COMP 280 : Assignment 10 Solutions

1. (4 pts) A simple airline database consists of a relation direct_flight(from_city, to_city). For each of the following queries over this database, either express it as a conjunctive query (no negation) or prove that it cannot be written as a conjunctive query.

   (a) There exists a route with only two stopovers from Houston to Paris.
   The following conjunctive query captures the above statement.

   ans() :- direct_flight(Houston, Stop1),
   direct_flight(Stop1, Stop2),
   direct_flight(Stop2, Paris),

   (b) All routes from Houston to Paris contain two stopovers.
   This query is not expressible as a conjunctive query. Consider

   $$DB_1 = \{\langle Houston, NewYork\rangle, \langle NewYork, London\rangle, tupleLondon, Paris\}$$

   and

   $$DB_2 = \{\langle Houston, NewYork\rangle, \langle NewYork, London\rangle, tupleLondon, Paris, \langle Houston, Paris\rangle\}.$$

   Clearly, $DB_1 \subseteq DB_2$. The query returns $\{true\}$ on DB_1 and returns $\{false\}$ on DB_2. Thus, for this query q, $DB_1 \subseteq DB_2$ does not imply that $q(DB_1) \subseteq q(DB_2)$. This contradicts the theorem that $DB_1 \subseteq DB_2 \rightarrow q(DB_1) \subseteq q(DB_2)$ for all conjunctive queries q. Therefore, this query is not conjunctive.

2. (3 pts) An employee database contains a relation EmpInfo with attributes Name, Division, Manager, Office, and Extension. The database has a functional dependency from Division → Manager. If the database also has a functional dependency from Name → Manager, must it also satisfy a functional dependency from Name → Division? Provide a proof or a counterexample.

   The functional dependency from Name → division need not hold. Consider the following database:

Name	Division	Manager	Office	Extension
Dave	Research	Sue	322	2548
Andrea	Sales	Bob	143	5587
Dave	Development	Sue	322	2548

   This database respects the functional dependencies Division → Manager and Name → Manager. However, it does not have a functional dependency from Name → Division because Dave works in both Research and Development.

3. (14 pts) In class, we considered the film database query "pairs of actors and directors such that the actor appeared in a film directed by the director". One proposal was the conjunctive query

   $$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor),$$

   which we determined was incorrect because a single film might have multiple directors. Instead, we decided that the correct query is

   $$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor'), Movies(Title, Dir', Actor)$$

   (a) (3 pts) Consider a movie database with a functional dependency from Title → Director. Are the two above queries equivalent in this case? Prove your answer or provide a counterexample.
   Yes, the two queries are equivalent in this case. Let Q_1 be the query

   $$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor)$$

and Q_2 be the query

$$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor'), Movies(Title, Dir', Actor).$$

Let $Movies$ be a database satisfying the functional dependency Title $\rightarrow$ Director. We must prove that $Q_1(Movies) = Q_2(Movies)$.

First, assume that $\langle D, A\rangle \in Q_1(Movies)$. We must prove that $\langle D, A\rangle \in Q_2(Movies)$. Since that $\langle D, A\rangle \in Q_1(Movies)$, then $Movies$ must contain a tuple $t = \langle T, D, A\rangle$ for some title T. By defintion, Q_2 will construct an answer $\langle D_1, A_2\rangle$ from any pair of tuples $t_1 = \langle T_1, D_1, A_1\rangle$ and $t_2 = \langle T_1, D_2, A_2\rangle$ in $Movies$. Letting $t = t_1 = t_2$, $D_1 = D$ and $A_2 = A$. Therefore, $\langle D, A\rangle$ is in $Q_2(Movies)$.

Next, assume that $\langle D, A\rangle \in Q_2(Movies)$. We must prove that $\langle D, A\rangle \in Q_1(Movies)$. Since $\langle D, A\rangle \in Q_2(Movies)$, $Movies$ must contain tuples $t_1 = \langle T, D, A_1\rangle$ and $t_2 = \langle T, D_2, A\rangle$ for some T, A_1, ad D_2. Since t_1 and t_2 have the same title T, The functional dependency Title $\rightarrow$ Director requires D and D_1 to be the same. Thus, we can rewrite t_2 as $\langle T, D, A\rangle$. By definition, $Q_1(Movies)$ projects the director and actor componments out of all tuples in $Movies$. Performing this projection on tuple t_2 yields answer $\langle D, A\rangle$, so $\langle D, A\rangle$ is in $Q_1(Movies)$.

(b) (3 pts) Consider a movie database with a join dependency

Movies: $\bowtie$[{Title, Actor}, {Title, Director}]

Are the two above queries equivalent in this case? Prove your answer or provide a counterexample.

Yes, the two queries are equivalent in this case. Let Q_1 be the query

$$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor)$$

and Q_2 be the query

$$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor'), Movies(Title, Dir', Actor).$$

Let $Movies$ be a database satisfying the join dependency

Movies: $\bowtie$[{Title, Actor}, {Title, Director}]

We must prove that $Q_1(Movies) = Q_2(Movies)$.

The proof that $Q_1(Movies) \subseteq Q_2(Movies)$ is the same as in part a.

Assume that $\langle D, A\rangle \in Q_2(Movies)$. We must prove that $\langle D, A\rangle \in Q_1(Movies)$. Since $\langle D, A\rangle \in Q_2(Movies)$, $Movies$ must contain tuples $t_1 = \langle T, D, A_1\rangle$ and $t_2 = \langle T, D_2, A\rangle$ for some T, A_1, ad D_2. By definition of the join dependency, $Movies = \pi_{Title,Actor}(Movies) \bowtie \pi_{Title,Director}(Movies)$. Since t_1 and t_2 are in $Movies$, $\pi_{Title,Actor}(Movies)$ contains tuples $\langle T, A_1\rangle$ and $\langle T, A\rangle$. Similarly, $\pi_{Title,Director}(Movies)$ contains tuples $\langle T, D\rangle$ and $\langle T, D_2\rangle$. The join of $\pi_{Title,Director}(Movies)$ and $\pi_{Title,Director}(Movies)$ therefore contains tuples $\langle T, D, A\rangle$, $\langle T, D, A_1\rangle$, $\langle T, D_2, A\rangle$, and $\langle T, D_2, A_1\rangle$. Since $Movies$ is equivalent to the result of this join, tuple $\langle T, D, A\rangle$ must be in $Movies$. By definition, $Q_1(Movies)$ projects the director and actor componments out of all tuples in $Movies$. Performing this projection on tuple $\langle T, D, A\rangle$ yields answer $\langle D, A\rangle$, so $\langle D, A\rangle$ is in $Q_1(Movies)$.

(c) (3 pts) Consider a movie database with a join dependency

Movies: $\bowtie$[{Director, Title}, {Director, Actor}]

Are the two above queries equivalent in this case? Prove your answer or provide a counterexample.

No, the two queries are not equivalent in this case. Let Q_1 be the query

$$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor)$$

and Q_2 be the query

$$ans(Dir, Actor) \leftarrow Movies(Title, Dir, Actor'), Movies(Title, Dir', Actor).$$

Consider the following $Movies$ database:

Title	Director	Actor
Fantasia	Algar	Mouse
Fantasia	Armstrong	Broom

This database satisfies the given join dependency because, by definition of the dependency,

$$Movies = \pi_{Director,Title}(Movies) \bowtie \pi_{Director,Actor}(Movies).$$

Calculating the projections:

$$\pi_{Director,Title}(Movies) = \{\langle Algar, Fantasia\rangle, \langle Armstrong, Fantasia\rangle\}$$

$$\pi_{Director,Actor}(Movies) = \{\langle Algar, Mouse\rangle, \langle Armstrong, Broom\rangle\}.$$

The join of these two sets produces

$$Movies = \{\langle Fantasia, Algar, Mouse\rangle, \langle Fantasia, Armstrong, Broom\rangle\}.$$

Thus, the $Movies$ database defined above satisfies the given join dependency.

$Q_1(Movies)$ returns $\{\langle Algar, Mouse\rangle, \langle Armstrong, Broom\rangle\}$. $Q_2(Movies)$ returns

$$\{\langle Algar, Mouse\rangle, \langle Algar, Broom\rangle, \langle Armstrong, Mouse\rangle, \langle Armstrong, Broom\rangle\}.$$

As $Q_2(Movies)$ contains elements not in $Q_1(Movies)$, these two queries are not equivalent in this case.

(d) (5 pts) Consider a film database with a functional dependency Title $\rightarrow$ Director and a join dependency

Movies: $\bowtie$[{Title, Actor}, {Title, Director}].

Is either dependency redundant in the presence of the other? Specifically, does the functional dependency imply the join dependency or vice-versa? For each case, provide a proof or counterexample to justify your answer.

- The join dependency does not imply the functional dependency. Consider the following $Movies$ database:

Title	Director	Actor
Fantasia	Algar	Mouse
Fantasia	Armstrong	Broom
Fantasia	Algar	Broom
Fantasia	Armstrong	Mouse

  This database satisfies the join dependency.

$$\pi_{\{Title,Actor\}}(Movies) = \{\langle Fantasia, Mouse\rangle, \langle Fantasia, Broom\rangle\}$$

$$\pi_{\{Title,Director\}}(Movies) = \{\langle Fantasia, Algar\rangle, \langle Fantasia, Armstrong\rangle\}$$

  The join of $\pi_{\{Title,Actor\}}(Movies)$ and $\pi_{\{Title,Director\}}(Movies)$ contains exactly the four tuples in the database listed above. SO the database satisfies the join dependency. However, the database does not satisfy the functional dependency, as $\langle Fantasia, Algar, Mouse\rangle$ and $\langle Fantasia, Armstrong, Mouse\rangle$ are both in the database but do not have the same Director.

- The functional dependency does imply the join dependency. To prove this, assume that the functional dependency holds, but that the join dependency does not hold. Since the join dependency does not hold, there must exist a tuple $\langle T, D, A\rangle$ which is in

$$\pi_{\{Title,Actor\}} \bowtie \pi_{\{Title,Director\}}$$

  , but which is not in the database. Since $\langle T, D, A\rangle$ is in $\pi_{\{}Title, Actor\} \bowtie \pi_{\{}Title, Director\}$, then by definition of $\bowtie$, there must exist tuples $\langle T, D_1, A\rangle$ and $\langle T, D, A_1\rangle$ in the database. Since the functional dependency holds, D and D_1 must be the same. Therefore, $\langle T, D, A\rangle$ must be in the database. This contradicts our assumption that $\langle T, D, A\rangle$ was not in the database. The assumption that the join dependency does not hold was therefore incorrect, so the functional dependency does imply the join dependency.