Decidable Order-Sorted Logic Programming for Ontologies and Rules with Argument Restructuring

Ken Kaneiwa[1] and Philip H.P. Nguyen[2]

[1] National Institute of Information and Communications Technology, Japan
kaneiwa@nict.go.jp
[2] Department of Justice, Government of South Australia
philip.nguyen@sa.gov.au

Abstract. This paper presents a decidable fragment for combining ontologies and rules in order-sorted logic programming. We describe order-sorted logic programming with sort, predicate, and meta-predicate hierarchies for deriving predicate and meta-predicate assertions. Meta-level predicates (predicates of predicates) are useful for representing relationships between predicate formulas, and further, they conceptually yield a hierarchy similar to the hierarchies of sorts and predicates. By extending the order-sorted Horn-clause calculus, we develop a query-answering system that can answer queries such as atoms and meta-atoms generalized by containing predicate variables. We show that the expressive query-answering system computes every generalized query in single exponential time, i.e., the complexity of our query system is equal to that of DATALOG.

1 Introduction

In the Semantic Web context, conceptual knowledge representation and reasoning [23] have been studied for modeling ontologies in OWL (Web Ontology Language) [20]. In general, concepts are interpreted by sets of individuals, and concept hierarchies are constructed by subsumption (similar to IS-A relations). The formal semantics and reasoning of concept description languages are guaranteed by logical formalizations. Order-sorted logic [4, 22, 14] (as first-order logic with partially ordered sorts) provides sorts and sort hierarchy that represent concepts and their concept hierarchy, respectively. A predicate hierarchy, which is an extension of the sort hierarchy, consists of n-ary predicates that are conceptually related to each other. In [13], order-sorted logic programming was extended by introducing such a predicate hierarchy. Furthermore, the conceptual structure theory [19] was extended to include relation types and their type hierarchy for building complex ontologies.

Meta-level predicates (predicates of predicates) are expressions that can be employed for representing relationships between facts in knowledge bases. Similar to hierarchies of sorts and predicates, these meta-predicates can be used to conceptually construct a hierarchy, e.g., the meta-predicate *causes* implies the super

meta-predicate *likelyCauses*. In the OWL family, meta-concepts are supported by OWL-Full (the most expressive language of OWL). The semantics of modeling for meta-concepts and the undecidability of meta-modeling in OWL-Full have been discussed in [18]. Further, higher-order expressions may conceptually yield a hierarchy when they are named using natural language words. However, order-sorted (or typed) logic programming lacks representation and reasoning for such meta-level predicates.

Alternatively, logic programming provides formal semantics and decidable reasoning services for RuleML (Rule Makeup Language) [1] in the Semantic Web. This language is a restricted fragment of first-order logic, and its complexities [5] have been studied in the area of automated deduction. It is known that full logic programming is undecidable, but function-free logic programming (i.e., DATALOG) is EXPTIME-complete with respect to the length of a program. In addition, non-recursive logic programming is NEXPTIME-complete, even if it includes functions.

However, SWRL (Semantic Web Rule Language) [11], a combination of OWL and RuleML, leads to undecidable reasoning between ontologies and rules (as shown in [10]). Several decidable fragments for combining ontologies and rules, such as DL-safe [9, 21], DLP (Description Logic Programs) [7], and the rule-based language ELP [16] (related to the tractable language profile OWL 2 EL [2]), have been proposed by restricting the expressive power of rules. Similar to the approaches adopted in past studies, in order to make ontologies and rules in logic programing expressive and at the same time retain decidability, the logic programming language must be carefully extended for conceptual knowledge representation and reasoning. HILOG [3], which involves the second-order expression of meta-level predicates, has been developed as a decidable higher-order language for logic programming, and it may be more complex than the EXPTIME complexity of DATALOG. Unfortunately, in most cases, higher-order logic programming [12] makes reasoning increasingly difficult because complex structures of higher-order terms need to be treated.

To overcome the aforementioned difficulties related to expressiveness and complexity, we introduce meta-predicates and their hierarchy in a restricted and decidable fragment for combining ontologies and rules. In particular, we formalize an order-sorted logic programming language with a meta-predicate hierarchy. As a result, three kinds of hierarchies (sort hierarchy, predicate hierarchy, and meta-predicate hierarchy) are included in the syntax and semantics of the sorted logic programming. We develop the order-sorted Horn-clause calculus [8], which serves as a sorted deductive system, for reasoning on concept hierarchies where predicate assertions and relationships among the assertions are derived. This calculus terminates if the knowledge bases are function free (i.e., only constants such as 0-ary functions are allowed). Using this calculus, we develop a query-answering system that is extended by generalizing queries with predicate variables. Our result shows that the complexity of the expressive query system (even for meta-predicates and predicate variables) is single exponential time and equal to the complexity of DATALOG.

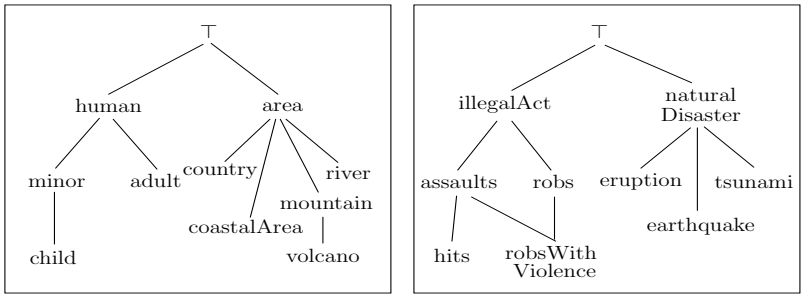

Fig. 1. Sort and predicate hierarchies

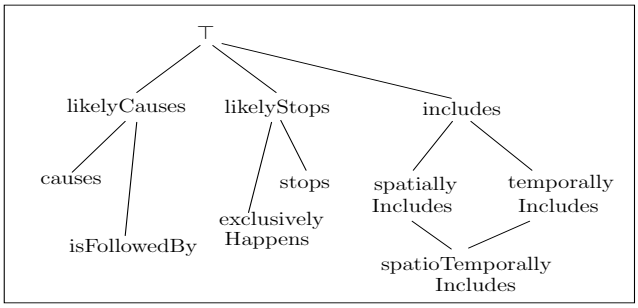

Fig. 2. A meta-predicate hierarchy

2 Motivating Examples

We now present some examples of hierarchies in a query-answering system. Given the sort, predicate, and meta-predicate hierarchies in Figs. 1 and 2, we consider logical reasoning using a knowledge base for the hierarchies. If the fact `hits(tom:minor,john:adult)` is valid, then the super predicate `illegalAct` can be derived in the predicate hierarchy (shown in Fig. 1).

```
hits(tom:minor,john:adult)
?-illegalAct(x:human)
yes
x=tom:minor
```

In this derivation, the second argument `john:adult` is deleted if the argument structure of the predicate `illegalAct` lacks the second argument of the predicate `hits`. Conceptually, both the name and argument structure of `illegalAct` are more abstract than `hits` in the predicate hierarchy.

Moreover, we employ meta-predicates (predicates of predicates) to express relationships among facts in the knowledge base. For example, the meta-predicate `isFollowedBy` is used to indicate that a tsunami in Phuket c_2 occurred after the earthquake in Indonesia c_1.

```
isFollowedBy(earthquake(c1:country),tsunami(c2:coastalArea))
?-likelyCauses(earthquake(c1:country),tsunami(c2:coastalArea))
yes
```

If the relationship between the two facts is valid, the super meta-predicate `likelyCauses` can be inferred in the meta-predicate hierarchy (shown in Fig. 2).

Additionally, the fact `earthquake(c1:country)` is derived from this relationship because it is the first argument of the meta-predicate `isFollowedBy`.

```
?-earthquake(c1:country)
yes
```

The assumption underlying the abovementioned derivation is that the meta-predicate points to the occurrence of facts in addition to indicating the existence of a relationship between them.

An expression with predicate variables X and Y is used to query the validity of a causal relationship between two natural disasters as follows.

```
?-likelyCauses(X:naturalDisaster(x:area),
                          Y:naturalDisaster(y:area))
yes
X=earthquake, x=c1:country, Y=tsunami, y=c2:coastalArea
```

Using the meta-predicate hierarchy, the reasoning engine should return the answer `yes` with a successful substitution of the variables, such as `X=earthquake`, `x=c1:country`, `Y=tsunami`, and `y=c2:coastalArea`.

In the Semantic Web context, the argument manipulation shown above is very useful when software agents derive semantically related terms and assertions using ontologies. This is because the differences between argument structures in predicates must facilitate such flexible reasoning for predicate assertions in the sorted logic programming.

3 Order-Sorted Logic with Meta-Predicates

We introduce meta-predicates as new conceptual symbols in a sorted language. These meta-predicates represent n-ary relations among atomic predicate formulas and are used to construct a concept hierarchy.

Definition 1 *The alphabet of a sorted first-order language $\mathcal{L}$ with sort, predicate, and meta-predicate hierarchies contains the following symbols:*

1. *S: a countable set of sort symbols*
2. *F_n: a countable set of n-ary function symbols*
3. *P_n: a countable set of n-ary predicate symbols*
4. *Ψ_n: a countable set of n-ary meta-predicate symbols*
5. *$\leftarrow, \{, \}$: the connective and auxiliary symbols*
6. *V_s: an infinite set of variables $x\colon s, y\colon s, z\colon s, \ldots$ of sort s*

The set of all predicates is denoted by $P = \bigcup_{n \geq 1} P_n$, and the set of variables of all sorts is denoted by $V = \bigcup_{s \in S} V_s$.

Definition 2 (Sorted Signatures) *A signature of a sorted first-order language $\mathcal{L}$ with sort, predicate, and meta-predicate hierarchies (called a sorted signature) is a tuple $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ such that:*

1. *$(S, \leq)$ is a partially ordered set of sorts (called a sort hierarchy);*
2. *$(P, \leq)$ is a partially ordered set of predicates (called a predicate hierarchy);*
3. *$(\Psi_n, \leq)$ is a partially ordered set of n-ary meta-predicates (called a meta-predicate hierarchy);*
4. *Ω is a set of function and predicate declarations such that*
   (a) *if $f \in F_n$, then there is a unique function declaration of the form $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$, and*
   (b) *if $p \in P_n$, then there is a unique predicate declaration of the form $p\colon s_1 \times \cdots \times s_n \in \Omega$.*

The predicate hierarchy includes predicates with different argument structures, e.g., a binary predicate can be a subpredicate of a unary predicate. On the contrary, the meta-predicate hierarchy only contains meta-predicates with a fixed arity. In the sorted signature, Ω contains function and predicate declarations that determine the domains and ranges of functions f and predicates p. In particular, F_0 is the set of 0-ary functions (i.e., constants), and each constant $c \in F_0$ has a unique constant declaration of the form $c\colon \to s$.

We generally call sorts, predicates, and meta-predicates *concepts*. Let cp_1, cp_2, and cp_3 be three concepts. A concept cp_2 is called a upper bound for cp_1 if $cp_1 \leq cp_2$, and a concept cp_2 is called a lower bound for cp_1 if $cp_2 \leq cp_1$. The least upper bound $cp_1 \sqcup cp_2$ is a upper bound for cp_1 and cp_2 such that $cp_1 \sqcup cp_2 \leq cp_3$ holds for any other upper bound cp_3. The greatest lower bound $cp_1 \sqcap cp_2$ is a lower bound for cp_1 and cp_2 such that $cp_3 \leq cp_1 \sqcap cp_2$ holds for any other lower bound cp_3.

We define the following sorted expressions in the sorted signature Σ: terms, atoms (atomic formulas), meta-atoms (meta atomic formulas), goals, and clauses.

Definition 3 (Sorted Terms) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature. The set $\mathcal{T}_s$ of terms of sort s is defined by the following:*

1. *If $x\colon s \in V_s$, then $x\colon s \in \mathcal{T}_s$.*
2. *If $t_1 \in \mathcal{T}_{s_1}, \ldots, t_n \in \mathcal{T}_{s_n}$, $f \in F_n$, and $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$, then $f(t_1, \ldots, t_n)\colon s \in \mathcal{T}_s$.*
3. *If $t \in \mathcal{T}_{s'}$ and $s' \leq s$, then $t \in \mathcal{T}_s$.*

Note that $\mathcal{T}_s$ contains not only terms of sort s but also terms of subsorts s' of sort s if $s' \leq s$. The set of terms of all sorts is denoted by $\mathcal{T} = \bigcup_{s \in S} \mathcal{T}_s$.

The function *sort* is a mapping from sorted terms to their sorts, defined by (i) $sort(x\colon s) = s$ and (ii) $sort(f(t_1, \ldots, t_n)\colon s) = s$. Let $Var(t)$ denote the set of variables occurring in a sorted term t. A sorted term t is called *ground* if

$Var(t) = \emptyset$. $\mathcal{T}_0 = \{t \in \mathcal{T} \mid Var(t) = \emptyset\}$ is the set of sorted ground terms, and the set of ground terms of sort s is denoted by $\mathcal{T}_{0,s} = \mathcal{T}_0 \cap \mathcal{T}_s$. We write $\mathcal{T}_s^\Sigma$, $\mathcal{T}_0^\Sigma$, $\mathcal{T}_{s,0}^\Sigma$, and $\mathcal{T}^\Sigma$ for explicitly representing the sorted signature Σ.

In the following definition, sorted Horn clauses [17, 6] are extended by meta-atoms $\psi(A_1, \ldots, A_n)$ that consist of meta-predicates ψ and atoms $A_1, \ldots, A_n$.

Definition 4 (Atoms, Meta-Atoms, Goals, and Clauses) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature. The set $\mathcal{A}$ of atoms, the set $\mathcal{MA}$ of meta-atoms, the set $\mathcal{G}$ of goals, and the set $\mathcal{C}$ of clauses are defined by:*

1. *If $t_1 \in \mathcal{T}_{s_1}, \ldots, t_n \in \mathcal{T}_{s_n}$, $p \in P_n$, and $p\colon s_1 \times \cdots \times s_n \in \Omega$, then $p(t_1, \ldots, t_n) \in \mathcal{A}$.*
2. *If $A_1, \ldots, A_n \in \mathcal{A}$ and $\psi \in \Psi_n$, then $\psi(A_1, \ldots, A_n) \in \mathcal{MA}$.*
3. *If $L_1, \ldots, L_n \in \mathcal{A} \cup \mathcal{MA}$ $(n \geq 0)$, then $\{L_1, \ldots, L_n\} \in \mathcal{G}$.*
4. *If $G \in \mathcal{G}$ and $L \in \mathcal{A} \cup \mathcal{MA}$, then $L \leftarrow G \in \mathcal{C}$.*

Meta-atoms assert n-ary relations ψ over atoms $A_1, \ldots, A_n$ and can appear in the heads and bodies of extended Horn clauses. For example, the atoms $earthquake(c_1\colon country)$ and $tsunami(c_2\colon coastalArea)$ are used to assert the meta-atom $causes(earthquake(c_1\colon country), tsunami(c_2\colon coastalArea))$, where *causes* is a binary meta-predicate. A clause $L \leftarrow G$ is denoted by $L \leftarrow$ if $G = \emptyset$.

We define a sorted substitution such that each sorted variable $x\colon s$ is replaced with a sorted term in $\mathcal{T}_s$. Each sorted substitution is represented by $\{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$. Let θ be a sorted substitution. Then, θ is said to be a sorted ground substitution if for every variable $x\colon s \in Dom(\theta)$, $\theta(x\colon s)$ is a sorted ground term. Let E be a sorted expression. The substitution θ is a sorted ground substitution for E if $E\theta$ is ground and $Dom(\theta) = Var(E)$. The composition $\theta_1\theta_2$ of sorted substitutions θ_1 and θ_2 is defined by $\theta_1\theta_2(x\colon s) = \theta_2(\theta_1(x\colon s))$.

In Σ, there are various argument structures in the predicate hierarchy $(P, \leq)$ because P contains predicates with various arities. Additionally, we declare the argument structure for each predicate $p \in P$ in Σ as follows.

Definition 5 (Argument Declaration) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature. An argument declaration Λ is a pair (AN, Π) of a set AN of argument names and a set Π of argument structures of the form $p\colon \langle a_1, \ldots, a_n\rangle$ where $p \in P_n$, $a_1, \ldots, a_n \in AN$, and for any $i \neq j$, $a_i \neq a_j$.*

Given an argument declaration $\Lambda = (AN, \Pi)$, we define an argument function $Arg\colon P \to 2^{AN}$ such that $Arg(p) = \{a_1, \ldots, a_n\}$ for each $p\colon \langle a_1, \ldots, a_n\rangle \in \Pi$. An argument declaration Λ is well arranged in the predicate hierarchy if $Arg(q) \subseteq Arg(p)$ for any $p, q \in P$ with $p \leq q$. Intuitively, the well-arranged argument declaration implies that the predicate q does not have any argument that its subpredicate p does not have.

Definition 6 (Argument Elimination) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \ldots, d_n\rangle$ be an n-tuple, and let $p \in P_n$, $q \in P_m$ with $Arg(q) \subseteq Arg(p)$. An argument elimination*

from p to q is a function $\sigma^{-}_{p\to q}(\langle d_1,\ldots,d_n\rangle) = \langle d'_1,\ldots,d'_m\rangle$ such that

$$d'_i = d_j \text{ if } a'_i = a_j \text{ for each } 1 \le i \le m$$

where $p\colon \langle a_1,\ldots,a_n\rangle$ and $q\colon \langle a'_1,\ldots,a'_m\rangle$ in Π.

The argument eliminations will be used in the semantics and inference system of the order-sorted logic. An important property of argument eliminations that is used for the development of predicate-hierarchy reasoning is expressed as follows.

Proposition 1 (Transitivity of Argument Eliminations)
Let Σ be a sorted signature with an argument declaration Λ, let τ be an n-tuple, and let $p \in P_n$, $q \in P_m$, and $r \in P_k$. If $p \le q$, $q \le r$, and Λ is well arranged in Σ, then $\sigma^{-}_{q\to r}(\sigma^{-}_{p\to q}(\tau)) = \sigma^{-}_{p\to r}(\tau)$.

This proposition guarantees that argument eliminations are safely embedded in predicate-hierarchy reasoning if the argument declaration is well arranged.

We define the semantics of the order-sorted logic with sort, predicate, and meta-predicate hierarchies as follows.

Definition 7 (Σ-Models) *Let Σ be a sorted signature with a well-arranged argument declaration Λ. A Σ-model M is a tuple $(U, U_{\mathcal{F}}, I)$ such that*

1. *U is a non-empty set of individuals;*
2. *$U_{\mathcal{F}}$ is a non-empty set of facts;*
3. *I is a function with the following conditions:*
   (a) *if $s \in S$, then $I(s) \subseteq U$ (in particular, $I(\top) = U$),*
   (b) *if $s_i \le s_j$ for $s_i, s_j \in S$, then $I(s_i) \subseteq I(s_j)$,*
   (c) *if $f \in F_n$ and $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$, then $I(f)\colon I(s_1) \times \cdots \times I(s_n) \to I(s)$,*
   (d) *if $p \in P_n$ and $p\colon s_1 \times \cdots \times s_n \in \Omega$, then $I(p)\colon I(s_1) \times \cdots \times I(s_n) \to 2^{U_{\mathcal{F}}}$,*
   (e) *if $p \le q$ for $p \in P_n$ and $q \in P_m$, then $I(p)(\tau) \subseteq I(q)(\sigma^{-}_{p\to q}(\tau))$,*
   (f) *if $\psi \in \Psi_n$, then $I(\psi) \subseteq U^n_{\mathcal{F}}$,*
   (g) *if $\psi \le \phi$ for $\psi, \phi \in \Psi_n$, then $I(\psi) \subseteq I(\phi)$.*

The class of Σ-models is a restricted class of standard models such that the domains and ranges of functions and predicates are constrained by sorts and the hierarchies of sorts, predicates, and meta-predicates are interpreted by subset relations over U, $U_{\mathcal{F}}$, and $U^n_{\mathcal{F}}$.

By the argument eliminations in the predicate hierarchy, the following two properties are derived in the class of Σ-models.

Proposition 2 (Conceptuality of Predicates) *Let $p \in P_n$, $q \in P_m$, and $r \in P_k$ and let $\tau_1 \in U^n$, $\tau_2 \in U^m$, and $\tau \in U^k$. Every Σ-model M has the following properties:*

1. *$p \sqcup q \le r$ implies $I(p)(\tau_1) \cup I(q)(\tau_2) \subseteq I(r)(\tau)$ with $\tau = \sigma^{-}_{p\to r}(\tau_1) = \sigma^{-}_{q\to r}(\tau_2)$.*
2. *$r \le p \sqcap q$ implies $I(r)(\tau) \subseteq I(p)(\sigma^{-}_{r\to p}(\tau)) \cap I(q)(\sigma^{-}_{r\to q}(\tau))$.*

This property is important for showing that predicates are consistently conceptualized in a hierarchy. However, this is not simple because predicates have their respective arguments that have different structures in the predicate hierarchy.

Even if predicates are conceptually interpreted as sets of tuples, it is necessary to define a model that can identify each fact expressed by predicate formulas.

Proposition 3 (Identifiability of Predicates) *Let τ be an n-tuple in U^n, and let $p \in P_n$, $q \in P_m$ ($p \neq q$). Some Σ-models M have the following properties:*

1. *If $Arg(p) = Arg(q)$, then there are two facts $e_1 \in I(p)(\tau)$ and $e_2 \in I(q)(\tau)$.*
2. *If $Arg(p) \supsetneq Arg(q)$, then there are two facts $e_1 \in I(p)(\tau)$ and $e_2 \in I(q)(\sigma^{-}_{p \to q}(\tau))$.*

This proposition indicates that any two ground atoms with identical arguments $p(t_1, \ldots, t_n)$ and $q(t_1, \ldots, t_n)$ can be identified as distinct facts, if necessary. In the Σ-models, the set of facts $U_{\mathcal{F}}$ is used to identify ground atoms such that predicate assertions correspond to different elements in $U_{\mathcal{F}}$.

A variable assignment on a Σ-model $M = (U, U_{\mathcal{F}}, I)$ is a function $\alpha \colon V \to U$ where $\alpha(x \colon s) \in I(s)$. The variable assignment $\alpha[x \colon s/d]$ is defined by $(\alpha - \{(x \colon s, \alpha(x \colon s))\}) \cup \{(x \colon s, d)\}$. In other words, if $v = x \colon s$, then $\alpha[x \colon s/d](v) = d$, and otherwise $\alpha[x \colon s/d](v) = \alpha(v)$. Let $\Delta \subseteq U_{\mathcal{F}}$ be a valuation of facts on M. A Σ-interpretation $\mathcal{I}$ is a tuple (M, Δ, α) of a Σ-model M, a valuation of facts Δ on M, and a variable assignment α on M. The Σ-interpretation $(M, \Delta, \alpha[x \colon s/d])$ is simply denoted by $\mathcal{I}\alpha[x \colon s/d]$.

We define an interpretation of sorted terms and atoms as follows.

Definition 8 *Let $\mathcal{I} = (M, \Delta, \alpha)$ be a Σ-interpretation. The denotation function $[\![\ ]\!]_\alpha \colon \mathcal{T} \to U$ is defined by the following:*

1. $[\![x \colon s]\!]_\alpha = \alpha(x \colon s)$,
2. $[\![f(t_1, \ldots, t_n) \colon s]\!]_\alpha = I(f)([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha)$ *with* $f \colon s_1 \times \cdots \times s_n \to s \in \Omega$,
3. $[\![p(t_1, \ldots, t_n)]\!]_\alpha = I(p)([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha)$ *with* $p \colon s_1 \times \cdots \times s_n \in \Omega$.

The satisfiability of atoms, meta-atoms, goals, and clauses is defined by a Σ-interpretation $\mathcal{I}$.

Definition 9 (Σ-Satisfiability Relation) *Let $\mathcal{I} = (M, \Delta, \alpha)$ with $M = (U, U_{\mathcal{F}}, I)$ be a Σ-interpretation and let $F \in \mathcal{A} \cup \mathcal{MA} \cup \mathcal{G} \cup \mathcal{C}$. The Σ-satisfiability relation $\mathcal{I} \models F$ is defined inductively as follows:*

1. *$\mathcal{I} \models A$ iff $[\![A]\!]_\alpha \cap \Delta \neq \emptyset$.*
2. *$\mathcal{I} \models \psi(A_1, \ldots, A_n)$ iff $\mathcal{I} \models A_1, \ldots, \mathcal{I} \models A_n$ and $([\![A_1]\!]_\alpha \times \cdots \times [\![A_n]\!]_\alpha) \cap I(\psi) \neq \emptyset$.*
3. *$\mathcal{I} \models \{L_1, \ldots, L_n\}$ iff $\mathcal{I} \models L_1, \ldots, \mathcal{I} \models L_n$.*
4. *$\mathcal{I} \models L \leftarrow G$ iff for all $d_1 \in I(s_1), \ldots, d_n \in I(s_n)$, $\mathcal{I}\alpha[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n] \models G$ implies $\mathcal{I}\alpha[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n] \models L$ where $Var(L \leftarrow G) = \{x_1 \colon s_1, \ldots, x_n \colon s_n\}$.*

Let $F \in \mathcal{A} \cup \mathcal{MA} \cup \mathcal{G} \cup \mathcal{C}$. An expression F is said to be Σ-satisfiable if for some Σ-interpretation $\mathcal{I}$, $\mathcal{I} \models F$. Otherwise, it is Σ-unsatisfiable. F is a consequence of a set of expressions $\mathcal{S}$ in the class of Σ-interpretations (denoted $\mathcal{S} \models F$) if for every Σ-interpretation $\mathcal{I}$, $\mathcal{I} \models \mathcal{S}$ implies $\mathcal{I} \models F$.

4 Horn-Clause Calculus for Predicate Hierarchies

In this section, we define the order-sorted Horn-clause calculus that is extended by adding inference rules for predicate and meta-predicate hierarchies. A knowledge base $\mathcal{K}$ is a finite set of sorted clauses in Σ where $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ is a sorted signature with a well-arranged argument declaration Λ.

Definition 10 (Sorted Horn-Clause Calculus) *Let C be a ground clause, $\mathcal{K}$ be a knowledge base, and l be a label (non-negative integer). A derivation of C from $\mathcal{K}$ (denoted $\mathcal{K} \vdash l\colon C$) in the sorted Horn-clause calculus is defined as follows:*

- **Sorted substitution rule:** *Let $L \leftarrow G \in \mathcal{K}$ and θ be a sorted ground substitution for $L \leftarrow G$. Then, $\mathcal{K} \vdash l\colon (L \leftarrow G)\theta$ and l is incremented.*
- **Cut rule:** *Let $L \leftarrow G$ and $L' \leftarrow G' \cup \{L\}$ be ground clauses. If $\mathcal{K} \vdash l_1\colon L \leftarrow G$ and $\mathcal{K} \vdash l_2\colon L' \leftarrow G' \cup \{L\}$, then $\mathcal{K} \vdash l_2\colon L' \leftarrow G \cup G'$.*
- **Predicate hierarchy rule:** *Let $p(t_1, \ldots, t_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1\colon p(t_1, \ldots, t_n) \leftarrow G$ and $p \leq q$, then $\mathcal{K} \vdash l_1\colon q(t'_1, \ldots, t'_m) \leftarrow G$ where $\sigma^{-}_{p \to q}(\langle t_1, \ldots, t_n \rangle) = \langle t'_1, \ldots, t'_m \rangle$.*
- **Meta-predicate hierarchy rule:** *Let $\psi(A_1, \ldots, A_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1\colon \psi(A_1, \ldots, A_n) \leftarrow G$ and $\psi \leq \phi$, then $\mathcal{K} \vdash l_1\colon \phi(A_1, \ldots, A_n) \leftarrow G$.*
- **Fact derivation rule:** *Let $\psi(A_1, \ldots, A_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1\colon \psi(A_1, \ldots, A_n) \leftarrow G$, then $\mathcal{K} \vdash l\colon A_i \leftarrow G$ with $1 \leq i \leq n$ and l is incremented.*

We simply write $\mathcal{K} \vdash l\colon L$ if $\mathcal{K} \vdash l\colon L \leftarrow$. The sorted substitution rule and the cut rule serve as sorted inference rules in ordinary order-sorted logic. The sorted substitution rule yields well-sorted ground clauses in the sort hierarchy. The predicate hierarchy rule and the meta-predicate hierarchy rule can be used to derive predicate and meta-predicate assertions in the predicate and meta-predicate hierarchies, respectively. The fact derivation rule derives atoms from meta-atoms, which was used in the third motivating example of Section 2.

To prove the completeness of the Horn-clause calculus, we construct extended Herbrand models for knowledge bases where positive atoms labeled by non-negative integers are used to identify different facts. We write $\mathcal{K} \vdash_{\psi(A_1,\ldots,A_n)} l\colon A_i$ if a labeled atom $l\colon A_i$ is directly derived from a labeled meta-atom $l_1\colon \psi(A_1, \ldots, A_n)$ using the fact derivation rule. Let $L \leftarrow G$ be a clause. We define $\mathit{ground}(L \leftarrow G)$ as the set of sorted ground clauses for $L \leftarrow G$. We define $\mathit{ground}(\mathcal{K}) = \bigcup_{L \leftarrow G \in \mathcal{K}} \mathit{ground}(L \leftarrow G)$ as the set of sorted ground clauses for all $L \leftarrow G$ in $\mathcal{K}$.

Definition 11 (Herbrand Models) *Let $\mathcal{K}$ be a knowledge base. A Herbrand model M_H for $\mathcal{K}$ is a tuple $(U_H, U_{\mathcal{F},H}, I_H)$ such that*

1. $U_H = \mathcal{T}_0$,
2. $U_{\mathcal{F},H} = \mathbb{N} - \{l \in \mathbb{N} \mid \mathit{ground}(\mathcal{K}) \vdash l\colon L \leftarrow G \ \&\ L \in \mathcal{MA}\}$,

3. I_H *is a function with the following conditions:*
   (a) $I_H(s) = \mathcal{T}_{0,s}$ *for each sort* $s \in S$,
   (b) *if* $f \in F_n$ *and* $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$, *then* $I_H(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)\colon s$ *where* $t_1 \in I_H(s_1), \ldots, t_n \in I_H(s_n)$,
   (c) *if* $p \in P_n$ *and* $p\colon s_1 \times \cdots \times s_n \in \Omega$, *then* $I_H(p)(\tau) = \bigcup_{q \leq p}\{l \in U_{\mathcal{F},H} \mid ground(\mathcal{K}) \vdash l\colon q(\tau')\}$ *with* $\sigma^{-}_{q \to p}(\tau') = \tau$,
   (d) *if* $\psi \in \Psi_n$, *then* $I_H(\psi) = \bigcup_{\phi \leq \psi}\{(l_1, \ldots, l_n) \in U^n_{\mathcal{F},H} \mid$ *for every* $1 \leq i \leq n$, $ground(\mathcal{K}) \vdash_{\phi(A_1,\ldots,A_n)} l_i\colon A_i\}$.

A Herbrand interpretation $\mathcal{I}_H$ for $\mathcal{K}$ is a tuple (M_H, Δ_H, α) such that $M_H = (U_H, U_{\mathcal{F},H}, I_H)$ is a Herbrand model for $\mathcal{K}$, $\Delta_H = \bigcup_{p \in P} \bigcup_{\tau \in \mathcal{T}_{0,s_1} \times \cdots \times \mathcal{T}_{0,s_n}} I_H(p)(\tau)$ with $p\colon s_1 \times \cdots \times s_n \in \Omega$ is a valuation of facts on M_H, and α is a variable assignment on M_H.

We show that a Herbrand interpretation $\mathcal{I}_H$ is a Σ-interpretation that satisfies a knowledge base $\mathcal{K}$.

Lemma 1 *Let* $\mathcal{K}$ *be a knowledge base, let* $\mathcal{I}_H$ *be a Herbrand interpretation for* $\mathcal{K}$, *and let* $L \leftarrow G$ *be a clause. Then, the following statements hold:*

1. $\mathcal{I}_H \models L \leftarrow G$ *if and only if* $\mathcal{I}_H \models ground\ (L \leftarrow G)$.
2. $\mathcal{I}_H$ *is a* Σ*-interpretation of* $\mathcal{K}$.

We use the Herbrand model and the abovementioned lemma to prove the completeness of the Horn-clause calculus as follows.

Theorem 1 (Completeness of Horn-Clause Calculus) *Let* $\mathcal{K}$ *be a knowledge base in a sorted signature* Σ *and* L *be a ground atom or meta-atom.* $\mathcal{K} \models L$ *iff* $\mathcal{K} \vdash l\colon L$.

We show the termination of the Horn-clause calculus where a sorted signature is function-free.

Theorem 2 (Termination of Horn-Clause Calculus) *Let* $\mathcal{K}$ *be a knowledge base in a sorted signature* Σ. *Then, the Horn-clause calculus terminates if* Σ *is function-free.*

The termination of the calculus is proved by the fact that the set of derivable clauses $Con(\mathcal{K}) = \{L \leftarrow G \mid \mathcal{K} \vdash l\colon L \leftarrow G\}$ is finite. In other words, the calculus cannot generate terms and clauses infinitely because the cardinality of $Con(\mathcal{K})$ is bounded by finite constant, predicate, and meta-predicate symbols in $\mathcal{K}$.

We show the complexity of the derivation for atoms or meta-atoms L (not limited to ground) from a knowledge base where the set of ground atoms or meta-atoms $L\theta$ is computed using the Horn-clause calculus.

Corollary 1 (Complexity of Derivation for Atoms or Meta-Atoms) *Let* $\mathcal{K}$ *be a knowledge base in a sorted signature* Σ, L *be an atom or meta-atom, and* θ *be a sorted ground substitution for* L. *If* Σ *is function-free, then deriving the set of ground atoms or meta-atoms* $L\theta$ *with* $\mathcal{K} \vdash l\colon L\theta$ *is (single) EXPTIME-complete (w.r.t. the length of* $\mathcal{K}$*).*

5 Query System

We describe a query-answering system for our order-sorted logic programming. In this system, query expressions are generalized by adding predicate variables in meta-atoms. The set of predicate variables is denoted by $\mathcal{V}$. The set of atoms with predicate variables is defined by $\mathcal{A}^{\mathcal{V}} = \{X\colon p(t_1, \ldots, t_n) \mid X \in \mathcal{V},\ p(t_1, \ldots, t_n) \in \mathcal{A}\}$. We call the form $X\colon p(t_1, \ldots, t_n)$ a predicate variable atom.

Definition 12 (Queries) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with a well-arranged argument declaration Λ, and let $\mathcal{MA}^{\mathcal{V}} = \{\psi(A_1^+, \ldots, A_n^+) \mid \psi \in \Psi_n,\ A_1^+, \ldots, A_n^+ \in \mathcal{A} \cup \mathcal{A}^{\mathcal{V}}\}$ be the set of meta-atoms with predicate variables. The set $\mathcal{Q}$ of queries is defined by that if $L_1, \ldots, L_h \in \mathcal{A} \cup \mathcal{A}^{\mathcal{V}} \cup \mathcal{MA}^{\mathcal{V}}$, then $\{L_1, \ldots, L_h\} \in \mathcal{Q}$.*

We introduce substitutions for predicate variables $X \in \mathcal{V}$ such that each predicate variable atom $X\colon q(t'_1, \ldots, t'_m)$ is replaced with an atom $A \in \mathcal{A}$. We denote the set of atoms restricted to the subpredicates p of q by $\mathcal{A}_q = \{p(t_1, \ldots, t_n) \in \mathcal{A} \mid p \leq q \ \&\ \sigma^-_{p \to q}(\langle t_1, \ldots, t_n \rangle) = \langle t'_1, \ldots, t'_m \rangle\}$.

Definition 13 (Substitutions for Predicate Variables)
A substitution for predicate variables is a partial function $\delta\colon \mathcal{A}^{\mathcal{V}} \to \mathcal{A}$ such that $\delta(X\colon q(t'_1, \ldots, t'_m)) \in \mathcal{A}_q$ and the domain of δ (denoted $Dom(\delta)$) is finite.

The substitutions for predicate variables follow the predicate hierarchy, i.e., a subpredicate p of q is substituted for the predicate variable atom $X\colon q(\tau)$. A substitution δ is a most specific substitution for a predicate variable atom $X\colon q(\tau)$ if $\delta(X\colon q(\tau)) = p(\tau')$ with $\sigma^-_{p \to q}(\tau') = \tau$ and there is no other substitution δ' such that $\delta'(X\colon q(\tau)) = r(\tau'')$ with $\sigma^-_{r \to q}(\tau'') = \tau$ and $r \leq p$.

Definition 14 (Query System) *Let Q be a query in $\mathcal{Q}$, δ be a substitution for predicate variables in Q, and θ be a sorted substitution for $Q\delta$. Then, the query system $Query\colon \mathcal{Q} \to \{yes, no\}$ is defined by the following rule.*

(i) If there exists $\mathcal{K} \vdash l\colon Q\delta\theta$ such that $Var(Q\delta) \cap \mathcal{V} = \emptyset$ and $Var(Q\delta\theta) = \emptyset$, then $Query(Q) = yes$.
(ii) Otherwise, $Query(Q) = no$.

Without losing decidability, the query system is realized in the following two steps. First, atoms are substituted for predicate variable atoms in a query Q along with the predicate hierarchy. Second, predicate and meta-predicate assertions in the substituted query $Q\delta$ are derived using the Horn-clause calculus.

Theorem 3 (Termination of Query System) *Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ. Then, the query system terminates if Σ is function-free.*

The termination leads to the following corollary that the complexity of the query-answering system is unaffected by the introduction of predicate variables in the queries.

Corollary 2 (Complexity of Query System) *Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ and let Q be a query. If Σ is function-free, then deciding $Query(Q)$ is (single) EXPTIME-complete (w.r.t. the length of $\mathcal{K}$).*

6 Derivation using Argument Restructuring

In the Horn-clause calculus (discussed in Section 4), redundant arguments in each predicate are deleted during the derivation of super predicates if the argument structures are well-arranged in a hierarchy. In this section, we generalize sorted signatures by removing the condition of their being well-arranged, i.e., some predicates may have an argument that their subpredicates do not have.

We give some examples of hierarchies in a query-answering system for the case where argument structures are not well-arranged in the sort, predicate, and meta-predicate hierarchies shown in Figs. 1 and 2. If the fact `assaults(tom:minor)` is valid, then the super predicate `illegalAct` can be derived in the predicate hierarchy as follows.

```
assaults(tom:minor)
?-illegalAct(x:human,mary:woman)
no
?-illegalAct(x:human,y:human)
yes
x=tom:minor, y=c:human
```

In the first case, there is no fact that indicates someone acts against the second argument `mary:woman` in the query. Thus, the answer to the first query is `no`. In the second case, we can obtain the answer `yes` to the second query from the fact `assaults(tom:minor)` and the predicate hierarchy. A new constant `c:human` is substituted for the variable y because the argument structure of the predicate `assaults` lacks the second argument of the predicate `illegalAct`.

For such argument structures in a predicate hierarchy (in a sorted signature), we perform the addition of missing arguments for the derivation of super predicates as follows.

Definition 15 (Naive Argument Restructuring) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \ldots, d_n \rangle$ be an n-tuple, and let $p \in P_n$ and $q \in P_m$. An argument restructuring from p to q is a function $\sigma^+_{p \to q}(\langle d_1, \ldots, d_n \rangle) = \langle d'_1, \ldots, d'_m \rangle$ such that*

$$d'_i = \begin{cases} d_j & \text{if } a'_i = a_j \\ c_i & \text{otherwise} \end{cases}$$

where $p\colon \langle a_1, \ldots, a_n \rangle$ and $q\colon \langle a'_1, \ldots, a'_m \rangle$ in Π and each c_i is a new element.

We refine the definition of Σ-models such a way that every argument elimination $\sigma^-_{p \to q}$ is replaced with an argument restructuring $\sigma^+_{p \to q}$. The satisfiability relation $\models$ is denoted by $\models_{\sigma^+}$ if an argument restructuring σ^+ is employed in each Σ-model. The conceptuality and identifiability of predicates in Propositions 2 and 3 hold for the case where the Σ-models are refined by replacement with an argument restructuring σ^+.

In order to embed an argument restructuring σ^+ in the Horn-clause calculus, we further extend the calculus as follows.

Definition 16 (Extended Sorted Horn-Clause Calculus)
Let C be a ground clause and $\mathcal{K}$ be a knowledge base. A derivation of C from $\mathcal{K}$ (denoted $\mathcal{K} \vdash_{\sigma^+} l\colon C$) in the sorted Horn-clause calculus is extended by replacing the predicate hierarchy rule with the following rule:

- **Predicate hierarchy rule$^+$:** *Let $p(t_1, \ldots, t_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1\colon p(t_1, \ldots, t_n) \leftarrow G$ and $p \leq q$, then $\mathcal{K} \vdash l_1\colon q(t'_1, \ldots, t'_m) \leftarrow G$ where $\sigma^+_{p \to q}(\langle t_1, \ldots, t_n \rangle) = \langle t'_1, \ldots, t'_m \rangle$.*

An atom A_1 is a parent of another atom A_2 if $\mathcal{K} \vdash_{\sigma^+} l\colon A_2 \leftarrow G$ is derived from $\mathcal{K} \vdash_{\sigma^+} l\colon A_1 \leftarrow G$ by an application of the predicate hierarchy rule. An atom A_1 is an ancestor of another atom A_2 if (i) A_1 is a parent of A_2 or (ii) A_1 is an ancestor of an atom A and A is a parent of A_2. Let A be an atom $p(t_1, \ldots, t_n)$ with $p\colon \langle a_1, \ldots, a_n \rangle \in \Pi$. We denote the occurrence of an argument name a_k and a term t_k in A by $A[a_k, t_k]$ if $1 \leq k \leq n$. The set of pairs of argument names and terms for a labeled atom $l\colon A$ is defined by $AL(l\colon A) = \{(a, t) \mid A[a, t]\} \cup \{(a, t) \mid A'[a, t]$ is an ancestor of $A\}$.

In the following definition, we introduce a label-based argument restructuring in order to solve the problem of incomplete derivation, i.e., the transitivity in Proposition 1 no longer holds if the argument structures are not well-arranged. Hence, it is necessary to solve the problem to prove the completeness of the extended sorted Horn-clause calculus.

Definition 17 (Label-Based Argument Restructuring in Derivation)
Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \ldots, d_n \rangle$ be an n-tuple, let $p \in P_n$ and $q \in P_m$, and l be a label (non-negative integer). An argument restructuring from p to q is label-based if it is defined as a function $\sigma^*_{p \to q}(\langle t_1, \ldots, t_n \rangle) = \langle t'_1, \ldots, t'_m \rangle$ such that

$$t'_i = \begin{cases} t_j & \text{if } a'_i = a_j \text{ with } (a_j, t_j) \in AL(l\colon p(t_1, \ldots, t_n)) \\ c_{l, a'_i} & \text{otherwise} \end{cases}$$

where $p\colon \langle a_1, \ldots, a_n \rangle$ and $q\colon \langle a'_1, \ldots, a'_m \rangle$ in Π and each c_{l,a'_i} is a new constant indexed by the pair of the label l and the argument name a'_i.

We denote the set of new constants that are used to add missing arguments in a label-based argument restructuring σ^* by $F_{0,new}$. The label-based argument restructuring σ^* can be applied to a tuple of terms $t_1, \ldots, t_n$ in a labeled atom $l\colon p(t_1, \ldots, t_n)$ in the derivation. This leads to the following transitivity, although the transitivity of naive argument restructurings σ^+ does not hold.

Proposition 4 (Transitivity of Label-Based Argument Restructurings)
Let Σ be a sorted signature with an argument declaration Λ, let τ be an n-tuple, and let $p \in P_n$, $q \in P_m$, and $r \in P_k$. If $p \leq q$ and $q \leq r$, then $\sigma^*_{q \to r}(\sigma^*_{p \to q}(\tau)) = \sigma^*_{p \to r}(\tau)$.

The transitivity of label-based argument restructurings will be used to show the completeness of the extended sorted Horn-clause calculus.

Theorem 4 (Completeness of Extended Horn-Clause Calculus) *Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ and L be a ground atom or meta-atom. $\mathcal{K} \models_{\sigma^+} L$ iff $\mathcal{K} \vdash_{\sigma^*} L$.*

Note that the consequence relation $\mathcal{K} \models_{\sigma^+} L$ is defined with a naive argument restructuring σ^+ but the derivation $\mathcal{K} \vdash_{\sigma^*} L$ is extended to contain a label-based argument restructuring σ^*. This is because $\mathcal{K} \vdash_{\sigma^+} L$ is incomplete for $\mathcal{K} \models_{\sigma^+} L$, i.e., the derivation is insufficient for the semantics.

However, the label-based argument restructurings σ^* lead to the undecidability of the extended sorted Horn-clause calculus as follows.

Theorem 5 (Undecidability of Extended Horn-Clause Calculus) *The extended Horn-clause calculus does not terminate for a knowledge base $\mathcal{K}$ in a function-free sorted signature Σ.*

Let p be an n-ary predicate and τ be an n-tuple of sorted terms. We denote an atom or meta-atom L by L_p if $L = p(\tau)$ or $L = \psi(A_1, \ldots, A_m)$ with $A_i = p(\tau)$ for some $1 \leq i \leq m$.

Definition 18 (Paths in a Knowledge Base) *Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ, let L_p, L_q be atoms or meta-atoms, let a, a' be argument names, and let t be a sorted term. Then, there is a path from $L_p[a, t]$ to $L_q[a', t]$ in $\mathcal{K}$ if one of the following conditions holds:*

1. *$a = a'$, $p \leq q$, and $a \in \arg(p) \cap \arg(q)$,*
2. *$L_q[a', x\colon s] \leftarrow G$ where $L_p[a, x\colon s] \in G$ and $t \in \mathcal{T}_s$, and*
3. *there are two paths from $L_p[a, t]$ to $L_r[a'', t]$ and from $L_r[a'', t]$ to $L_q[a', t]$.*

In order to avoid the undecidability, we define a restricted set of knowledge bases (called safe knowledge bases).

Definition 19 (Safe Knowledge Bases) *A knowledge base $\mathcal{K}$ is safe if*

1. *$Var(L) \subseteq Var(G)$ for every clause $L \leftarrow G$ in $\mathcal{K}$, and*
2. *there is no path from $L_p[a, t]$ to $L_q[a', t]$ in $\mathcal{K}$ such that $q \leq p$, $a \neq a'$, $a \notin \arg(q)$, and $a \in \arg(p)$.*

Lemma 2 *Let $\mathcal{K}$ be a safe knowledge base in a sorted signature Σ. Then, the extended Horn-clause calculus with label-based argument restructuring does not generate new constants infinitely.*

Furthermore, we can show the complexity of the extended sorted Horn-clause calculus with label-based argument restructuring where Σ is function-free.

Theorem 6 (Complexity of Derivation for Atoms or Meta-Atoms) *Let $\mathcal{K}$ be a safe knowledge base in a sorted signature Σ, L be an atom or meta-atom, and θ be a sorted ground substitution for L. If Σ is function-free, then deriving the set of ground atoms or meta-atoms $L\theta$ with $\mathcal{K} \vdash_{\sigma^*} l\colon L\theta$ is (single) EXPTIME-complete (w.r.t. the length of $\mathcal{K}$).*

Table 1. The Complexities of Horn-Clause Calculus with Argument Manipulation

Horn-clause calculus	complexities
argument elimination	EXPTIME
naive argument restructuring	undecidable and incomplete
label-based argument restructuring	undecidable and complete
label-based argument restructuring for safe knowledge bases	EXPTIME

Table 1 lists the complexities of the Horn-clause calculus with argument elimination, naive argument restructuring, and label-based argument restructuring. We can extend the query system by using the Horn-clause calculus with label-based argument restructuring.

Theorem 7 (Complexity of Extended Query System)
Let $\mathcal{K}$ be a safe knowledge base in a sorted signature Σ and let Q be a query. If Σ is function-free, then deciding Query(Q) is (single) EXPTIME-complete (w.r.t. the length of $\mathcal{K}$).

Due to spatial constraints, detailed proofs of the lemmas and theorems in this paper have been omitted (see http://kc.nict.go.jp/kaneiwa/).

7 Conclusions

We have developed an order-sorted logic programming language equipped with concept hierarchies of sorts, predicates, and meta-predicates. Predicates with differently structured arguments are conceptually interpreted in the semantics. According to the semantics, predicate-hierarchy reasoning is realized in the hierarchies of predicates and meta-predicates such that predicate assertions are used as arguments of meta-level predicates. To achieve such enhanced reasoning, we design inference rules for predicate and meta-predicate hierarchies in the order-sorted Horn-clause calculus. We employ the calculus to develop a query-answering system for generalized queries containing predicate variables. We show that the complexity of our expressive query-answering system is identical to that of DATALOG. We analyze several complexity results where argument restructuring gives rise to undecidable reasoning services in the derivation of super predicates in a predicate hierarchy, but a set of safe knowledge bases preserves the decidability of the derivation with argument restructuring.

References

1. http://www.ruleml.org/.
2. http://www.w3.org/tr/owl2-profiles/.
3. W. Chen and M. Kifer. Sorted HiLog: Sorts in higher-order logic data languages. In *Proc. of the 5th International Conference on Database Theory (ICDT'95)*, LNCS 893, pages 252–265. Springer, 1995.

4. A. G. Cohn. Taxonomic reasoning with many sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.
5. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. In *IEEE Conference on Computational Complexity*, pages 82–101, 1997.
6. K. Doets. *From Logic to Logic Programming*. The MIT Press, 1994.
7. B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logics. In *Proc. of the Twelfth International World Wide Web Conference (WWW 2003)*, Budapest, Hungary, 2003.
8. M. Hanus. Logic programming with type specifications. In F. Pfenning, editor, *Types in Logic Programming*. The MIT Press, 1992.
9. P. Hitzler and B. Parsia. Ontologies and rules. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, 2nd ed., 2009.
10. I. Horrocks and P. F. Patel-Schneider. A proposal for an owl rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.
11. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Recommendation, http://www.w3.org/submission/swrl/.
12. J.-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming (ICALP91)*, LNCS510, pages 455–468, 1991.
13. K. Kaneiwa. Order-sorted logic programming with predicate hierarchy. *Artificial Intelligence*, 158(2):155–188, 2004.
14. K. Kaneiwa and R. Mizoguchi. An order-sorted quantified modal logic for meta-ontology. In *Proceedings of the International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX2005)*, pages 169–184. LNCS 3702, Springer–Verlag, 2005.
15. K. Kaneiwa and R. Mizoguchi. Distributed reasoning with ontologies and rules in order-sorted logic programming. *Journal of Web Semantics*, 2009 (in press).
16. M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proceedings of the 7th International Semantic Web Conference (ISWC 2008)*, LNCS 5318, pages 649–664, 2008.
17. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.
18. B. Motik. On the Properties of Metamodeling in OWL. *Journal of Logic and Computation*, 17(4):617–637, 2007.
19. P.H.P. Nguyen, K. Kaneiwa, D.R. Corbett, and M.-Q. Nguyen. An ontology formalization of relation type hierarchy in conceptual structure theory. In *Proceedings of the 21th Australian Joint Conference on Artificial Intelligence (AI2008)*, pages 79–85. LNCS 5360, Springer–Verlag, 2007.
20. P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, http://www.w3.org/tr/2004/rec-owl-semantics-20040210/.
21. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
22. R. Socher-Ambrosius and P. Johann. *Deduction Systems*. Springer-Verlag, 1996.
23. W. Woods and J. Schmolze. The KL-ONE family. *Computers and Mathematics with Applications, Special Issue on Semantic Networks in Artificial Intelligence, Part 1*, 23(2–5):133–178, 1992.