Universität Freiburg
Institut für Informatik
Michael Meier
Fang Wei

Georges-Köhler Allee, Geb. 51
D-79110 Freiburg
Tel. (0761) 203-8126
Tel. (0761) 203-8125

Foundations of Query Languages
Summer semester 2010
May 18, 2010

5. Exercise Set: Conjunctive Queries and Chase

Exercise 1
Consider the following database schema.

$$\texttt{hasAirport}(\mathit{c\_id})$$
$$\texttt{fly}(\mathit{c\_id1},\mathit{c\_id2},\mathit{dist})$$
$$\texttt{rail}(\mathit{c\_id1},\mathit{c\_id2},\mathit{dist})$$

Specify the constraints below in First-order Logic and indicate if your specification is a tuple-generating dependency, equality-generating dependency, or none of both. In case of tuple-generating or equality-generating dependencies additionally compute their *body* and *head*.

a) α_1: If a city has an airport, then there is at least one flight departing from this city.
b) α_2: The distance of a rail connection functionally depends on the departure and destination station.
c) α_3: There is at least one flight and one train connection listed in the database.
d) α_4: Starting from Frankfurt, all cities with airport can be reached either by direct flight or by a flight with only one intermediate stop.
e) α_5: All pairs of cities with airport that have a direct train connection also have a direct flight connection.

Hint: In order to solve this exercise you may use tuple-generating dependencies that have an empty body.

Exercise 2
Consider the schema from the previous exercise, the constraint set $\Sigma := \{\alpha_1, \alpha_2, \alpha_3\}$ with

$$\alpha_1 := \forall\ c_1, c_2, c_3, d_1, d_2\ (\texttt{rail}(c_1, c_2, d_1), \texttt{rail}(c_2, c_3, d_2) \rightarrow \exists\ d_3\ \texttt{rail}(c_1, c_3, d_3))$$
$$\alpha_2 := \forall\ c_1, c_2, d_1, d_2\ (\texttt{fly}(c_1, c_2, d_1) \wedge \texttt{fly}(c_2, c_1, d_2) \rightarrow d_1 = d_2)$$
$$\alpha_3 := \forall\ c_1, c_2, d_1\ (\texttt{fly}(c_1, c_2, d_1) \rightarrow \exists\ d_2\ \texttt{fly}(c_2, c_1, d_2))$$

and the conjunctive query

$$Q: \quad \texttt{ans}(C_3) \leftarrow \texttt{rail}(\mathit{Freiburg}, C_1, D_1),\ \texttt{rail}(C_1, C_2, D_2),\ \texttt{fly}(C_2, C_3, D_3).$$

a) Describe the semantics of the constraints and the query informally.
b) Which constraints from Σ are satisfied by $\mathit{body}(Q)$? Does $\mathit{body}(Q)$ satsify Σ?

c) Chase query Q with Σ. Provide all intermediate results (= chase steps). Does it hold that $body(Q^{\Sigma}) \models \Sigma$?

Exercise 3
Consider the database schema with relations

$\texttt{Person}(SSN,Name)$
$\texttt{Professor}(SSN,Name)$
$\texttt{Course}(CourseName,SSN)$
$\texttt{Enrolled}(CourseName,Participant)$

where `Person` stores persons including social security number and name, `Professor` stores professors including social security number and name, `Course` contains course names and the SSN of the lecturer, and `Enrolled` stores course inscriptions. Further let $\Sigma := \{\beta_1, \beta_2, \beta_3\}$ be the set of the following constraints.

$\beta_1 := \forall\ s, n\ (\texttt{Professor}(s,n) \rightarrow \texttt{Person}(s,n))$
$\beta_2 := \forall\ c, s, n\ (\texttt{Course}(c,s) \wedge \texttt{Person}(s,n) \rightarrow \texttt{Professor}(s,n))$
$\beta_3 := \forall\ c, s\ (\texttt{Course}(c,s) \rightarrow \exists\ p\ \texttt{Enrolled}(c,p))$

Further consider the Conjunctive Query

Q: $\quad \texttt{ans}(C,N) \leftarrow \texttt{Professor}(S,N),\ \texttt{Course}(C,S)$

a) Describe the constraints informally.
b) Compute Q^{Σ}.
c) Compute – starting from Q^{Σ} – the set of all minimal Σ-equivalent queries.

Exercise 4
Let $\texttt{E}(src,dest)$ store the edge relation of a graph and let Q: $\texttt{ans}(X) \leftarrow \texttt{E}(X,Y)$. Find a tuple-generating dependency α such that the chase of Q with $\Sigma := \{\alpha\}$ does not terminate.

Exercise 5
Check if the constraint sets Σ_1, Σ_2, and Σ_3 are *acyclic*. Depict the relation graph and check if termination guarantees can be derived for the respective constraint set.

$$\Sigma_1 := \{\ \forall\ c_1, c_2, c_3, d_1, d_2\ (\texttt{rail}(c_1, c_2, d_1),\ \texttt{rail}(c_2, c_3, d_2) \rightarrow \exists\ d_3\ \texttt{rail}(c_1, c_3, d_3))\ \}$$

$$\begin{aligned}\Sigma_2 := \{\ & \forall\ c_1, c_2, d_1, d_2\ (\texttt{fly}(c_1, c_2, d_1) \wedge \texttt{fly}(c_2, c_1, d_2) \rightarrow d_1 = d_2),\\ & \forall\ c_1, c_2\ (\texttt{hasAirport}(c_1) \wedge \texttt{hasAirport}(c_2) \rightarrow \exists\ d\ \texttt{fly}(c_1,c_2,d)),\\ & \forall\ c_1, c_2, d_1\ (\texttt{fly}(c_1,c_2,d_1) \rightarrow \exists\ d_2\ \texttt{rail}(c_1,c_2,d_2))\ \}\end{aligned}$$

$$\Sigma_3 := \Sigma_2 \cup \{\ \forall\ x_1, x_2, x_3, d_1, d_2\ (\texttt{rail}(x_1,x_2,d_1) \wedge \texttt{fly}(x_2,x_3,d_2) \rightarrow \texttt{hasAirport}(x_2) \wedge \texttt{hasAirport}(x_3)\ \}$$

Exercise 6
An improvement of the *Acyclicity* condition is *Weak Acyclicity*. The latter is defined on top of positions inside relations rather than complete relations. For instance, the relation $\texttt{fly}(c\_id1,c\_id2,dist)$ has three positions, namely $\texttt{fly}^1$ (attribute *c_id1*), $\texttt{fly}^2$ (attribute *c_id2*), $\texttt{fly}^3$ (attribute *dist*). Basing upon the notion of positions, *Weak Acyclicity* is defined as follows.

Definition 1 Given a set of tuple-generating and equality-generating dependencies Σ, its dependency graph $\text{dep}(\Sigma) := (V, E)$ is the directed graph defined as follows. V is the set of positions that occur in the tuple-generating dependencies of Σ. There are two kind of edges in E. Add them as follows: for every tuple-generating dependency

$$\forall \overline{x}(\phi(\overline{x}) \rightarrow \exists \overline{y}\psi(\overline{x}, \overline{y})) \in \Sigma$$

and for every x in $\overline{x}$ that occurs in ψ and every occurrence of x in ϕ in position π_1

- for every occurrence of x in ψ in position π_2, add an edge $\pi_1 \rightarrow \pi_2$ (if it does not already exist).
- for every existentially quantified variable y and for every occurrence of y in a position π_2, add a special edge $\pi_1 \stackrel{*}{\rightarrow} \pi_2$ (if it does not already exist).

Σ is called *weakly acyclic* if and only if dep(Σ) has no cycles through a special edge.

Like *acyclicity*, *weak acyclicity* guarantees chase termination for every database instance.

a) Depict the dependency graphs of the constraint sets Σ_1, Σ_2, and Σ_3 from the previous exercise. Are these constraint sets *weakly acyclic*? Is it possible to derive termination guarantees?

b) Find a unary constraint set over an edge relation `E`(*src*,*dest*) that is not weakly acyclic.

Due by: June 2, 2010 before the tutorial starts.