CS632 Prelim IA

A. Demers

28 Mar 2003 – due 4 April 2003

I am so far behind (equivalently, dissatisfied with what I've got) that I am forced to post this is stages. This is stage A. The next stage will appear within a couple of days at the latest, with due date adjusted.

As usual, you may consult others for ideas and proof approaches, but please *reference your sources* and write up answers independently.

1 Armstrong Relations for FDs

Let $R(U)$ be a relation schema, and let F be a set of FDs over it. An *Armstrong Relation* for F is a relation instance r that satisfies F but violates *every* FD not in F^+.

Question 1: Is there an Armstrong Relation for every finite set of FDs?

If yes, show how to construct one (no need for efficiency).

If no, give a counterexample – a set of FDs along with an argument why no Armstrong Relation exists.

2 Dependencies as First Order Formulas

Here we generalize FDs and JDs to a framework reminiscent of DRC.

Let

$$\boldsymbol{R}(U) \;=\; (R_1(X_1), \ldots, R_n(X_n))$$

be a relation scheme. Let $\{x_1, x_2, \ldots\}$ be a countable set of variables. As usual, we use symbols like y, z_i, w' and so forth as arbitrary variables from this set. We allow two kinds of atomic formulas (atoms),

a *relation atom* of the form $R_i(y_1, \ldots, y_m)$ which is true when the tuple $\langle y_1, \ldots, y_m \rangle$ is in the relation instance r_i named R_i; and

an *equality* atom of the form $y = z$, which is the usual equality formula.

Now we define a *dependency* as a first order formula of the form

$$(\forall x_1 \ldots \forall x_m)[\phi(x_1 \ldots x_m) \;\Rightarrow\; (\exists y_1 \ldots \exists y_k)\psi(x_1, \ldots, x_m, y_1, \ldots, y_k)]$$

where

formula ϕ is a conjunction of relation atoms in which every variable x_i is mentioned; and

formula ψ is a conjunction of relation and equality atoms in which none of the equality atoms mentions a variable y_j.

We classify dependencies along four axes:

Full vs embedded: A dependency is *full* if it has no existential quantifiers.

Tuple-generating vs equality-generating: A *tuple-generating* dependency (*tgd*) is one in which no equality atoms occur (in either the left or right hand sides of the implication). An *equality-generating* dependency (*egd*) is one in which the right hand side of the implication consists of a *single* equality atom.

Typed vs untyped: A *typed* dependency is on for which there is some assignment of variables to relation attributes (column positions) such that (1) variables in relation atoms always occur in their assigned column positions, and (2) each equality atom compares two variables assigned to the same column position.

Single-head vs multi-head: A dependency is *single-head* if the right hand side of the implication involves only a single atom; it is *multi-head* otherwise.

The formulas used in dependencies are a subset of the formulas used in DRC. so we can use the DRC definition to say when a dependency δ is *satisfied* by an instance $\boldsymbol{r}$. And we can use the usual notion of implication: if Γ is a set of dependencies and δ is a dependency, then $\Gamma \models \delta$ if every instance that satisfies all the formulas of Γ also satisfies δ. We extend this to $\Gamma \models \Gamma'$ in the natural way, and define equivalence by $\Gamma \equiv \Gamma'$ iff both $\Gamma \models \Gamma'$ and $\Gamma' \models \Gamma$.

It is easy to see that every (typed) dependency is equivalent to a finite set of typed tgd's and (typed) single-head egd's. The tgd's cannot in general be single-head, as you will show below.

Problem 2a: Show how to express any FD as a (full) typed egd.

Show how to express any JD as a full typed tgd.

Show how to express any ID as an untyped embedded tgd. (recall an ID is an *inclusion dependency*, a pair of sequences of attribute names $\langle A_1 \ldots A_k, B_1 \ldots B_k \rangle$, where the ID is satisfied if every sequence of values occurring in the A attributes of any tuple also occurs in the B attributes of some (other) tuple).

Problem 2b: Consider the following multi-head tgd over a binary relation scheme $R(A, B)$:

$$(\forall x_1, x_2, x_3, x_4, x_5)[(R(x_1, x_2) \wedge R(x_3, x_2) \wedge R(x_3, x_4) \wedge R(x_5, x_4)) \\ \Rightarrow \ (\exists y)(R(x1, y) \wedge R(x_5, y))]$$

Argue that there is no set of single-head tgd's that is equivalent to it.

Problem 2c: Exhibit an infinite sequence $\tau_1, \tau_2, \ldots$ of typed tgd's over a binary relation such that each is strictly weaker than the previous one; that is, $\tau_i \models \tau_{i+1}$ but $\tau_{i+1} \not\models \tau_i$ for each $i = 1$.

Problem 2d: In class we discussed the use of tableaux and the Chase procedure to test implication for FDs and JDs. Show how to adapt this technique to the more general case of full typed dependencies. That is, let Σ is a set of full typed dependencies, and τ is a full typed dependency (either a (full) typed egd or a full typed tgd – you will need to treat these cases separately). Give appropriate definitions of initial tableau, of $CHASE_\Sigma$, and of the final test to determine whether $\Sigma \models \tau$.

A detailed proof of correctness is *not* required.