14.2 Unions of Conjunctive Queries

The next step in making query formalisms more expressive is to consider several conjunctive queries (CQs) whose bodies uses relation symbols from the same vocabulary (schema) and such that all their heads have the same arity. For example

$$\begin{aligned} ans(x,d) &:- R(x,z),\ S(x,d,z) \\ ans(c,y) &:- R(c,z),\ S(c,y,z) \\ ans(x,x) &:- R(x,z),\ S(x,e,z) \end{aligned}$$

The semantics of such a query is the union of the semantics of each individual CQ hence such queries are called unions of conjunctive queries (UCQs). A characterization analogous to Proposition 14.1 can be given except we now have "such that there exists a rule and a valuation for the body of that rule".

As the CQs correspond to the SPC fragment of relational algebra, the UCQs correspond to the SPCU fragment. There is, however, a new twist. While the translation from CQs and UCQs to the relational algebra is linear in the size of the query the converse translation has this property only for CQs. Translating an SPCU query may result in an exponential blow-up as the following example shows:

$$(R_1 \cup S_1) \times \cdots \times (R_k \cup S_k)$$

CQs and UCQs also correspond precisely to certain syntactically restricted classes of FO queries. Namely CQs corrrespond to the fragment that built from atomic formulae, conjunction and existential quantification while for UCQs we add disjunction. Notice that we omit negation and universal quantification. The translations from rules to logical formulae are straightforward (existentially quantify the variables that appear in the body but not in the head). In the other direction put the formulae in prenex and disjunctive normal form. Again the translation to UCQs may involve an exponential blow-up because of the disjunctive normal form.

Equivalence of UCQs is also decidable and like equivalence of CQs it is NP-complete. However, equivalence of SPCU relational algebras expressions is Π_2^p-complete showing that the exponential blow-up in the translation we mentioned above cannot be avoided (assuming, of course, that $PTIME \neq \Pi_2^P$).

UCQs (SPCU queries) are sometimes called *positive* FO queries because they miss negation (equivalently, the difference operator in relational algebra). They are also sometimes call *monotone*. Indeed, assume that the vocabulary is $R_1, \ldots, R_k$. Abusing notation by blurring the distinction between syntax and semantics, and FO query q defines a function $q(R_1, \ldots, R_k)$. It is easy to see that if if q is an UCQ and if $R_1 \subseteq S_1, \ldots, R_k \subseteq S_k$ then $q(R_1, \ldots, R_k) \subseteq q(S_1, \ldots, S_k)$.

14.3 Datalog

We now allow predicate symbols in the heads of the rules. That is, in addition to the original vocabulary Σ_e, whose symbols we will call *extensional* we use a disjoint vocabulary Σ_i whose symbols we call *intensional* and allow both extensional and intensional predicates in the rule bodies but

only intensional predicates in the rules heads. For example, here a tortuous and highly redundant way to compute transitive closure:

$$\begin{array}{lcl} R(x,y) & :- & E(x,y) \\ S(x,y) & :- & E(x,y) \\ T(x,y) & :- & E(x,y) \\ R(x,y) & :- & E(x,z),\ S(z,y) \\ S(x,y) & :- & R(x,z),\ E(z,y) \\ T(x,y) & :- & R(x,z),\ S(z,y) \end{array}$$

Here we have one extensional symbol, E and three intensional ones R, S, T. We specify a Datalog query by additionally identifying one of the intensional predicates as the intended output.

It should be clear that we can associate a *system of fixpoint equations* in which each equation is defined as an UCQ (or an SPCU query) with every Datalog program. For example, for the program above, abusing notation we have:

$$\begin{array}{lcl} R & = & E \cup E \circ S \\ S & = & R \cup R \circ E \\ T & = & E \cup R \circ S \end{array}$$

or alternatively, in FO syntax:

$$\begin{array}{lcl} R(x,y) & = & E(x,y) \vee \exists z\, E(x,z) \wedge S(z,y) \\ S(x,y) & = & E(x,y) \vee \exists z\, R(x,z) \wedge E(z,y) \\ T(x,y) & = & E(x,y) \vee \exists z\, R(x,z) \wedge S(z,y) \end{array}$$

Since the functions defined by UCQs are monotone, the last formulation shows that Datalog is a fragment of FO(LFP) [9] .(Actually, our definition of FO(FP) does not allow for systems of equations but there exist standard tricks to encode these as single equations.) In particular, the 0-1 Law holds for Datalog. However, Datalog is strictly less expressive than FO(LFP). We can express in Datalog that two specific nodes are connected by a path (transitive closure) but we cannot express that they are *not* connected. More subtly, we cannot express that *all* pairs of nodes are connected! To prove these inexpressibility results it suffices to show that Datalog defines only monotone queries (i.e., the least fixpoint depends monotonically on the extensional predicates).

15 Probabilistic Databases

See the paper "Models for Incomplete and Probabilistic Information" by Green and Tannen, EDBT Workshops 2006, LNCS 4254, pp.278-296 which will be posted on the course wiki.

[9] The semantics that we get for Datalog via this embedding in FO(LFP) can also be show to be equivalent to a model-theoretic semantics (least Herbrand model) and two different proof-theoretic semantics, one by derivation trees and one by SLD resolution (see Abiteboul, Hull, and Vianu "Foundations of Databases").