Chapter 3

Disjunctive Sources

We examine the query planning problem in data integration systems in the presence of sources that contain disjunctive data. We show that datalog, the language of choice for representing query plans in data integration systems, is not sufficiently expressive in this case. We prove that disjunctive datalog with inequality, on the other hand, is sufficiently expressive by presenting a construction of query plans that are guaranteed to extract all available information from disjunctive sources.

3.1 Introduction

We examine the query planning problem in data integration systems in the presence of sources that contain disjunctive data. The query planning problem in such systems can be formally stated as the problem of answering queries using views as described in Chapter 2. View definitions describe the data stored by sources, and query planning requires rewriting a query into one that only uses these views. In this chapter we are going to extend the algorithm for answering queries using conjunctive views introduced in Section 2.3 so that it can handle disjunction in the view definitions as well.

Example 3.1.1 Assume a data source stores flight information. More precisely, the source stores nonstop flights by United Airlines (ua) and Southwest Airlines (sw), and flights out of San Francisco International Airport (sfo) with one stopover. The data stored by this source can be described as being a view over a database with a relation *flight* that stores all nonstop flights. The view definition that describes this source is the following:

$$
\begin{aligned}
v(ua, From, To) &\ :- \ flight(ua, From, To) \\
v(sw, From, To) &\ :- \ flight(sw, From, To) \\
v(Airline, sfo, To) &\ :- \ flight(Airline, sfo, Stopover), \\
&\qquad\quad flight(Airline, Stopover, To)
\end{aligned}
$$

A user might be interested in all cities that have nonstop flights to Seattle (sea):

$$
\mathcal{Q}: \quad q(From) \ :- \ flight(Airline, From, sea)
$$

If $\langle ua,jfk,sea\rangle$ is a tuple stored by the data source, then there is clearly a nonstop flight from New York (jfk) to Seattle. On the other hand, if the tuple $\langle ua,sfo,sea\rangle$ is stored by the data source then a nonstop flight from San Francisco to Seattle does not necessarily exist. Indeed, this tuple might be stored because there is a flight with one stopover from San Francisco to Seattle. The task of query planning in data integration systems is to find a query plan, i.e. a query that only requires views, that extracts as much information as possible from the available sources. All flights to Seattle stored by the data source with the exception of flights departing from San Francisco International Airport are nonstop flights. Therefore, the query plan is the following:

$\mathcal{P}$: $\quad q(From) \ :- \ v(Airline,From,sea), \ \ From \neq sfo$

Note that without the use of the inequality constraint "$From \neq sfo$" it wouldn't be possible to guarantee that all cities returned by the query plan indeed have nonstop flights to Seattle. □

In Chapter 2, we showed that the expressive power of datalog is both required and sufficient to represent "good" query plans in data integration systems when view definitions are restricted to be conjunctive. As we have seen in Example 3.1.1, the presence of disjunctive sources in addition requires the use of inequality constraints in query plans. So far, there are no algorithms that generate query plans with inequality constraints. But the differences between conjunctive sources and disjunctive sources are much more extensive. We will see in Example 3.1.2 that the expressive power of datalog, even with inequality, is insufficient to represent query plans that extract all available data from disjunctive sources.

Example 3.1.2 Assume that there are two data sources available which are described by the following view definitions:

$$\begin{aligned} v_1(X) &:- color(X, red) \\ v_1(X) &:- color(X, green) \\ v_1(X) &:- color(X, blue) \\ v_2(X,Y) &:- edge(X,Y) \end{aligned}$$

View v_1 stores vertices that are colored red, green, or blue. View v_2 stores pairs of vertices that are connected by an edge. Assume a user wants to know whether there is a pair of vertices of the same color that are connected by an edge:

$\mathcal{Q}$: $\quad q('yes') :- edge(X,Y), \ color(X,Z), \ color(Y,Z).$

Consider the graphs G_1, G_2, and G_3 in Figure 3.1. All of these graphs are not three-colorable, i.e. for every possible coloring of the vertices with at most three colors, there will be one edge that connects vertices with the same color. Therefore, every graph that contains G_1, G_2, or G_3 as a subgraph contains an edge that connects two vertices with the same color if the vertices in G_1, G_2, and G_3 are colored by at most three colors. Query plans $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ output 'yes' exactly when the input graph contains G_1, G_2, or G_3 respectively as a subgraph and when the vertices in G_1, G_2, and G_3 respectively are colored by at most three colors:

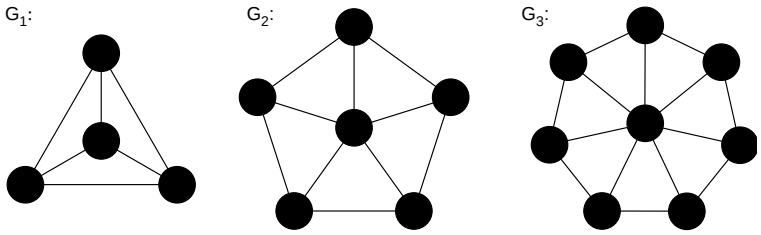

Figure 3.1: *Examples of graphs that are not 3-colorable.*

$$\mathcal{P}_1\text{:}\quad q('yes') :- v_1(X_1),\ v_1(X_2),\ v_1(X_3),\ v_1(Y),\ v_2(X_1,X_2),\ v_2(X_2,X_3),\ v_2(X_3,X_1), v_2(X_1,Y),\ v_2(X_2,Y),\ v_2(X_3,Y)$$

$$\mathcal{P}_2\text{:}\quad q('yes') :- v_1(X_1),\ v_1(X_2),\ v_1(X_3),\ v_1(X_4),\ v_1(X_5),\ v_1(Y), v_2(X_1,X_2),\ v_2(X_2,X_3),\ v_2(X_3,X_4),\ v_2(X_4,X_5),\ v_2(X_5,X_1), v_2(X_1,Y).\ v_2(X_2,Y),\ v_2(X_3,Y),\ v_2(X_4,Y),\ v_2(X_5,Y)$$

$$\mathcal{P}_3\text{:}\quad q('yes') :- v_1(X_1),\ v_1(X_2),\ v_1(X_3),\ v_1(X_4),\ v_1(X_5),\ v_1(X_6),\ v_1(X_7), v_1(Y),\ v_2(X_1,X_2),\ v_2(X_2,X_3),\ v_2(X_3,X_4),\ v_2(X_4,X_5), v_2(X_5,X_6),\ v_2(X_6,X_7), v_2(X_7,X_1),\ v_2(X_1,Y).\ v_2(X_2,Y), v_2(X_3,Y),\ v_2(X_4,Y),\ v_2(X_5,Y),\ v_2(X_6,Y), v_2(X_7,Y)$$

It follows that query plans $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$ are contained in query $\mathcal{Q}$. More generally, *every* query plan that checks that the input graph contains a not three-colorable subgraph, and that all the vertices in the subgraph are colored by at most three colors, is contained in $\mathcal{Q}$.

It is well known that deciding whether a graph is three-colorable is NP-complete [31]. Because the problem of evaluating a datalog program has polynomial data complexity [56], this shows that there is *no* datalog query plan that contains *all* the query plans that are contained in $\mathcal{Q}$. Intuitively, the reason is that for every datalog query plan $\mathcal{P}$ that is contained in $\mathcal{Q}$, an additional conjunctive query that tests for one more not three-colorable graph can be added to create a query plan that is still contained in $\mathcal{Q}$, but that is not contained in $\mathcal{P}$. □

Example 3.1.2 showed that the expressive power of datalog is insufficient to represent query plans that extract all available data from disjunctive sources. In this chapter, we will present a construction of query plans formulated in *disjunctive datalog with inequality* that do guarantee to extract all data. Example 3.1.3 shows the query plan resulting from our construction when applied to the query planning problem in Example 3.1.2.

Example 3.1.3 Let us continue Example 3.1.2. The disjunctive datalog query plan that contains all query plans contained in query $\mathcal{Q}$ is the following:

$$\mathcal{P}\text{:}\quad q('yes') \ :- v_2(X,Y),\ c(X,Z),\ c(Y,Z)$$
$$c(X,red) \vee c(X,green) \vee c(X,blue) \ :- v_1(X)$$