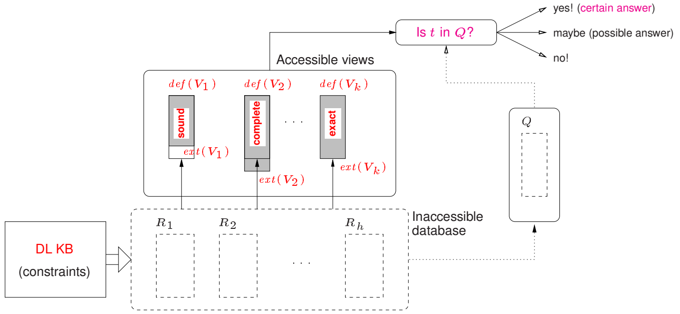
We have the following information:

- constraints on the database in terms of a Description Logic knowledge base
- information on a set of views: for each view V
  - a definition $def(V)$ in terms of a query over the database
  - information about the extension of V in terms of a set $ext(V)$ of tuples of individuals
  - a specification of which assumption to adopt for V
- a specification of which assumption to adopt for the database domain

We want to know whether a tuple t is in $ans(Q, DB)$ for every DB consistent with the information.

Slide 5

Answering queries using views – Framework



- Closed Domain Assumption: the set of objects in the database coincides with those appearing in the view extensions
- Open Domain Assumption: the set of objects in the database is a superset of those appearing in the view extensions

Slide 6

Logical reconstruction

Sound view: $\forall \vec{\mathbf{x}}\,(V(\vec{\mathbf{x}}) \to def(V)(\vec{\mathbf{x}}))$,
$V(\vec{\mathbf{a}})$, for each $\vec{\mathbf{a}} \in ext(V)$

Complete view: $\forall \vec{\mathbf{x}}\,(def(V)(\vec{\mathbf{x}}) \to (\vec{\mathbf{x}} = \vec{\mathbf{a}}_1 \vee \cdots \vee \vec{\mathbf{x}} = \vec{\mathbf{a}}_m))$,
where $\{\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_m\} = ext(V)$

Exact view: sound and complete

Closed domain assumption: $\forall x\,(x = c_1 \vee \cdots \vee x = c_s)$,
where $c_1, \ldots, c_s$ are all constants appearing in the view extensions

$\Longrightarrow$ Answering queries using views is logical implication

Slide 7

Previous work in incomplete information

[Reiter '84]
relational setting, databases with incomplete information modeled as a first order theory

[Vardi '86]
relational setting, complexity of reasoning in closed world databases with unknown values

[Donini, Lenzerini, Nardi, Schaerf '94]
reasoning with individuals in Description Logics

[van der Meyden '98]
survey on logical approaches to incomplete information

Slide 8

Previous work on answering queries using views

[Grahne, Mendelzon '99]
relational setting, conjunctive queries and views, flexible use of exact, sound, and complete views

[Abiteboul, Duschka '98]
relational setting, data complexity for different types of queries and views

[Calvanese, De Giacomo, Lenzerini, Vardi '99]
regular path queries and views, exact, sound, and complete views, different domain assumptions, data and expression complexity

[Rousset '99]
conjunctive queries over acyclic KBs in DL $\mathcal{ALN}$, atomic views (ABox)

[Horrocks, Tessaris '00]
queries over KBs in DL $\mathcal{ALCQIH}$, atomic views (ABox)

Slide 9

Outline

- The problem of answering queries using views
- The framework
- Previous results
- $\mathcal{DLR}$: the language for expressing the KB and the queries
- Answering queries using views in $\mathcal{DLR}$
- Conclusions

Slide 10

The Description Logic $\mathcal{DLR}$

Domain knowledge represented by means of:

- relations: interpreted as **sets of tuples** of a given arity
- concepts: interpreted as **sets of objects**
- individuals: interpreted as **objects** (under the unique name assumption)

Equipped with a formal semantics (first-order).

Slide 10

The Description Logic $\mathcal{DLR}$

Domain knowledge represented by means of:

- relations: interpreted as **sets of tuples** of a given arity

  $R \ ::= \ \top_n \mid P \mid \neg R \mid R_1 \sqcap R_2 \mid \$i/n : C$

- concepts: interpreted as **sets of objects**

  $C \ ::= \ \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]R \mid (\leqslant k\,[\$i]R)$

- individuals: interpreted as **objects** (under the unique name assumption)

  $a_1, a_2, a_3, \ldots$

Equipped with a formal semantics (first-order).

$\mathcal{DLR}$ knowledge base

Set of inclusion assertions on relations and concepts (TBox)

$$R_1 \sqsubseteq R_2 \qquad\qquad C_1 \sqsubseteq C_2$$

Slide 11

Set of membership assertions (ABox)

$$C(a) \qquad\qquad R(a_1, \ldots, a_n)$$

Expressiveness of $\mathcal{DLR}$

Slide 12

- subsumes most decidable DLs studied so far
- can capture conceptual and object-oriented data models used in Databases + several types of constraints:
  - arbitrary boolean constructs
  - unary and typed inclusion dependencies (e.g., referential integrity)
    $\exists[\$1]\mathtt{SOLD} \sqsubseteq \exists[\$2]\mathtt{CONTROLS}$
  - existence and exclusion dependencies (by means of $\exists$ and $\neg$)
  - unary functional dependencies
    $\mathtt{Dept} \sqsubseteq (\leqslant 1\, [\$2]\mathtt{CONTROLS})$
  - view definitions
    $P \sqsubseteq R,\ R \sqsubseteq P$ defines the view P in terms of the expression R

$\mathcal{DLR}$ knowledge base

Set of inclusion assertions on relations and concepts (TBox)

$$R_1 \sqsubseteq R_2 \qquad\qquad C_1 \sqsubseteq C_2$$

$$\begin{array}{rcl}
\mathtt{CONTROLS} & \sqsubseteq & \$1\!:\!\mathtt{Dept} \sqcap \$2\!:\!\mathtt{Dept} \\
\mathtt{Dept} & \sqsubseteq & (\leqslant 5\, [\$1]\mathtt{CONTROLS}) \sqcap (\leqslant 1\, [\$2]\mathtt{CONTROLS}) \\
\mathtt{MainDept} & \equiv & \mathtt{Dept} \sqcap \neg\exists[\$2]\mathtt{CONTROLS} \\
\mathtt{SOLD} & \sqsubseteq & \$1\!:\!\mathtt{Dept} \sqcap \$2\!:\!\mathtt{Company} \sqcap \$3\!:\!\mathtt{Money}
\end{array}$$

Slide 11

Set of membership assertions (ABox)

$$C(a) \qquad\qquad R(a_1, \ldots, a_n)$$

$$\mathtt{Dept(SWD)} \qquad \mathtt{Company(MS)} \qquad \mathtt{SOLD(SWD, MS, 30M)}$$

Queries in $\mathcal{DLR}$

A query over a $\mathcal{DLR}$ knowledge base has the form:

$$q(\vec{\mathbf{x}}) \;\leftarrow\; conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

where

Slide 13

- each $conj_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of atoms (with variables $\vec{\mathbf{x}}, \vec{\mathbf{y}}_i$)
- each atom has the form $R(t_1, \ldots, t_n)$ or $C(t)$
  (R is a relation and C a concept of the KB, $t, t_1, \ldots, t_n$ are variables or individuals)

Example: Retrieve all pairs of departments that have been sold to the same company:

$$q(d_1, d_2) \;\leftarrow\; \mathtt{Dept}(d_1) \wedge \mathtt{Dept}(d_2) \wedge \mathtt{SOLD}(d_1, c, m_1) \wedge \mathtt{SOLD}(d_2, c, m_2)$$

Reasoning in $\mathcal{DLR}$

Slide 14

Concept satisfiability and logical implication are EXPTIME-complete (reduction to satisfiability in the DL $\mathcal{CIQ}$) [De Giacomo, Lenzerini '96].

Reasoning systems for $\mathcal{DLR}$ have been implemented (iFaCT) [Horrocks '98].

Query containment and satisf. are EXPTIME-hard and in 2EXPTIME [Calvanese, De Giacomo, Lenzerini '98].

More precisely: whether $\mathcal{K} \models q_1 \subseteq q_2$ can be decided in time $O(2^{p(|\mathcal{K}|\cdot \ell_1^{\ell_2})})$

- $|\mathcal{K}|$ is the size of $\mathcal{K}$,
- ℓ_1 is the sum of the number of variables in q_1 and the number of constants in q_1, q_2, and $\mathcal{K}$,
- ℓ_2 is the number of existentially quantified variables in q_2 that appear in a semi-cycle of the tuple-graph for q_2.

Answering queries using views in $\mathcal{DLR}$ ...

Slide 15

- $\mathcal{DLR}$ knowledge base constrains the database (schema)
- query and views are unions of conjunctive queries over the $\mathcal{DLR}$ knowledge base

... under closed domain assumption

Slide 16

coNP-complete wrt data complexity

- hardness: reduction from graph-non-3-colorability
- in coNP: to check if tuple is not in the query, guess an interpretation and evaluate queries and views

in Δ_3^P wrt combined complexity (maximum arity fixed)

- coNP-complete when query and views are atomic

... under open domain assumption

Slide 17

EXPTIME-hard and in 2EXPTIME wrt. combined complexity

Reduction to satisfiability in the DL $\mathcal{CIQ}$ (which is EXPTIME-decidable):

1. encode view extensions by means of special assertions
2. translate to $\mathcal{CIQ}$ with object-names (exponential)
3. eliminate object names, thus obtaining a $\mathcal{CIQ}$ concept

EXPTIME-complete wrt. combined complexity when query, all complete views, and all exact views are atomic

Slide 18

Reduction to satisfiability in $\mathcal{CIQ}$

Is done in three steps:

1. Encode view extensions by means of special assertions:
   - For each view $V \in \mathcal{V}$ with $def(V) = v(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ and $ext(V) = \{\vec{\mathbf{a}}_1, \ldots, \vec{\mathbf{a}}_m\}$:
     - if V is sound, include $\exists \vec{\mathbf{y}}\, v(\vec{\mathbf{a}}, \vec{\mathbf{y}})$
     - if V is complete, include $\forall \vec{\mathbf{x}} \forall \vec{\mathbf{y}}\, ((\vec{\mathbf{x}} \neq \vec{\mathbf{a}}_1 \wedge \cdots \wedge \vec{\mathbf{x}} \neq \vec{\mathbf{a}}_m) \rightarrow \neg v(\vec{\mathbf{x}}, \vec{\mathbf{y}}))$
     - if V is exact treat it as both sound and complete
   - To check whether $\vec{\mathbf{d}} \in cert(Q, \mathcal{K}, \mathcal{V})$, include $\forall \vec{\mathbf{y}}\, \neg q(\vec{\mathbf{d}}, \vec{\mathbf{y}})$

Slide 19

Reduction to satisfiability in $\mathcal{CIQ}$ (cont.)

2. Translate to $\mathcal{CIQ}$ with object-names:
   - eliminate n-ary relations by means of reification
   - internalize the assertions in $\mathcal{K}$
   - translate each existentially quantified formula into a concept, treating every existentially quantified variable as a new object name
   - translate each universally quantified formula into a conjunction of concepts, one for each possible instantiation of the universally quantified variables with the object names in $\mathcal{K}$
3. Eliminate object names, thus obtaining a $\mathcal{CIQ}$ concept.

Slide 20

Conclusions

First approach to answering queries using views in expressive DLs
$\Longrightarrow$ algorithm provides the perfect rewriting [Beeri, Levy, Rousset '97]

Further work

- combined complexity for closed domain assumption
- data complexity for open domain assumption
- more expressive languages ($\mathcal{DLR}_{reg}$, $\mathcal{DLR}_{\mu}$)
- query rewriting using views:
  - perfect rewriting in a declarative language
  - maximal rewritings in languages with polynomial data complexity
- implementation (which version of FaCT?)