Constraint Deductive Databases

Datalog is a rule-based language that is related to Prolog, which is a popular language for implementing expert systems. Each rule is a statement saying that if some points belong to some relations, then other points must also belong to a defined relation. Each Datalog query contains a Datalog program and an input database.

12.1 Syntax

We divide the set of relation names $\mathcal{R}$ into defined relation names $\mathcal{D}$ and input relation names $\mathcal{I}$. Each Datalog query consists of a finite set of rules of the form:

$$R_0(x_1, \ldots, x_k) := R_1(x_{1,1}, \ldots, x_{1,k_1}), \ldots, R_n(x_{n,1}, \ldots, x_{n,k_n}).$$

where each $R_i$ is either an input relation name or a defined relation name, and the $x$s are either variables or constants.

The relation names $R_0, \ldots, R_n$ are not necessarily distinct. They could also be built-in relation names, such as $\leq, +, \times$, which we will write using the usual infix notation. For example, we will write $x + y = z$ instead of $+(x, y, z)$ for the ternary input relation $+$ that contains all tuples $(a, b, c)$ such that $a$ plus $b$ equals $c$.

The preceding rule is read “$R_0$ is true if $R_1$ and $\ldots$ and $R_n$ are all true.” $R_0$ is called the head and $R_1, \ldots, R_n$ is called the body of the rule.

Example 12.1.1 The tax accountant from Chapter 1 may use the following query $Q_{\text{tax}}$ to find the Social Security number and taxes due for each

person:

\[ \text{Tax\_Due}(s, t) \ := \ \text{Taxrecord}(s, w, i, c), \ \text{Taxable}(i, t), \ w + i + c = \text{inc}. \]

Here \( \text{Taxrecord}, \ \text{Taxable} \in \mathcal{I} \) are input database relations and \( \text{Tax\_Due} \in \mathcal{D} \) is the only defined relation.

Example 12.1.2 Suppose that relation \( \text{Street}(n, x, y) \) contains all combinations of street name \( n \) and locations \( (x, y) \) such that a location belongs to the street. Find the set of streets that are reachable from point \( (x_0, y_0) \).

\[ \text{Reach}(n) \ := \ \text{Street}(n, x_0, y_0). \]
\[ \text{Reach}(n) \ := \ \text{Reach}(m), \ \text{Street}(m, x, y), \ \text{Street}(n, x, y). \]

Here \( \text{Street} \in \mathcal{I} \) and \( \text{Reach} \in \mathcal{D} \). The first rule says that street \( n \) is reachable if it contains the initial point. The second rule says that if \( m \) is reachable and \( m \) and \( n \) intersect on some point, then \( n \) is also reachable.

Next we give several more examples of Datalog queries where the domain of the attributes is scalar values, i.e., \( \mathcal{Z}, \mathcal{Q}, \mathcal{R} \), or \( \mathcal{W} \) as defined in Section 4.4.1.

Example 12.1.3 The following query \( Q_{\text{travel}} \) defines the \( \text{Travel}(x, y, t) \) relation, which is true if it is possible to travel from city \( x \) to city \( y \) in time \( t \):

\[ \text{Travel}(x, y, t) \ := \ \text{Go}(x, 0, y, t). \]
\[ \text{Travel}(x, y, t) \ := \ \text{Travel}(x, z, t_2), \ \text{Go}(z, t_2, y, t). \]

Example 12.1.4 Let relation \( \text{Go2}(x, y, c) \) represent the fact that we can go from city \( x \) to city \( y \) by a direct route in \( c \) units of time. The following query \( Q_{\text{travel2}} \) defines the travel relation, which is true if it is possible to travel from city \( x \) to city \( y \) in time \( t \):

\[ \text{Travel2}(x, y, t) \ := \ \text{Go2}(x, y, c), \ t \geq c. \]
\[ \text{Travel2}(x, y, t) \ := \ \text{Travel2}(x, z, t_2), \ \text{Go2}(z, y, c), \ t \geq t_2 + c. \]

The main difference between this query and that in Example 12.1.3 is that this query puts the constraint in the query while the other puts the constraint in the database.

Example 12.1.5 Suppose a river and its tributaries are subdivided into a set of river segments such that the river flows through each segment in \( c \) hours, where \( c \) is some constant. Let the input relation \( \text{River}(r, x, y) \)