SEDATALOG: A SET EXTENSION OF DATALOG

Qing Zhou
The Software Institute of Zhongshan University
Guangzhou, Guangdong 510275, P.R.China
lnszq@zulink.zsu.edu.cn

Ligong Long
The Software Institute of Zhongshan University
Guangzhou, Guangdong 510275, P.R.China
longligong@tom.com

Abstract In this paper we propose an extension, SEDatalog, of Datalog so that sets can be naturally constructed in logic programming. In SEDatalog, sets can be defined by statements so it has a strong capability in creating sets. Three deductive rules are also introduced in this paper, which make SEDatalog strong in deductions and programming even when sets are involved in deductions. The syntactical description and the semantical interpretation of SEDatalog are comprehensively discussed in detail. The soundness and completeness theorem of SEDatalog is proved, which provides a solid foundation of SEDatalog.

Keywords: The order of a set, The order of predicate, n-th order set

1. Introduction

In this paper we propose an extension, SEDatalog, of Datalog so that sets can be constructed in logic programming. As the extension is entirely based on what is common in every logic programming language, the extension could apply to Prolog and other logic programming languages almost without any modification.

In SEDatalog, we can define a set A such that \( A = \{ x : p(x) \} \) for every formula. By defining sets by statements we can not only construct finite sets but also infinite sets or more complicated sets such as a set of sets with certain properties. Also this is the way we define sets when we are working in mathematics, or in other areas. This definition, if without proper restrictions, would involve confusions among set construction levels and would lead to Russell’s paradox. To avoid this, we restrict the elements to be contained in a set to only
those which already exist, thus helping achieve clear indication of a set hierarchy to be constructed in \textit{SEDatalog}. As hierarchies constructed this way are in line with the underlying principles of the axiomatic set theory i.e. \textit{ZF}, avoidance of such paradoxes as "the set of all sets" or "a set containing itself" can be assured.

For this purpose we need an "order" for every set which indicates the level in which the set is constructed. Then statements in \textit{Datalog} can be used to define sets. We consider that these sets are of the first order. Using these sets and individuals in \textit{Datalog}, which are of order 0, we can construct a group of statements, which are not in \textit{Datalog} in general, by which second order sets can be defined. Continuing by this way, we may have all the sets constructed. To ensure avoidance of any possible confusion in the construction, we have to give an order to every statement in \textit{SEDatalog}, too. This means that every predicate in \textit{SEDatalog} can only allow those sets with order less than a constant integer (the order of the predicate) as its variables. So every predicate in \textit{SEDatalog} is a partial predicate, i.e. its domain can not contain any set which has the order larger than or equal to the order of the predicate. This takes care of the problem mentioned in the last paragraph.

2. The Syntactical Description Of \textit{SEDatalog}

The \textit{alphabet} of \textit{SEDatalog} consists of variables, constants, predicates and connectives. Each constant, variable and predicate is assigned an unique integer to it. A \textit{n}-th order term is a constant or variable which has been assigned \textit{n} to it. The order of a predicate is \textit{n}, if it is assigned \textit{n}. We will use \textit{o(t)} to indicate the order of \textit{t} if \textit{t} is a variable, a constant or a predicate.

\textbf{Definition 1} \textit{Formulas and their orders are defined as follows:}

(1) If \textit{p} is a \textit{k}-order \textit{n}-ary predicate symbol, and for every \textit{i}, \textit{1} \leq \textit{i} \leq \textit{n}, \textit{t[i]} is a term with \textit{o(t[i])} < \textit{k}, then \textit{p(t[1], \ldots , t[n])} is an atom formula and \textit{o(p(t[1], \ldots , t[n]))} = \textit{k};

(2) If \textit{t[1]} and \textit{t[2]} are terms, and \textit{o(t[1])} < \textit{o(t[2])}, then \textit{t[1] \in t[2]} is an atom formula and \textit{o(t[1] \in t[2])} = \textit{o(t[2])} + 1;

(3) If \textit{t[1]} and \textit{t[2]} are terms, and \textit{0 < o(t[1]) \leq o(t[2])}, then \textit{t[1] \subseteq t[2]} is an atom formula and \textit{o(t[1] \subseteq t[2])} = \textit{o(t[2])} + 1;

(4) If \textit{t[1]} and \textit{t[2]} are terms, and \textit{o(t[1])} = \textit{o(t[2])}, then \textit{t[1] = t[2]} is an atom formula and \textit{o(t[1] = t[2])} = \textit{o(t[2])} + 1.

(5) If \textit{F, G} are formulas, then \textit{F \land G}, \textit{F \lor G} are formulas and \textit{o(F \land G)} = max(\{\textit{o(F)}, \textit{o(G)}\}), \textit{o(F \lor G)} = max(\{\textit{o(F)}, \textit{o(G)}\}).

An atom with no free variables is called a \textit{ground atom}. A \textit{closed formula} is a formula with no free variables. An atom is called a \textit{literal} in \textit{SEDatalog}.

\textbf{Remark} In the above definition, \textit{F(x) \lor G(x)} might not be well defined from Definition 1 for some \textit{x} with order \textit{o(F(x))} \leq \textit{o(x)} < \textit{o(G(x))} when
\( o(F(x)) < o(G(x)) \). To let the definition make sense, we consider it as \( F'(x) \lor G(x) \), where \( F'(x) \leftrightarrow F(x) \) and \( o(F'(x)) = o(G(x)) \), i.e. the domain of \( F'(x) \) is extended to all terms with the order up to \( o(G(x)) \), although \( F'(x) \leftrightarrow F(x) \). The same treatment is made for \( F(x) \land G(x) \).

**Definition 2** For each formula \( F(x) \) with only one free variable \( x \), the set defined by \( F(x) \) is a constant \( C_{F(x)} \in C_n \) with \( n = o(F(x)) \) such that for all \( t, t \in C_{F(x)} \) if and only if \( F(t) \).

We often write \( C_{F(x)} = \{ t : F(t) \} \) to indicate the definition of \( C_{F(x)} \).

Directly from our definitions, it is not hard to see that for every formula \( F(x) \), \( o(C_{F(x)}) \) is a finite integer.

Then we turn to the deduction rules of \( SEDatalog \). As an extension of Datalog, we first extend the deduction rule of Datalog to be used on statements which contain sets as their variables. We call it "the ordinary rule". In addition to this ordinary rule, we add two more deduction rules, "the universal rule" and "the existential rule".

**Definition 3** A rule of \( SEDatalog \) is of the form \( H : -A_1, ... , A_n \) where \( n \geq 0 \). The left hand side of \( - \) is a literal, called the head of the rule, while the right hand side is a conjunction of literals, called the body of the rule.

A fact is a special rule, whose head is a ground literal and whose body is empty.

For convenience, we use the notation \( vars(T) \) to indicate all variables occurring in \( T \), where \( T \) is a term or a formula. Then \( vars \) is a mapping from the set of terms and formulas to the power set of \( Var \). We use \( H(y) \) to represent a literal with variable \( y \) and \( A(x) \) to represent a literal with variable \( x \). Now we give the following three rules, which will be called safe. They are the basic deductive rules in \( SEDatalog \):

1. **Ordinary rule** is of the form

   \[ H : -O \ A_1, ... , A_n \]

   where \( n > 0 \), \( vars(H) \subseteq vars(A_1 \land ... \land A_n) \) and \( o(H) = o(A_1 \land ... \land A_n) \).

   The informal semantics of this rule is to mean that "for every assignment to each variable, if \( A_1, ... , A_n \) are true, then \( H \) is true".

2. **Universal rule** is of the form

   \[ H(y) : -U \ A_1(x), ... , A_m(x), A_{m+1}, ... , A_n \]

   where \( vars(H(y)) = \{ y \} \subseteq vars(A_1(x) \land ... \land A_m(x) \land A_{m+1} \land ... \land A_n) \), \( y \neq x \), \( o(H(y)) > o(A_1(x) \land ... \land A_m(x) \land A_{m+1} \land ... \land A_n) \) and \( y \subseteq \{ x : A_1(x) \land ... \land A_m(x) \land A_{m+1} \land ... \land A_n \} \).
The informal semantics of this rule is to mean that “if every element $x$ in $y$ has properties $A_1(x), \ldots, A_m(x)$, then $y$ has the property $H$.”

In this case, $y$ is called the **universal property** $H$.

3. **Existential rule** is of the form

$$H(y) : = E_A \neg A_1(x_1, \ldots, x_k), \ldots, A_m(x_1, \ldots, x_k), A_{m+1}, \ldots, A_n$$

where $\text{vars}(H(y)) = \{y\} \subseteq \text{vars}(A_1(x_1, \ldots, x_k) \land \ldots \land A_m(x_1, \ldots, x_k) \land A_{m+1} \land \ldots \land A_n - \{x_1, \ldots, x_k\}, y \neq x_1, \ldots, y \neq x_k, o(H(y)) > o(A_1(x_1, \ldots, x_k) \land \ldots \land A_m(x_1, \ldots, x_k) \land A_{m+1} \land \ldots \land A_n)$ and $x_1 \in y, \ldots, x_k \in y$.

The informal semantics of this rule is to mean that “if some elements $x_1, \ldots, x_k$ in $y$ have properties $A_1(x_1, \ldots, x_k), \ldots, A_m(x_1, \ldots, x_k)$, then $y$ has the property $H$.”

In this case, $y$ is called the **existential property** $H$.

**Definition 4** A SEDatalog program is a finite sequence of rules.

**Definition 5** A substitution $\theta$ is a finite set of the form $\{x_1/t_1, \ldots, x_n/t_n\}$, where $x_1, \ldots, x_n$ are distinct variables and each $t_i$ is a term such that $x_i \neq t_i$, and $o(t_i) \leq o(x_i)$.

The set of variables $\{x_1, \ldots, x_n\}$ is called the domain of $\theta$.

If $T$ is a term, a literal or a rule then $T\theta$ denotes the corresponding item obtained from $T$ by simultaneously replacing each $x_i$ that occurs in $T$ by the corresponding term $t_i$, if $x_i/t_i$ is an element of $\theta$.

If each $t_i$ is ground, then $\theta$ is a ground substitution.

3. **The Semantical Interpretations of SEDatalog**

Let $M_0 = (\mathcal{V}, \mathcal{P}_0, T_0)$ be an interpretation of Datalog, where $\mathcal{V}$ is the universe of $M_0$; $\mathcal{P}_0$ is the set of the interpretations of predicate symbols of Datalog; $T_0$ is the set of interpretations of those ground atoms of Datalog which are interpreted as true; respectively. We define $\mathcal{U}_0 = \mathcal{V}$, and $\mathcal{U}_n = \mathcal{U}_{n-1} \cup \mathcal{P}_n$ where $\mathcal{P}_n = \{A : A \subseteq \mathcal{U}_{n-1}\}$. Then we give the full description of the interpretation of SEDatalog as follows:

An interpretation $M$ of SEDatalog is a tuple: $M = (\mathcal{U}, \mathcal{P}, T)$, here $\mathcal{U} = \bigcup_{k=0}^{\infty} \mathcal{U}_k$ is the universe of $M$; $\mathcal{P}$ is the set of the interpretations of predicate symbols; $T$ is the set of interpretations of those ground literals which are interpreted as true, respectively, such that:

1. Each $n$-ary predicate symbol $q_{(k)}$ is interpreted as a predicate $q_M \subseteq \mathcal{P}$, i.e. $q_M \subseteq \mathcal{U}_k^n$, and $q_{(1)}$ is interpreted as a predicate $q_M \subseteq \mathcal{P}_0$.

Especially, $\in_{(m)}$, $\subseteq_{(m)}$ and $\Rightarrow_{(m)}$ are interpreted as the usual meanings.
(2) Each constant c in C_n (n > 0) is interpreted as an object (set) M(c) of 
U_n; and each constant c in C_0 is interpreted as same as in M_0, i.e. an object 
(individual) M(c) of U_0.

(3) A ground atom q(t[1], ..., t[n]) is interpreted as M(q(t[1], ..., t[n])) \iff
q_M(M(t[1]), ..., M(t[n])), for more complicated formulas such as F(x) \lor
G(x) and F(x) \land G(x), their truth values are interpreted as usual;

(4) Now we define T as follows:
T_0 is as same as the T_0 in M_0;
T_k \subseteq T_{k-1} \cup \{q_M(M(t[1]), ..., M(t[n])) : q \in q_k(t[1], ..., t[n]) \text{ is a ground}
\text{ literal}\} \text{ which satisfies } T_{k-1} \subseteq T_k \text{ and:}
\text{i) } M(c \in C_{A(x)}) \in T_k \text{ if and only if } M(A(c)) \in T_{k-1}.
\text{ii) } M(C_F(x) \subseteq C_{G(x)}) \in T_k \text{ if and only if for all } x \in U, M((F)(x)) \in T
\text{ implies that } M((G)(x)) \in T_k.

and finally, let \( T = \bigcup_{k=1}^{\infty} T_k \).

With the interpretation of SEDatalog described above, our next job is to
give the description of the model of a SEDatalog program:
Let P be a program. An interpretation \( M = \langle U, P, T \rangle \) is a model of P if
and only if

(1) If A is a fact in P, then \( M(A) \in T \);

(2) If \( r : H : -O A_1, ..., A_n \) is an ordinary rule in P, then for each ground
and legal substitution \( \theta \) with \( \text{domain}(\theta) \supseteq \text{vars}(r) \), if \( M(A_1 \theta) \in T, ..., M(A_n \theta) \in T \),
then \( M(H \theta) \in T \);

(3) If \( r : H(y) : -U A_1(x), ..., A_m(x), A_{m+1}, ..., A_n \) is a universal rule in
P, then for each ground and legal substitution \( \theta \) with \( \text{domain}(\theta) = \text{vars}(r) - \{x\} \), if \( M(A_{m+1} \theta) \in T, ..., M(A_n \theta) \in T \), and \( M(y \theta) \subseteq C_{A_1(x) \theta} \in T, ..., M(y \theta) \subseteq C_{A_m(x) \theta} \in T \),
then \( M(H(y) \theta) \in T \), here \( M(y \theta) \subseteq C_{A_1(x) \theta} \in T, \ 1 \leq i \leq m; \)

(4) If \( r : H(y) : -E A_1(x_1, ..., x_k), ..., A_m(x_1, ..., x_k), A_{m+1}, ..., A_n \) is an existential rule in P, then for each ground and legal substitution \( \theta \) with
\( \text{domain}(\theta) = \text{vars}(r) \), if \( M(A_1(x_1, ..., x_k) \theta) \in T, ..., M(A_m(x_1, ..., x_k) \theta) \in T \), \( M(A_{m+1} \theta) \in T, ..., M(A_n \theta) \in T \), and \( M((x_1 \in y) \theta) \in T, ..., M((x_k \in y) \theta) \in T \) then \( M(H(y) \theta) \in T \).

Definition 6 Let A be a ground literal. An interpretation \( M = \langle U, P, T \rangle \)
is a model of A if and only if \( M(A) \in T \).

Now we are going to discuss the soundness and completeness theorem of
SE Datalog. Before that let us introduce some related notions first.

Definition 7 A ground literal A is a consequence of a SEDatalog program P
denoted by \( P \models A \) if and only if each model M of P is also a model
of A.
DEFINITION 8 A ground literal $A$ is inferred from a SEDatalog program $P$ (denoted by $P \vdash A$) is defined as follows:

(1) If $A = H$ and $H$ is a fact in $P$, then $P \vdash A$;

(2) If there exists an ordinary rule $r : H : \neg O \ A_1, \ldots, A_n \in P$ and a ground and legal substitution $\theta$, where $\text{domain}(\theta) = \text{vars}(r)$, such that $A = H\theta$ and $P \vdash A_1\theta, \ldots, P \vdash A_n\theta$, then $P \vdash A$;

(3) If there exists a universal rule $r : H(y) : \forall x_1, \ldots, x_k, A_m(x), A_{m+1}, \ldots, A_n \in P$ and a ground and legal substitution $\theta$, where $\text{domain}(\theta) = \text{vars}(r) \setminus \{x\}$, such that $A = H(y)\theta$ and $P \vdash A_{m+1}\theta, \ldots, P \vdash A_n\theta$, and $P \vdash y\theta \subseteq C_{A_1}(x_\theta), \ldots, P \vdash y\theta \subseteq C_{A_m}(x_\theta)$, then $P \vdash A$, here $P \vdash y\theta \subseteq C_{A_i}(x_\theta)$, $1 \leq i \leq m$;

(4) If there exists an existential rule $r : H(y) : \exists x_1, \ldots, x_k, x_1, A_m(x), A_{m+1}, \ldots, A_n \in P$ and a ground and legal substitution $\theta$, where $\text{domain}(\theta) = \text{vars}(r)$, such that $A = H(y)\theta$ and $P \vdash A_1(x_1, \ldots, x_k)\theta, \ldots, A_n\theta$, and $P \vdash x_1\theta \in y\theta$, $\ldots, P \vdash x_k\theta \in y\theta$, then $P \vdash A$.

Let $\text{infer}(P) = \{A : P \vdash A\}$ and $\text{cons}(P) = \{A : P \models A\}$. It is easy to show that, $\text{cons}(P) = \bigcap_M \{A : M = (\mathcal{U}, \mathcal{P}, T) \text{ is a model of } P \text{ and } M(A) \in T\}$. Then we can prove The Soundness and Completeness Theorem:

THEOREM 9 $\text{infer}(P) = \text{cons}(P)$.

References
Chimenti, D., Gamboa, R., Krishnamurthy, R., Naqvi, S., Tsur, S., Zaniolo, C.; The LDL System Prototype, IEEE Transactions on Knowledge and Data Engineering, 2(1), pp.76-90, 1990.