Chapter 2.17
Rewriting and Efficient Computation of Bound Disjunctive Datalog Queries

Sergio Greco
DEIS Università della Calabria, Italy

Ester Zumpano
DEIS Università della Calabria, Italy

INTRODUCTION

A strong interest in enhancing Datalog programs by the capability of disjunction emerged in several areas, such as databases, artificial intelligence, logic programming, and so forth (Lobo, Minker & Rajasekar, 1992). Disjunctive rules have been profitably used in several contexts including knowledge representation, databases querying, and representation of incomplete information (Eiter, Gottlob & Mannila, 1997a; Gelfond & Lifschitz, 1991). Most of the works on disjunction are concerned with the definition of intuitive and expressive semantics, which are commonly based on the paradigm of minimal models (Abiteboul, Hull & Vianu, 1995; Lobo et al., 1992; Ullman, 1989).

Disjunction in the head of rules increases the expressive power of the language but makes the computation of queries very difficult, as it allows the presence of multiple models whose number is generally exponential with respect to the size of the input (Abiteboul et al., 1995; Eiter et al., 1997a).

Therefore, a great interest has been devoted to the definition of efficient algorithms for both computing the semantics of programs and answering queries.

Most of the research has concentrated on the definition of efficient fixpoint algorithms computing the semantics of programs, and the more effective proposals are based on the evaluation of the (intelligent) ground instantiation of programs and on the use of heuristics (Leone, Rullo & Scarcello, 1997). However, different from standard datalog, for disjunctive queries, there are few effective methodologies which systematically utilize the query to propagate bindings into

the body of the rules to avoid computing all the models of the program.

The following example taken from Greco (2003), shows a program in which only a strict subset of the minimal models needs to be considered to answer the query.

Example 1. Consider the disjunctive program P1 consisting of the following rule:

$$p(X) \vee q(X) \leftarrow a(X,Y)$$

and a database D consisting of the set of facts {a(1,2),a(2,3),..., a(k,k+1)}. Consider now a query asking if there is some model for $P \cup D$ containing the atom p(1). A "brute force" approach, based on an exhaustive search of the minimal models of $P \cup D$, would consider $\mathbf{2^k}$ minimal models.

However, to answer the query, we could only consider the ground rule: $p(1) \vee q(1) \leftarrow a(1,2)$ and, therefore, only two minimal (partial) (for partial model we intend a subset of the model containing all the atoms necessary to answer the query) models: M1 = { p(1) } ∪ D and M2 = { q(1) }.

From the above example, it is evident that using the query goal to reduce the size and the number of models to be considered for answering the query is necessary.

Constraints play an active role in the deductive database process as they model the interaction among data and define properties that have to be satisfied by models.

Thus, this article explores the efficient computation of bound queries over disjunctive Datalog programs enriched with constraints.

The technique here proposed is very relevant for the optimization of queries expressing hard problems. Indeed, the usual way of expressing declaratively hard problems, such as NP problems and problems in the second level of the polynomial hierarchy, is based on the *guess-and-check* technique, where the *guess* part is expressed by means of disjunctive rules, and the *check* part is expressed by means of constraints. However, as shown by the following example, the presence of constraints reduces the number of feasible models but could increase both the number of ground rules and the number of (partial) models to be considered for answering the query.

Example 2. Consider the query goal p(1) over the disjunctive program P2 consisting of the rule in the above example plus the following constraint (a rule with empty head which is true if the body is false)

$$\leftarrow p(X), a(X,Y), q(Y), X \leq 1.$$

To answer the query, we have to consider other than the ground rule:

r1: $p(1) \vee q(1) \leftarrow a(1,2)$.
also the ground rule:
r2: $p(2) \vee q(2) \leftarrow a(2,3)$.

as the atom p(1) in the head of the rule r1 is influenced by the atom q(2) through the ground constraint c1

$$\leftarrow p(1), a(1,2), q(2), 1 \leq 1.$$

Consequently, there are three stable models to be considered:

N1 = { p(1), p(2) } ∪ D,
N2 = { q(1), p(2) } ∪ D, and
N3 = { q(1), q(2) } ∪ D.

The interpretation { p(1), q(2) } ∪ D is not a model as it does not satisfy the constraint.

The above example shows the necessity of an effective methodology that uses the properties specified by the constraints and the query goal in order to compute correct answers avoiding the evaluation of useless models. However, in the presence of constraints, we have to consider ground rules defining atoms on which the query