$SEDATALOG$: A SET EXTENSION OF $DATALOG$

Qing Zhou
The Software Institute of Zhongshan University
Guangzhou, Guangdong 510275, P.R.China
lnszq@zsulink.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 $SEDatalog$. As hierarchies constructed this way are in line with the underlying principles of the axiomatic set theory i.e. 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 $Datalog$ can be used to define sets. We consider that these sets are of the first order. Using these sets and individuals in $Datalog$, which are of order 0, we can construct a group of statements, which are not in $Datalog$ 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 $SEDatalog$, too. This means that every predicate in $SEDatalog$ can only allow those sets with order less than a constant integer (the order of the predicate) as its variables. So every predicate in $SEDatalog$ 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 $SEDatalog$

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

DEFINITION 1 ***Formulas*** *and their orders are defined as follows:*

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

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

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

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

(5) If F, G are formulas, then $F \wedge G$, $F \vee G$ are formulas and $o(F \wedge G) = max(\{o(F), o(G)\})$, $o(F \vee G) = max(\{o(F), o(G)\})$.

*An atom with no free variables is called a **ground atom**. A **closed formula** is a formula with no free variables. An atom is called a literal in $SEDatalog$.*

Remark In the above definition, $F(x) \vee G(x)$ might not be well defined from Definition 1 for some x with order $o(F(x)) \leq o(x) < o(G(x))$ when

$o(F(x)) < o(G(x))$. To let the definition make sense, we consider it as $F'(x) \vee G(x)$, 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 $F(x) \wedge G(x)$.

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 $H : -A_1, \dots, A_n$ 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

$$H : -_O A_1, \dots, A_n$$

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

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

(2) **Universal rule** is of the form

$$H(y) : -_U A_1(x), \dots, A_m(x), A_{m+1}, \dots, A_n$$

where $vars(H(y)) - \{y\} \subseteq vars(A_1(x) \wedge \dots \wedge A_m(x) \wedge A_{m+1} \wedge \dots \wedge A_n) - \{x\}$, $y \neq x$, $o(H(y)) > o(A_1(x) \wedge \dots \wedge A_m(x) \wedge A_{m+1} \wedge \dots \wedge A_n)$ and $y \subseteq \{x : A_1(x) \wedge \dots \wedge A_m(x) \wedge A_{m+1} \wedge \dots \wedge A_n\}$.

The informal semantics of this rule is to mean that "if every element x in y has properties $A_1(x)$, ... , $A_m(x)$, then y has the property H".

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

(3) **Existential rule** is of the form

$$H(y) :-_E \quad A_1(x_1, ..., x_k), ..., A_m(x_1, ..., x_k), A_{m+1}, ..., A_n$$

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

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

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

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

DEFINITION 5 *A **substitution** θ is a finite set of the form $\{x_1/t_1, \dots, x_n/t_n\}$, where $x_1, \dots, x_n$ 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 $\{x_1, \dots, x_n\}$ is called the **domain** of θ.*

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 θ.

*If each t_i is ground, then θ is a **ground substitution**.*

3. The Semantical Interpretations of $SEDatalog$

Let $M_0 = \langle \mathcal{V}, \mathcal{P}_0, \mathcal{T}_0 \rangle$ 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 $Datalog$; $\mathcal{T}_0$ 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 $\mathcal{U}_n = \mathcal{U}_{n-1} \cup \wp(\mathcal{U}_{n-1})$ where $\wp(\mathcal{U}_{n-1})$ is the power set of $\mathcal{U}_{n-1}$, i.e. $\wp(\mathcal{U}_{n-1}) = \{A : A \subseteq \mathcal{U}_{n-1}\}$. Then we give the full description of the interpretation of $SEDatalog$ as follows:

An interpretation M of $SEDatalog$ is a tuple: $M = \langle \mathcal{U}, \mathcal{P}, \mathcal{T} \rangle$, 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; $\mathcal{T}$ is the set of interpretations of those ground literals which are interpreted as true, respectively, such that:

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

Especially, $\in_{\langle m \rangle}$, $\subseteq_{\langle m \rangle}$ and $=_{\langle m \rangle}$ are interpreted as the usual meanings.

(2) Each constant c in C_n $(n > 0)$ is interpreted as an object (set) $M(c)$ of $\mathcal{U}_n$; and each constant c in C_0 is interpreted as same as in M_0, i.e. an object (individual) $M(c)$ of $\mathcal{U}_0$;

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

(4) Now we define $\mathcal{T}$ as follows:

$\mathcal{T}_0$ is as same as the $\mathcal{T}_0$ in M_0;

$\mathcal{T}_k \subseteq \mathcal{T}_{k-1} \cup \{q_M(M(t[1]), \ldots, M(t[n])) : q_{<k>}(t[1], \ldots, t[n])$ is a ground literal$\}$ which satisfies $\mathcal{T}_{k-1} \subset \mathcal{T}_k$ and:

i) $M(c \in C_{A(x)})) \in \mathcal{T}_k$ if and only if $M(A(c)) \in \mathcal{T}_{k-1}$.

ii) $M(C_{F(x)} \subseteq C_{G(x)}) \in \mathcal{T}_k$ if and only if for all $x \in \mathcal{U}$, $M(F)(x) \in \mathcal{T}$ implies that $M(G)(x) \in \mathcal{T}_k$.

and finally, let $\mathcal{T} = \bigcup_{k=1}^{\infty} \mathcal{T}_k$.

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 $M = \langle \mathcal{U}, \mathcal{P}, \mathcal{T} \rangle$ is a *model* of P if and only if

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

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

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

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

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

Now we are going to discuss the soundness and completeness theorem of $SEDatalog$. 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 $r : H :-_O A_1, \dots, A_n$ in P and a ground and legal substitution θ, where $domain(\theta) = vars(r)$, such that $A = H\theta$ and $P \vdash A_1\theta, \dots, P \vdash A_n\theta$, then $P \vdash A$;

(3) If there exists a universal rule $r : H(y) :-_U A_1(x), \dots, A_m(x), A_{m+1}, \dots, A_n$ in P and a ground and legal substitution θ, where $domain(\theta) = vars(r) - \{x\}$, such that $A = H(y)\theta$ and $P \vdash A_{m+1}\theta, \dots, P \vdash A_n\theta$, and $P \vdash y\theta \subseteq C_{A_1(x)\theta}, \dots, P \vdash y\theta \subseteq C_{A_m(x)\theta}$, 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 $r : H(y) :-_E A_1(x_1, \dots, x_k), \dots, A_m(x_1, \dots, x_k), A_{m+1}, \dots, A_n$ in P and a ground and legal substitution θ, where $domain(\theta) = vars(r)$, such that $A = H(y)\theta$ and $P \vdash A_1(x_1, \dots, x_k)\theta, \dots, P \vdash A_m(x_1, \dots, x_k)\theta$, $P \vdash A_{m+1}\theta, \dots, P \vdash A_n\theta$, $P \vdash x_1\theta \in y\theta, \dots, P \vdash x_k\theta \in y\theta$, then $P \vdash A$.

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

THEOREM 9 *$infer(P) = cons(P)$.*

References

Abiteboul, S.,Grumbach, S.,: COL: A Logic-based Language for Complex Objects, ACM TODS, 16(1), *pp.*1-30, 1991.

Ceri, S., Gottlob, G.,Tanca, L.,: Logic Programming and Databases, Springer Verlag, 1990.

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.

Dovier, A., Omodeo, E. G., Pontelli, E., Rossi, G.,: {log}: A Language for Programming in Logic with Finite Sets, J. of Logic Programming, 28(1), *pp.*1-44, 1996.

Jana, D.,: Semantics of Subset-Logic Languages, Ph.D. Dissertation, Department of Computer Science, SUNY-Buffalo, 1994.

Jayaraman, B.,: The SuRE Programming Framework, TR 91-011, Department of Computer Science, SUNY-Buffalo, August 1991.

Jayaraman, B.,Jana, D.,: Set Constructors, Finite Sets, and Logical Semantics, J. of Logic Programming, 38, *pp.*55-77, 1999.

Kuper, G. M.,: Logic Programming with Sets, J. of Computer and System Science, 41(1), *pp.*44-64, 1990.

Liu, M.,: Relationlog: A Typed Extension to *Datalog* with Sets and Tuples, J. of Logic Programming, 36, *pp.*271-299, 1998.

Lloyd, J. W.,: Foundations of Logic Programming, Springer Verlag, 1987.

Moon, K.,: Implementation of Subset Logic Languages, Ph.D. Dissertation, Department of Computer Science, SUNY-Buffalo, February 1995.

Osorio, M.,: Semantics of Logic Programs with Sets, Ph.D. Dissertation, Department of Computer Science, SUNY-Buffalo, 1995.