Logic and Probability, EPFL, Fall 2011
Homework 1

(due Wednesday, October 12, 2011, at the beginning of class)

Val Tannen

The theme of this homework is "First-Order Logic as a query language". This idea was brought into Computer Science by E.F. Codd in 1970 and it underlies the entire field of relational database management systems.

The treatment follows, more or less, the textbook "Foundations of Databases" by Abiteboul, Hull, and Vianu, Addison-Wesley 1995.

A *relational database schema* is a non-empty set Σ of relation symbols with their arities. Relational database formalisms also permit constants. In fact, we fix a countably infinite set $\mathbb{D}$ whose elements we call *constants*. These constants can appear in formulas, in other words we work with the FO vocabulary $\Sigma \cup \mathbb{D}$.

For semantics it is more covenient—although not essential—to give a treatment a little different than that of standard FOL. Namely the set $\mathbb{D}$ is also understood as the sole *universe of discourse* for the interpretation of formulas. A *relational database instance* for a given schema Σ (a Σ-*instance*) is a first-order structure whose domain, or universe, is $\mathbb{D}$ and in which the relation symbols are interpreted by *finite* relations, while the constants are interpreted as themselves. Although, $\mathbb{D}$ is infinite, only finitely many of its elements *should* matter, namely those appearing in the relations or in a formula. This is discussed below as *domain-independence*.

Definition 1 *Fix a schema Σ. An* **FO query** *has the form $\{\mathbf{x} \mid \varphi\}$ where $\mathbf{x}$ is a tuple of variables or constants and φ is a first-order formula* with equality *over Σ such that the free variables of φ occur in $\mathbf{x}$.*

The *inputs* of the query are the Σ-instances. For each input $\mathcal{I}$, the *output* of the query $q \equiv \{\mathbf{x} \mid \varphi\}$ is the n-ary relation (where n is the length of $\mathbf{x}$)

$$q(\mathcal{I}) = \{v(\mathbf{x}) \ \mid \ \text{valuation } v \text{ such that } \mathcal{I}, v \models \varphi\}.$$

But what is a *database*? Clearly, we expect it to be finite so we would only consider the case when the schema, Σ, is finite. The database consists of interpretations for the schema symbols, hence it is a finite collection of finite relations. A database cannot encompass the entire domain $\mathbb{D}$ which is infinite. Let $\mathcal{I}$ be an instance. The *active domain* of $\mathcal{I}$, notation $adom(\mathcal{I})$, is the set of all elements of $\mathbb{D}$ that actually appear in the interpretations in $\mathcal{I}$ for the relation symbols. While $\mathbb{D}$ is infinite,

$adom(\mathcal{I})$ is always finite. Moreover, given a query $q \equiv \{\mathbf{x} \mid \varphi\}$, we will denote by $adom(q)$ the (finite) set of constants that occur in φ or $\mathbf{x}$.

It is our expectation that the database $\mathcal{I}$ together with the query q completely determine the output $q(\mathcal{I})$. In particular, only the elements in $adom(\mathcal{I}) \cup adom(q)$ can appear in the output. Moreover, our exact choice of $\mathbb{D}$ should not matter either. This is not the case for all FO queries.

Problem 1 *For each of the following FO queries explain intuitively (in a couple of sentences!) why their output is not completely determined by the database together with the query:*

(a) $\{x \mid \neg R(x)\}$

(b) $\{(x, y) \mid R(x) \vee S(y)\}$

(c) $\{x \mid \forall y R(x, y)\}$

These three queries are "dependent on the domain". To capture this precisely, given a query $q \equiv \{\mathbf{x} \mid \varphi\}$, for any instance $\mathcal{I}$ and any D such that $adom(\mathcal{I}) \cup adom(q) \subseteq D \subseteq \mathbb{D}$, we denote by $q(\mathcal{I}/D)$ the output of the query on the input obtained by resticting the domain to D.

Definition 2 A query q is **domain independent** if for any $\mathcal{I}$ and any D_1, D_2 where $adom(\mathcal{I}) \cup adom(q) \subseteq D_i \subseteq \mathbb{D}, i = 1, 2$ we have $q(\mathcal{I}/D_1) = q(\mathcal{I}/D_2)$.

Problem 2 *It seems from the examples of Problem 1 that negation, disjunction, and universal quantification "cause" domain dependence. It's more subtle that this:*

(a) *Earlier you gave an intuitive explanation but now* **prove** *that* $\{x \mid \forall y R(x, y)\}$ *is domain dependent.*

(b) *Keep the universal quantification but modify this query slightly to make it domain independent.*

Are existential quantification and conjunction OK? Almost:

(c) *If the FO query* $\{\mathbf{x} \mid \varphi\}$ *is such that* φ *uses only conjunction, existential quantification, and relational atoms then it is domain independent (such a query is a called a* **conjunctive query***). (Hint: recall that by Definition 1 all the variables in* $\mathbf{x}$ *must occur in* φ*.)*

(d) *Give an example of a* domain dependent *conjunctive query if in addition to relational atoms we are allowed also equality atoms. How would you define conjunctive queries with equality so they are guaranteed to be domain independent?*

It is generally agreed that in a reasonable *query language*, all the queries should be domain independent. Therefore, general first-order queries do not make a good query language. Worse

Problem 3 *By exhibiting a reduction from FIN-VALID prove that Trakhtenbrot's Theorem implies that it is undecidable whether a first-order query is domain independent.*

So what do we do to get a reasonable query language? It is possible to define decidable *safety* restrictions on general first-order formulas such that the safe queries are domain independent and moreover for any domain independent query there exists an equivalent safe query. But the safety restrictions are ugly and, in any case, Codd had a better idea (which led eventually to SQL).

The **relational algebra** is a *many-sorted* algebra, where the sorts are the natural numbers. The idea is that the elements of sort n are finite n-ary relations. Recall the domain $\mathbb{D}$. The *carrier* of sort n of the algebra is $REL(\mathbb{D}^n)$ (the set of finite n-ary relations on $\mathbb{D}$).

If f is a many-sorted k-ary operation symbol that takes arguments of sorts $n_1, \ldots, n_k$ (in this order) and returns a result of sort n then we write its type as follows: $f : n_1 \times \cdots \times n_k \longrightarrow n_0$, and we simplify this to n for nullary ($k = 0$) operations.

The operations of the algebra, with their types and their interpretation over the relational carriers are the following:

constant singletons $\{c\} \ : \ 1 \ \ (c \in \mathbb{D})$

selection1 $\sigma^n_{ij} \ : \ n \longrightarrow n \quad (1 \leq i < j \leq n)$ interpreted as $\sigma^n_{ij}(R) = \{\mathbf{x} \in R \mid x_i = x_j\}$.

selection2 $\sigma^n_{ic} \ : \ n \longrightarrow n \quad (1 \leq i \leq n, \ c \in \mathbb{D})$ interpreted as $\sigma^n_{ic}(R) = \{\mathbf{x} \in R \mid x_i = c\}$.

projection $\pi^n_{i_1 \ldots i_k} \ : \ n \longrightarrow k \quad (1 \leq i_1, \ldots, i_k \leq n,$ not necessarily distinct)
interpreted as $\pi^n_{i_1 \ldots i_k}(R) = \{x_{i_1}, \ldots, x_{i_k} \mid \mathbf{x} \in R\}$.

cartesian(cross-) product $\times^{mn} \ : \ m \times n \longrightarrow m + n$
interpreted as $\times^{mn}(R, S) = \{x_1, \ldots, x_m, y_1, \ldots, y_n \mid \mathbf{x} \in R \wedge \mathbf{y} \in S\}$.

union $\cup^n \ : \ n \times n \longrightarrow n$ interpreted as $\cup^n(R, S) = \{\mathbf{x} \mid \mathbf{x} \in R \vee \mathbf{x} \in S\}$.

difference $-^n \ : \ n \times n \longrightarrow n$ interpreted as $-^n(R, S) = \{\mathbf{x} \mid \mathbf{x} \in R \wedge \mathbf{x} \notin S\}$.

Relational algebra *expressions* are built, respecting the sorting, from these operation symbols, using the relational schema symbols *as variables.*

Note that an obvious operation, intersection, is missing. Of course, intersection can be defined from union and difference, by De Morgan's laws. Interestingly, we also have the following:

Problem 4 *Show that intersection is definable just from cartesian product, selection, and projection.*

Given a relational schema Σ, a relational algebra *query* is an algebraic expression constructed from the symbols in Σ and the relational algebra operation symbols, for example if R, S are binary, the expression $\pi_{2414}(\sigma_{13}(R \times S)) - (R \times R)$ defines a query that returns a 4-ary relation (we omit the operation's superscripts because they can usually be reconstructed and we use infix notation for the binary operations). Given a database instance $\mathcal{I}$ as input, such a query e returns a relation $e(\mathcal{I})$ as output.

Clearly, each of the operations of the relational algebra maps finite relations to finite relations, even when the domain of the instance is infinite. In fact, using a definition similar to the one given for first-order queries, the algebra queries are all obviously domain-independent.

Next, you will show that the relational algebra and the domain-independent first-order queries have the same expressive power.

Problem 5 *Give a translation that takes any relational algebra query into an equivalent domain-independent first-order query. Proceed by induction on the structure of relational algebra expressions.*

The converse is slightly more delicate. Because the set of domain-independent FO queries is not decidable, we cannot define a translation just for these queries. Therefore, you will define a translation for *all* FO queries, such that domain-independent FO queries are translated to equivalent algebraic queries. In fact, in the next (optional0 problem you can prove a slightly more general result that will also allow you to transfer undecidability and lower bound results from logic over finite models to databases.

Problem 6 (OPTIONAL) *Give a translation that takes any first-order query q over the schema Σ into a relational algebra query e over the schema $\Sigma \cup \{D\}$ where D is a fresh unary relation symbol, such that for any $\Sigma \cup \{D\}$-instance $\mathcal{I}$, we have*

$$e(\mathcal{I}) = q(\mathcal{I}/D)$$

(Abuse of notation: we denote the interpretation of D in $\mathcal{I}$ also by D.)

Conclude that there exists a computable translation that takes any first-order query q into a relational algebra query e over the same schema such that for any instance $\mathcal{I}$, we have

$$e(\mathcal{I}) = q(\mathcal{I}/adom(\mathcal{I}) \cup adom(q))$$

(which further equals $q(\mathcal{I})$ whenever q is domain-independent). (Hint: show that the active domain can be computed in the relational algebra.)

It is possible to define a notion of satisfiability for FO queries and also one of equivalence. But the familiar undecidability demons are still present.

Definition 3 A relational algebra expression e is *satisfiable* if there exists an instance $\mathcal{I}$ such that $e(\mathcal{I}) \neq \emptyset$.

Problem 7 (OPTIONAL) *Prove that satisfiability of relational algebra queries is undecidable.*

Problem 8 (OPTIONAL) *Define* equivalence *of relational algebra queries and prove that it is undecidable.*