Datalog Without Negation

Jef Wijsen

April 17, 2010

1 Syntax

A *datalog rule* looks like a rule-based conjunctive query; it is an expression of the form:

$$R_0(\vec{u}_0) \leftarrow R_1(\vec{u}_1), \ldots, R_n(\vec{u}_n)$$

where each variable that occurs in the head of the rule, must occur in the body. A *datalog program* P is a finite set of datalog rules.

A relation name is *intensional* if it occurs in the head of some rule of P; otherwise it is *extensional*. The set of extensional relation names in P is denoted $edb(P)$; the set of intensional relation names is denoted $idb(P)$. Finally, $schema(P) = edb(P) \cup idb(P)$. A datalog program defines a mapping from databases over $edb(P)$ to databases over $idb(P)$.[1] Note incidentally that a rule-based conjunctive query is a datalog program consisting of a single rule.

2 Fixpoint Semantics

Let P be a datalog program. The *immediate consequence operator*, denoted T_P, maps a database over $schema(P)$ to a database over $schema(P)$. For a database I over $schema(P)$, $T_P(I)$ is the smallest (under set containment) database over $schema(P)$ such that:

1. $T_P(I) \supseteq \{R(\vec{a}) \in I \mid R \in edb(P)\}$; and
2. for every rule $H \leftarrow B$ of P, $T_P(I) \supseteq \{\nu(H) \mid \nu \text{ is a valuation such that } \nu(B) \subseteq I\}$.

Intuitively, execute every rule once as if it were a conjunctive query. We say that database I is a *fixpoint* of T_P if $T_P(I) = I$.

For positive integer k, we write $T_P^k(I)$ as a shorthand for $\overbrace{T_P(T_P(\ldots T_P(}^{k \text{ times}} I)\ldots)))$.

Example 1

$$\begin{array}{rcl} S(x,y) & \leftarrow & G(x,y) \\ S(x,y) & \leftarrow & G(x,z), S(z,y) \end{array}$$

[1] Note that every database over $edb(P)$ is a database over $schema(P)$.

Let

$$\begin{aligned}
I_0 &= \{G(a,b), G(b,c), G(c,d)\} \ , \\
I_1 &= \{G(a,b), G(b,c), G(c,d), S(a,b), S(b,c), S(c,d)\} \ , \\
I_2 &= \{G(a,b), G(b,c), G(c,d), S(a,b), S(b,c), S(c,d), S(a,c), S(b,d)\} \ , \\
I_3 &= \{G(a,b), G(b,c), G(c,d), S(a,b), S(b,c), S(c,d), S(a,c), S(b,d), S(a,d)\} \ , \\
J_1 &= I_3 \cup \{G(a,1), G(1,f), S(a,1), S(1,f), S(a,f)\} \ , \\
J_2 &= I_3 \cup \{G(a,2), G(2,f), S(a,2), S(2,f), S(a,f)\}.
\end{aligned}$$

Then, $T_P(I_0) = I_1$, $T_P(I_1) = I_2$, $T_P(I_2) = I_3$, $T_P(I_3) = I_3$, so I_3 is a fixpoint.

J_1 and J_2 are also fixpoints, because $T_P(J_1) = J_1$ and $T_P(J_2) = J_2$. Notice also that $J_1 \cap J_2 = I_3 \cup \{S(a,f)\}$ is **not** a fixpoint since $T_P(J_1 \cap J_2) = I_3 \neq J_1 \cap J_2$.

Lemma 1 (Monotonicity) *Let I and J be databases over schema(P). If $I \subseteq J$, then $T_P(I) \subseteq T_P(J)$.*

Proof. Easy. □

Note incidentally that it is **not** generally true that $J \subseteq T_P(J)$. For example, for $J_3 = \{S(a,b)\}$, we have $J_3 \nsubseteq T_P(J_3) = \{\}$.

Lemma 2 *For each datalog program P and database I over edb(P), T_P has a minimum fixpoint containing I.*

Proof. Consider the sequence:

$$I, T_P(I), T_P^2(I), T_P^3(I), \ldots$$

Since all relation names in I are extensional, $I \subseteq T_P(I)$. By Lemma 1, $T_P(I) \subseteq T_P(T_P(I))$. By Lemma 1, $T_P(T_P(I)) \subseteq T_P(T_P(T_P(I)))$. And so on. It follows

$$I \subseteq T_P(I) \subseteq T_P^2(I) \subseteq T_P^3(I) \subseteq \ldots$$

All constants that occur in any database of this sequence occur in I or P. Only finitely many atoms can be constructed using these constants. Thus, the sequence must reach a fixpoint after a finite number N of steps (N depends on the size of I).

Let J be an arbitrary fixpoint such that $I \subseteq J$. By Lemma 1, $T_P(I) \subseteq T_P(J)$. Since J is a fixpoint, $T_P(J) = J$, hence $T_P(I) \subseteq J$. By Lemma 1, $T_P(T_P(I)) \subseteq T_P(J) = J$. And so on. It follows $T_P^N(I) \subseteq J$.

Consequently, every fixpoint that contains I, must necessarily contain the fixpoint $T_P^N(I)$. Thus, $T_P^N(I)$ is the minimum fixpoint. □

Let I be a database over *edb*(P). The *semantics* of P on input I, denoted $P(I)$, is the minimum fixpoint of T_P that contains I.

3 Program Dependency Graph

The vertexes of the *dependency graph* of datalog program P are the elements of *idb*(P). There is an edge from relation name R to relation name S if there is a rule in which R

occurs in the head and S in the body. If the dependency graph is acyclic, then the program is nonrecursive. But even a program with a cyclic dependency graph can be essentially nonrecursive [1].

$$\begin{array}{rcl} \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Trendy}(x), \mathsf{Buys}(z,y) \\ \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Likes}(x,y) \end{array}$$

(person x buys product y if x likes y or if x is trendy and someone buys y). The program is equivalent (why?) to the following:

$$\begin{array}{rcl} \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Trendy}(x), \mathsf{Likes}(z,y) \\ \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Likes}(x,y) \end{array}$$

The following program is inherently recursive:

$$\begin{array}{rcl} \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Knows}(x,z), \mathsf{Buys}(z,y) \\ \mathsf{Buys}(x,y) & \leftarrow & \mathsf{Likes}(x,y) \end{array}$$

(x buys y if x likes y or if x knows someone who bought y).

4 Exercises

1. From [1]. We are given two directed graphs G_{black} and G_{white} over the same set V of vertexes, represented as binary relations. Write a datalog program P that computes the set of pairs (a, b) of vertexes such that there exists a path from a to b where black and white edges alternate, starting with a white edge.

2. Given a directed graph G represented as a binary relation, write a datalog program that detects whether there is a cycle of odd length. A cycle of length n ($n \geq 1$) is a sequence of (not necessarily distinct) vertexes $a_0, a_1, \ldots, a_n$ where $a_n = a_0$ and for each $i \in \{0, 1, \ldots, n-1\}$, there is an edge from a_i to a_{i+1}.

3. Assume a single extensional relation name R. Show that the property that the number of elements in a database over $\{R\}$ is even is not definable in datalog.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.