

Data Integration: Query Evaluation

Jan Chomicki

University at Buffalo

Interpreting schema mappings

Semantics

- M : function mapping source instances to **sets** of target instances:

$$M : I(S) \mapsto 2^{I(T)}$$

where S is a source schema and T is a target schema

- specified using assertions (**source-to-target dependencies**) or queries
- **completeness** assumptions: OWA vs. CWA
- **special** classes: GAV, LAV, GLAV

Certain answers

A tuple \mathbf{t} is a **certain answer** to a query Q over the source instance $s \in I(S)$ with respect to M if $\mathbf{t} \in Q(w)$ for every target instance $w \in M(s)$.

CWA vs. OWA

- **Closed World Assumption (CWA)**: complete knowledge
- **Open World Assumption (OWA)**: incomplete knowledge

Setting

- *source-to-target dependencies*:
 - under OWA: $\forall \mathbf{t}. \phi_S(\mathbf{t}) \Rightarrow R(\mathbf{t})$
 - under CWA: $\forall \mathbf{t}. \phi_S(\mathbf{t}) \Leftrightarrow R(\mathbf{t})$
 - $\phi_S(\mathbf{t})$: disjunction of conjunctions of source atoms
- queries: unions of conjunctive queries (defined using Datalog)

Query evaluation by unfolding

- 1 **preprocessing**: each atom in the query is replaced by one with fresh variables and additional conditions added
- 2 **applicability**: can the head A of a rule r can be made identical to a query atom B by a renaming substitution θ of all variables?
- 3 **unfolding**: replace B by the body of a rule r to which θ has been applied
- 4 **termination**: stop when only source atoms are left
- 5 **result**: take the **union** Q_u of all obtained queries
- 6 **correctness**: the evaluation of Q_u over the source instances returns the **certain** answers (under both OWA and CWA)

Setting

- Databases:
 - *Source*: $\text{emp}(N, A), \text{num}(N, \text{Id})$
 - *Target*: $\text{name}(\text{Id}, N), \text{addr}(\text{Id}, A)$
- Source-to-target dependency (GAV):
 $\forall N, A, \text{Id}. \text{emp}(N, A) \wedge \text{num}(N, \text{Id}) \Rightarrow \text{name}(\text{Id}, N)$

1 Query:

$\text{query}(N) \text{ :- emp101}(N).$
 $\text{emp101}(N) \text{ :- name}(101, N).$

2 Preprocessing and renaming of the query atoms:

$\text{query}(N) \text{ :- emp101}(N).$
 $\text{emp101}(N1) \text{ :- name}(X, N1), X=101.$

3 Unfolding the first query rule with the second:

$\text{query}(N) \text{ :- name}(X, N), X=101.$

4 Renaming of the source-to-target dependency:

$\text{name}(\text{Id2}, N2) \text{ :- emp}(N2, A2), \text{num}(N2, \text{Id2}).$

5 Unfolding with the source-to-target dependency:

$\text{query}(N) \text{ :- emp}(N, A2), \text{num}(N, X), X=101.$

Setting

- *Source-to-target dependencies (OWA):*

$$\forall \mathbf{t}. R(\mathbf{t}) \Rightarrow \phi_T(\mathbf{t})$$

- $\phi_T(\mathbf{t})$: conjunctive query over the target
- queries: sets of Datalog rules (no inequalities).

Query rewriting

- the rewriting produces a set of Datalog rules with Skolem function symbols:
 - EDB predicates: source relations
 - IDB predicates: target relations
- function symbols can be eliminated.

Inverse rules

- for every source-to-target dependency:

$$\forall x_1, \dots, x_m. (A \Rightarrow \exists y_1, \dots, y_k. B_1 \wedge \dots \wedge B_n)$$

produce n inverse rules $B'_1 : -A, \dots, B'_n : -A$

- B'_i is like B_i , except that each of y_1, \dots, y_k is replaced by the (Skolem) term $f(x_1, \dots, x_m)$ where f is a different, unique function symbol.
- all the occurrences of the same variable are replaced by the same term

Query evaluation through rewriting

- 1 construct the inverse rules
- 2 the query rule and the inverse rules are evaluated bottom-up
- 3 the evaluation terminates
- 4 only the substitutions that do not contain Skolem terms are returned to the user
- 5 the result is the set of certain answers

Global-and-Local-as-view (GLAV)

Assertions

- **source-to-target (ST)** dependencies:

$$\forall \mathbf{t}. \phi_S(\mathbf{t}) \Rightarrow \phi_T(\mathbf{t})$$

where ϕ_S , ϕ_T , and ψ_T are conjunctive queries

- **target** integrity constraints Σ_t
 - tuple-generating dependencies (tgds): $\forall \mathbf{x} (\phi_T(\mathbf{x}) \Rightarrow \exists \mathbf{y} \psi_T(\mathbf{x}, \mathbf{y}))$
 - equality-generating dependencies: $\forall \mathbf{x} (\phi_T(\mathbf{x}) \Rightarrow \mathbf{x}_1 = \mathbf{x}_2)$.

Query evaluation in data exchange

- 1 construct any **universal solution** J_0
- 2 evaluate the query over J_0
- 3 discard answers with nulls
- 4 the above returns **certain answers** for unions of conjunctive queries without inequalities

Solutions and certain answers

Solution

Given a source instance I , a target instance J is

- a **solution** for I if J satisfies target integrity constraints and (I, J) satisfy source-to-target dependencies
- a **universal solution** for I if it is a solution for I and there is a homomorphism from it to any other solution for I
- solutions can contain **labelled nulls**

There may be multiple solutions...

Certain answers

- query answers obtained in every solution J for I

Building a universal solution

Apply repetitively a variant of the chase to the source instance using target and source-to-target dependencies.

Chasing a tgk

- 1 find a substitution h that (1) h makes the LHS true in the constructed instance, and (2) h cannot be extended to a substitution that makes the RHS true in that instance
- 2 apply h to the RHS, mapping the existentially quantified variables to fresh labelled nulls
- 3 add the resulting facts to the instance.

Chasing an egk

Find a substitution h such that makes the LHS true and $h(x_1) \neq h(x_2)$:

- if $h(x_1)$ and $h(x_2)$ are constants, then FAILURE
- otherwise, identify $h(x_1)$ and $h(x_2)$ (preferring constants).

Source and target databases

Source: $Emp(N, A), Num(N, Id)$ **Target:** $Name(Id, N), Addr(Id, A)$

Source-to-target dependencies

$\forall n, a. Emp(n, a) \Rightarrow \exists id. Name(id, n) \wedge Addr(id, a)$

$\forall n, a, id. Emp(n, a) \wedge Num(n, id) \Rightarrow Name(id, n)$

Target constraints

$Name : N \rightarrow Id, Id \rightarrow N, Addr : Id \rightarrow A.$

Chase sequence

$I_0 = \{Emp(Li, LA), Num(Li, 111)\}$

$I_1 = \{Emp(Li, LA), Num(Li, 111), Name(id_1, Li), Addr(id_1, LA)\}$

$I_2 = \{Emp(Li, LA), Num(Li, 111), Name(id_1, Li), Addr(id_1, LA), Name(111, Li)\}$

$I_3 = \{Emp(Li, LA), Num(Li, 111), Name(111, Li), Addr(111, LA)\}$

Result

- there is a sequence of chase applications that ends in failure: **no universal solution**
- otherwise: every finite sequence that cannot be extended yields a **universal solution**

Acyclic tgds

- no cycles in the program dependency graph
 - nodes: relations
 - edges from the relations in the body of a tgd to the one in the head
- prevent the recurrent generation of labelled nulls
- more fine-grained analysis possible

Termination

For acyclic tgds, each chase sequence is of length polynomial in the size of the input.