2. Conjunctive Queries

These are sample solutions to exercises that were given as coursework. They are not intended as models but show each one way to approach the problem set in the exercise. Some of the solutions have been produced by students for the course in 2006.

1. Normal Forms in the SPC Algebra

Demonstrate the normal form theorem for the SPC algebra (unnamed case): Every expression of the SPC algebra is equivalent to an expression of the form

$$\pi_{i_1, \ldots, i_m}(\sigma_C(R_1 \times \cdots \times R_n)).$$

*Hint:* Perform induction on the structure of the expression.

Sample solution by Ekaterina Lebedeva.

Throughout the exercise we refer to attributes of a relation according to the unnamed perspective. We use the symbols $R, S,$ possibly with subscripts to denote relations. A relation $R$ consists of a relation schema, which in the unnamed perspective is given by the number of attributes of $R$, denoted as $\text{arity}(R)$, and an instance, which is a set of tuples of length $\text{arity}(R)$. Algebra expressions are denoted as $E, F$. Also expression have an arity. If $R_1, \ldots, R_m$ are the relations occurring in $E$, then every instance of $R_1, \ldots, R_m$ uniquely determines an instance of $E$.

We use the letters $C$ or $D$ to denote conditions used in selections and $A$ or $B$ for denoting ascending sequences of positive integers. Let us note that for every sequence $A = \langle i_1, \ldots, i_n \rangle$ and algebra expression $E$, where $A$ specifies attributes of $E$, we have that $i_n \leq \text{arity}(E)$.

Suppose $R_1, \ldots, R_m$ are the relation symbols occurring in $E$ and $F$. We say that $E$ and $F$ are equivalent and write $E \equiv F$ if for all instances of $R_1, \ldots, R_m$ the instances of $E$ and $F$ are identical.

In the following we introduce some notation and highlight its properties.

**Notation 1.** Let $C$ be a condition over the set of attributes $1, \ldots, k$ and $A = \langle i_1, \ldots, i_k \rangle$ be a sequence of $k$ positive natural numbers. Then we denote by $C^A$ the condition obtained from $C$ by replacing each coordinate value $j$ occurring in $C$ by $i_j$. 
Proposition 2 (Pushing selection through projection). Let $C$ be a condition over the set of attributes $1, \ldots, k$, $A = \langle i_1, \ldots, i_k \rangle$ a sequence of attributes, and $E$ an expression such that $i_k \leq \text{arity}(E)$. Then
\[
\sigma_C(\pi_A(E)) \equiv \pi_A(\sigma_C(E)).
\]

Notation 3. Let $A = \langle i_1, \ldots, i_k \rangle$ and $B = \langle j_1, \ldots, j_n \rangle$ be two sequences of positive natural numbers such that $j_n \leq k$. Then we define the sequence $B^A$ as
\[
B^A \equiv \langle i_{j_1}, \ldots, i_{j_n} \rangle.
\]

Proposition 4 (Merging projections). Let $A = \langle i_1, \ldots, i_k \rangle$, $B = \langle j_1, \ldots, j_n \rangle$ be two sequences of positive natural numbers such that $j_n \leq k$ and let $E$ be an expression such that $i_k \leq \text{arity}(R)$. Then
\[
\pi_B(\pi_A(E)) \equiv \pi_{B^A}(E).
\]

Notation 5. By $\rho_m$ we denote the function $\rho_m : \mathbb{N} \rightarrow \mathbb{N}$ such that $\rho_m(i) = (m + i)$. If $B$ is a sequence then we denote by $\rho_m(B)$ the sequence obtained from $B$ by replacing every number $i$ in $B$ by $\rho_m(i)$. If $C$ is a selection formula then we denote by $\rho_m(C)$ the formula obtained from $C$ by replacing every coordinate $i$ of $C$ by $\rho_m(i)$.

Proposition 6 (Projections and Cartesian Products). Let $E$, $F$ be expressions, where $\text{arity}(R) = m$, and let $A = \langle i_1, \ldots, i_k \rangle$, $B = \langle j_1, \ldots, j_n \rangle$ be sequences of positive natural numbers such that $i_k \leq \text{arity}(E)$ and $j_n \leq \text{arity}(F)$. By $A \cdot \rho_m(B)$ we denote the concatenation of $A$ and $\rho_m(B)$. Then
\[
\pi_A(E) \times \pi_B(F) \equiv \pi_{A \cdot \rho_m(B)}(E \times F).
\]

Proposition 7 (Merging selections). Let $E$ be an expression and $C_1$, $C_2$ be conditions. Then
\[
\sigma_{C_1}(\sigma_{C_2}(E)) \equiv \sigma_{C_1 \land C_2}(E).
\]

Theorem 8. Every expression $E$ of the SPC Algebra is equivalent to an expression of the form
\[
\pi_A(\sigma_C(R_1 \times \cdots \times R_m)),
\]
where $A = \langle i_1, \ldots, i_n \rangle$ is a sequence of positive integers, $C$ is a selection formula, and $R_1, \ldots, R_m$ are relation names, $m > 0$.

Proof. We prove the theorem by induction over the structure of expressions in the SPC algebra. Let $E$ be an arbitrary expression in the algebra. Then $E$ is either a relation name, or $E$ is a complex expression with one of selection, projection or cross-product as the top level operator.

Base case: If $E$ is an $n$-ary relation symbol we can represent $E$ as $\pi_A(\sigma_C(E))$, where $A = \langle 1, \ldots, n \rangle$ and $C$ is equal to true.

Induction case: We make a case analysis according to the top level operator of the expression $E$.

Selection. Let $E = \sigma_C(E')$, where $C$ is a selection formula over the attributes of $E'$. By the induction hypothesis, $E'$ is equivalent to an expression $\pi_A(\sigma_{C'}(R_1 \times \cdots \times R_m))$. By Proposition 2, we have that
\[
E = \sigma_C(E')
\equiv \sigma_C(\pi_A(\sigma_{C'}(R_1 \times \cdots \times R_m)))
\equiv \pi_A(\sigma_C(\pi_A(\sigma_{C'}(R_1 \times \cdots \times R_m))))
\equiv \pi_A(\sigma_{C \land C'}(R_1 \times \cdots \times R_m)).
\]
where the last equivalence holds because of Proposition 7.

**Projection.** Let \( E = \pi_B(E') \), where \( B = \langle j_1, \ldots, j_n \rangle \) is a sequence of positive integers. By the induction hypothesis, \( E' \) is equivalent to an expression \( \pi_A(\sigma_{C'}(R_1 \times \cdots \times R_m)) \), where \( j_n \leq \text{arity}(E') \). Then by Proposition 4 we have that

\[
E = \pi_B(E') \\
\equiv \pi_B(\pi_A(\sigma_{C'}(R_1 \times \cdots \times R_m))) \\
\equiv \pi_{B^A}(\sigma_{C'}(R_1 \times \cdots \times R_m)),
\]

where \( B^A \) is a sequence defined as in Notation 3.

**Cross-product.** Let \( E = E_1 \times E_2 \). By the induction hypothesis, \( E_1 \) and \( E_2 \) can be equivalently represented in SPC normal form as

\[
E_1 \equiv \pi_A(\sigma_C(R_1 \times \cdots \times R_m)) \quad \text{and} \quad E_2 \equiv \pi_B(\sigma_D(R_{m+1} \times \cdots \times R_k)),
\]

where \( A, B \) refer to attributes of the expressions over which the corresponding projections are taken and \( C, D \) are selection formulas. Then

\[
E = E_1 \times E_2 \\
\equiv \pi_A(\sigma_C(R_1 \times \cdots \times R_m)) \times \pi_B(\sigma_D(R_{m+1} \times \cdots \times R_k)) \\
\equiv \pi_{A, \rho_m}(\sigma_{C \land \rho_m}(D))(R_1 \times \cdots \times R_m \times R_{m+1} \times \cdots \times R_k)
\]

where \( \rho_m(B) \) and \( \rho_m(D) \) are defined as in Notation 5.

Hence, we proved that for every expression in the SPC Algebra there is an equivalent expression in normal form. \( \square \)
2. Classes of Conjunctive Queries

We view queries as functions that map database instances to relation instances. Consider the following classes of conjunctive queries, which are distinguished by the form of the rules by which they can be defined:

\( \text{CQ} \): rules without equality "=" and disequality "\neq" atoms ("simple" conjunctive queries)

\( \text{CQ}_\text{=} \): rules that may have equality atoms, but no disequality atoms

\( \text{CQ}_{\neq} \): rules that may have disequality atoms, but no equality atoms

\( \text{CQ}_{=,\neq} \): rules that may have both, equality and disequality atoms (correspond to conjunctive queries as defined in the lecture)

\( \text{CQ}_\text{rep} \): rules that may repeat variables in the head, but do not have equality and disequality atoms

\( \text{CQ}_\text{const} \): rules that may have constants in the head, but do not have equality and disequality atoms

\( \text{CQ}_{\text{rep},\text{const}} \): rules that may repeat variables and may have constants in the head, but do not have equality and disequality atoms.

Determine which inclusions hold between these classes and which not:

- To show that class \( C \) is included in class \( C' \) (i.e., \( C \subseteq C' \)), indicate how any query in \( C \) can be equivalently expressed by a query in \( C' \).
- To show that \( C \) is not included in \( C' \) (i.e., \( C \nsubseteq C' \)), exhibit a query in \( C' \) for which you show that it cannot be expressed by a rule of the kind that defines queries in \( C \).

Sample solution by Werner Nutt.

Claim. The following inclusions hold:

- \( \text{CQ} \subset \text{CQ}_\text{rep} \), \( \text{CQ} \subset \text{CQ}_\text{const} \)
- \( \text{CQ}_\text{rep} \subset \text{CQ}_{\text{rep},\text{const}} \), \( \text{CQ}_\text{const} \subset \text{CQ}_{\text{rep},\text{const}} \)
- \( \text{CQ}_\text{rep} \subset \text{CQ}_{=,\neq} \)
- \( \text{CQ} \subset \text{CQ}_{=,\neq} \)
- \( \text{CQ}_{=,\neq} \subset \text{CQ}_{=,\neq} \), \( \text{CQ}_{=,\neq} \subset \text{CQ}_{=,\neq} \).

Moreover, all inclusions hold that follow by transitivity from the inclusions above. However, no other inclusions hold. In particular, all the above inclusions are strict.

The only inclusion that is not completely obvious is \( \text{CQ}_\text{rep} \subset \text{CQ}_{=,\neq} \). However, it is straightforward to show that the effect of repetition of variables in the head can be achieved as well by
equalities between variables in the body of a query. It remains to show that no other inclusions hold.

For prove our claim it remains to show that all inclusions are strict and that for any pair of classes \( C, C' \) that are not related by an inclusion there exist queries \( q \in C \setminus C' \) and \( q' \in C' \setminus C \). The following trivial lemma will be useful for our proof, since it allows us to exploit inclusions to conclude non-inclusions from other non-inclusions.

**Lemma 1.** Let \( A, C, C', B \) be sets such that \( A \subseteq C, C' \subseteq B \). Then

\[
C' \not\subseteq C \quad \text{implies} \quad C' \not\subseteq A \quad \text{and} \quad B \not\subseteq C.
\]

As a consequence, we only have to prove some crucial non-inclusions, from which others will follow. Of course, we get the best leverage of the lemma if we prove non-inclusions “\( C' \not\subseteq C \)” for sets \( C', C \), where \( C' \) has many supersets and \( C \) has many subsets.

Our non-inclusion proofs will all follow the same pattern. We identify a property \( P \) for which we show that all queries in \( C \) satisfy \( P \). Then we identify a query in \( C' \) that does not have this property.

**Proposition 2.** For every binary query \( q \in \text{CQ}_{\not\in} \) there exist constants \( a, b \) with \( a \neq b \) and an instance \( I \) such that \( (a, b) \in q(I) \).

**Proposition 3.** For every binary query \( q \in \text{CQ}_{\text{const}} \) one of the two following statements holds:

- \(|q(I)| \leq 1\) for all instances \( I \);
- there exist constants \( a, b \) with \( a \neq b \) and an instance \( I \) such that \( (a, b) \in q(I) \).

**Corollary 4.** \( \text{CQ}_{\text{rep}} \not\subseteq \text{CQ}_{\not\in} \) and \( \text{CQ}_{\text{rep}} \not\subseteq \text{CQ}_{\text{const}} \).

**Proof.** The query \( q(x, x) := r(x) \) is in \( \text{CQ}_{\text{rep}} \), but does not have the properties mentioned in Propositions 2 and 3. \( \square \)

**Proposition 5.** For every query \( q \in \text{CQ}_{=, \not\in} \) and every instance \( I \) we have that a constant \( c \) occurs in \( q(I) \) only if \( c \) occurs in \( I \).

**Corollary 6.** \( \text{CQ}_{\text{const}} \not\subseteq \text{CQ}_{=, \not\in} \).

**Proof.** The query \( q(a) := r(b) \) is in \( \text{CQ}_{\text{const}} \), but does not have the above property. \( \square \)

**Proposition 7.** All queries in \( \text{CQ}_{\text{rep}, \text{const}} \) are satisfiable.

**Corollary 8.** \( \text{CQ}_{=} \not\subseteq \text{CQ}_{\text{rep}, \text{const}} \).

**Proof.** The query \( q() := r(a), a=b \) is in \( \text{CQ}_{=} \) and is not satisfiable. Because of Proposition 7, \( q \) is not in \( \text{CQ}_{\text{rep}, \text{const}} \). \( \square \)

**Proposition 9.** For every binary query \( q \in \text{CQ}_{=} \) one of the two following statements holds:

- \(|q(I)| \leq 1\) for all instances \( I \);
- there exist a constants \( a \) and an instance \( I \) such that \( (a, a) \in q(I) \).

**Corollary 10.** \( \text{CQ}_{=} \not\subseteq \text{CQ}_{\text{rep}, \text{const}} \) and \( \text{CQ}_{=} \not\subseteq \text{CQ}_{=} \).

**Proof.** The query \( q() := r(a), a\neq a \) is in \( \text{CQ}_{=} \) and is not satisfiable. Because of Proposition 7, \( q \) is not in \( \text{CQ}_{\text{rep}, \text{const}} \).

The query \( q(x, y) := r(x), r(y), x\neq y \) is in \( \text{CQ}_{=} \), but does not satisfy the property mentioned in Proposition 9. Hence, \( q \) is not in \( \text{CQ}_{=} \). \( \square \)

This proves the claim.
3. Unions of Conjunctive Queries

Show that adding union to simple conjunctive queries strictly increases the expressivity of the resulting query language. (Recall from the previous exercise that simple conjunctive queries have neither equality nor disequality atoms.)

Hint 1: Consider the query defined by the two rules

\[
\text{ans}() \leftarrow p(1) \\
\text{ans}() \leftarrow p(2)
\]

and show that no query defined by a single rule is equivalent to it.

Sample solution by Evgeny Kharlamov.

Let us recall several definitions. We view a database instance as a set of ground atoms and a disjunctive query as a set of conjunctive queries. We define the answer set of a disjunctive query as the union of the answer sets of all conjunctive queries in it.

The following proposition gives a solution for the exercise.

Proposition 1. There is a disjunctive query that is not equivalent to any conjunctive query.

Proof. Consider the query \( q' \) given by the following two rules:

\[
\text{ans}() \leftarrow p(1), \\
\text{ans}() \leftarrow p(2).
\]

Assume there is a rule-based conjunctive query \( q \) that is equivalent to \( q' \). The query \( q \) is of the form

\[
\text{ans}() \leftarrow \text{R}_1(\vec{x}_1), \ldots, \text{R}_n(\vec{x}_n).
\]

Let us consider the three database instances \( \mathbf{I}_1 = \{ p(1) \} \), \( \mathbf{I}_2 = \{ p(2) \} \), \( \mathbf{I}_3 = \{ p(3) \} \). By our assumption \( q' \equiv q \). Therefore, \( q'(\mathbf{I}_i) = q(\mathbf{I}_i) \), for \( i \in \{ 1, 2, 3 \} \).

For the first database instance, \( q'(\mathbf{I}_1) = \{ () \} = q(\mathbf{I}_1) \). Therefore, by Equation (1), there is at least one valuation \( \gamma \) from the set of all variables occurring in the body of \( q \) to \( \text{adom}(\mathbf{I}_1) \), such that \( \text{R}_i(\gamma(\vec{x}_i)) \in \mathbf{I}_1 \), for \( i = 1, \ldots, n \). Taking into account that \( \mathbf{I}_1 \) consists of one element only we obtain \( \text{R}_i(\gamma(\vec{x}_i)) = p(1) \), for \( i = 1, \ldots, n \). Therefore, \( \text{R}_i = p \) and \( \gamma(\vec{x}_i) = 1 \). The latter means that either \( \vec{x}_i = 1 \) or \( \vec{x}_i = x \) and \( \gamma \colon x \mapsto 1 \). From this we conclude that \( q \) is of the form

\[
\text{ans}() \leftarrow p(t_1), \ldots, p(t_n),
\]

where \( t_i \in \{ x, 1 \} \). As a consequence, we obtain that \( q \) is equivalent to one of the following three queries:

\[
\text{ans}() \leftarrow p(x), \\
\text{ans}() \leftarrow p(1), \\
\text{ans}() \leftarrow p(x), p(1).
\]
Applying the same reasoning to $I_2$ we obtain another condition on $q$, namely that $q$ is equivalent to one of the following three queries:

\[
\begin{align*}
\text{ans}() & : - p(x), \\
\text{ans}() & : - p(2), \\
\text{ans}() & : - p(x), p(2).
\end{align*}
\]

The only way these two conditions can be satisfied is if $q$ has the form

\[
\text{ans}() : - p(x).
\]

Evaluation of the latter query over $I_3$ gives the empty tuple as the only answer. However, evaluating of $q'$ over $I_3$ gives the empty set as the answer set. This is in contradiction with our assumption. \qed