2. Positive and Conjunctive Queries

Instructions: Work in groups of 2 students. You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Note that experience has shown that Word is in general difficult to use for this kind of task. If you prefer to write up your solution by hand, submit a scanned electronic version. Please, include name and email address in your submission.

1. Safety of Positive Queries

We consider again simple positive queries, as introduced in Coursework 1. Remember that a first-order formula is *positive* if it contains only the logical symbols "$\wedge$", "$\vee$", and "$\exists$". A relational calculus query Q_φ is *positive* if the defining formula φ is positive.
We say that a positive formula is *simple* if it does *not* contain any of the built-in predicates "$<$", "$\leq$", "$=$", or "$\neq$". We call a relational calculus query a *simple* positive query if it is defined by a simple positive formula.

1. Is safety of simple positive queries decidable? If yes, what does an algorithm look like? If not, how do you prove undecidability?

2. Is every safe simple positive query domain independent? If yes, give a proof. If not, provide an example.

Hint: Review the concept of *range-restricted* variables in a formula, as defined by the algorithm on the slides. A suitably modified version may help in characterizing safety of positive queries.

(10 Points)

2. Conjunctive Queries vs. Positive Queries

In this problem we consider again simple positive queries as defined in Problem 1. A positive query is a *conjunctive query* if its defining formula does not contain disjunction. (Hence, the formula contains only conjunction and existential quantification as logical symbols.)

Show that adding union to simple conjunctive queries strictly increases the expressivity of the resulting query language. As in the previous exercise, we define simple conjunctive queries as conjunctive queries that have neither equality nor disequality atoms.

Hint 1: Consider the query defined by the formula

$$\varphi(x) = P(x) \vee R(x)$$

and show that no query defined using only conjunction and existential quantification is equivalent to it.

Hint 2: Assume there is an equivalent simple conjunctive query. Then consider several database instances distinguished by the atoms occurring in them.

(5 Points)

3. Classes of Conjunctive Queries

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

CQ: rules that have only relational atoms and where all variables in the head must also occur in some relational atom (we called these conjunctive queries "simple" conjunctive queries before);

$\mathsf{CQ}_{=}$: rules like those for simple conjunctive queries, with the exception that also equality atoms are allowed in the query bodies;

CQ_{rep}: rules like those for simple conjunctive queries, with the exception that variables in the head may be repeated;

CQ_{const}: rules like those for simple conjunctive queries, with the exception that constants may appear in the head;

$\mathsf{CQ}_{rep,const}$: rules like those for simple conjunctive queries, with the exception that constants may appear in the head and variables in the head may be repeated.

Determine which inclusions hold between these classes and which not:

- To show that class C_1 is included in class C_2 (i.e., $C_1 \subseteq C_2$), indicate how any query in C_1 can be equivalently expressed by a query in C_2.

- To show that C_1 is not included in C_2 (i.e., $C_1 \not\subseteq C_2$), identify first a property P_1 such that all queries in C_1 have property P_1, and then exhibit a query in C_2 that does not property P_1.

Clearly, some inclusions are obvious. Note also that you can derive some other inclusions exploiting the fact that set inclusion is transitive.
For this exercise it suffices to sketch the proofs.

Hint 1: The following trivial lemma will be useful for your proof, since it allows you to exploit inclusions to conclude non-inclusions from other non-inclusions. With that lemma, you should be able to classify all these query classes using *five non-inclusion proofs*.

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

$$C' \not\subseteq C \qquad \textit{implies} \qquad C' \not\subseteq A \quad \textit{and} \quad B \not\subseteq C.$$

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

(15 Points)

Tue, 19 April 2016, 23:55 hrs, to

the OLE submission page of the assignment.