

The Decidability and Complexity of Order-Sorted Logic Programming for Ontologies and Rules with Argument Restructuring

Ken Kaneiwa¹ and Philip H.P. Nguyen²

¹ National Institute of Information and Communications Technology
kaneiwa@nict.go.jp

² Department of Justice, Government of South Australia
philip.nguyen@sa.gov.au

Abstract. Decidable reasoning between ontologies and rules is required for the Semantic Web. 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 [27] have been studied for modeling ontologies in OWL (Web Ontology Language) [23]. 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, 26, 14, 15] (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 [22] was extended to include relation types and their type hierarchy for building complex ontologies.

¹ This paper is an extended version of [16], containing proofs as well as some additional definitions, theorems, lemmas, and examples.

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 [21]. 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, 24, 20], DLP (Description Logic Programs) [7], and the rule-based language ELP [17] (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 programming 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 higher-order language with a first-order semantics for logic programming, and it seems to 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

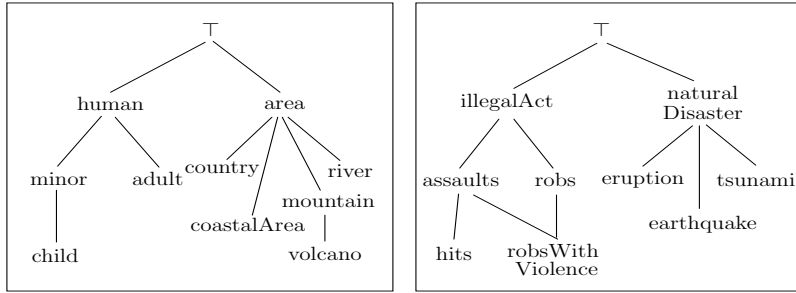


Fig. 1. Sort and predicate hierarchies

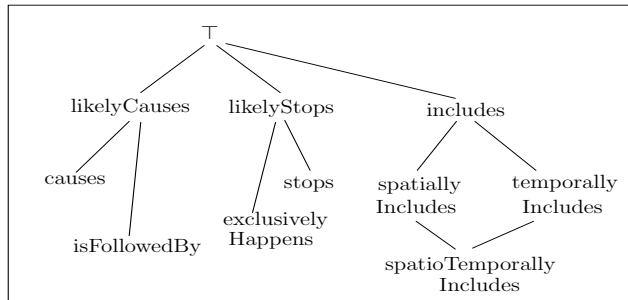


Fig. 2. A meta-predicate hierarchy

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.

The rest of this paper is organized as follows. In Section 2, we provide some illustrative examples of hierarchies of sorts, predicates, and meta-predicates. In Section 3, we formalize the syntax and semantics of predicate and meta-predicate hierarchies in order-sorted logic programming. In Section 4, we define inference rules for the hierarchies in the Horn-clause calculus, and in Section 5, we use the calculus to develop a query-answering system that can solve queries containing predicate variables. In Section 6, we introduce an argument restructuring operation in the Horn-clause calculus and the query-answering system that helps eliminate and supplement arguments in predicate derivation. In Section 7, we provide concluding remarks and discuss our future research plan.

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

First, we define the syntax and semantics of order-sorted logic with sort, predicate, and meta-predicate hierarchies (an extension of [26, 25, 19]).

3.1 Syntax

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 $x: s, y: s, z: s, \dots$ 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. (Ψ_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 $f: s_1 \times \dots \times s_n \rightarrow s \in \Omega$, and
 - (b) if $p \in P_n$, then there is a unique predicate declaration of the form $p: s_1 \times \dots \times s_n \in \Omega$.

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 $c: \rightarrow s$.

Example 1 Let us consider the sorted signature $\Sigma = (S, P, \Psi_2, \Omega, \leq^*)$ such that

$$\begin{aligned}
S &= \{ \text{child, minor, adult, human, bank, mountain, volcano, river, area,} \\
&\quad \text{coastalArea, country, } \top \}, \\
P &= \{ \text{hits, violates, robs, robsWithViolence, illegalAct, eruption,} \\
&\quad \text{earthquake, tsunami, naturalDisaster, } \top \}, \\
\Psi_2 &= \{ \text{causes, isFollowedBy, likelyCauses, stops,} \\
&\quad \text{exclusivelyHappens, likelyStops, temporalIncludes,} \\
&\quad \text{spatiallyIncludes, spatioTemporallyIncludes, includes, } \top \}, \\
\Omega &= \{ \text{tom: } \rightarrow \text{minor, john: } \rightarrow \text{adult, } c_1: \rightarrow \text{country, } c_2: \rightarrow \text{coastalArea,} \\
&\quad \text{hits: human} \times \text{human, violates: human} \times \text{human, illegalAct: human,} \\
&\quad \text{robsWithViolence: human} \times \text{bank} \times \text{human, robs: human} \times \text{bank,} \\
&\quad \text{earthquake: area, tsunami: area, naturalDisaster: area} \}, \\
\leq &= \{ \text{child} \leq \text{minor, minor} \leq \text{human, adult} \leq \text{human, river} \leq \text{area} \\
&\quad \text{country} \leq \text{area, volcano} \leq \text{mountain, mountain} \leq \text{area,} \\
&\quad \text{coastalArea} \leq \text{area} \} \cup \\
&\{ \text{hits} \leq \text{violates, violates} \leq \text{illegalAct, robs} \leq \text{illegalAct,} \\
&\quad \text{robsWithViolence} \leq \text{robs, robsWithViolence} \leq \text{violates,} \\
&\quad \text{eruption} \leq \text{naturalDisaster, tsunami} \leq \text{naturalDisaster,} \\
&\quad \text{earthquake} \leq \text{naturalDisaster} \} \cup \\
&\{ \text{causes} \leq \text{likelyCauses, isFollowedBy} \leq \text{likelyCauses,} \\
&\quad \text{stops} \leq \text{likelyStops, exclusivelyHappens} \leq \text{likelyStops,} \\
&\quad \text{spatioTemporallyIncludes} \leq \text{temporalIncludes,} \\
&\quad \text{spatioTemporallyIncludes} \leq \text{spatiallyIncludes,} \\
&\quad \text{temporalIncludes} \leq \text{includes, spatiallyIncludes} \leq \text{includes} \}.
\end{aligned}$$

We denote the transitive and reflexive closure of $\leq \cup \{cp \leq \top \mid cp \in S \cup P \cup \Psi_2\}$ by \leq^* . This sorted signature represents the sort, predicate, and meta-predicate hierarchies in Figs. 1 and 2.

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 $x: s \in V_s$, then $x: s \in \mathcal{T}_s$.
2. If $t_1 \in \mathcal{T}_{s_1}, \dots, t_n \in \mathcal{T}_{s_n}$, $f \in F_n$, and $f: s_1 \times \dots \times s_n \rightarrow s \in \Omega$, then $f(t_1, \dots, t_n): s \in \mathcal{T}_s$.
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) $\text{sort}(x: s) = s$ and (ii) $\text{sort}(f(t_1, \dots, t_n): s) = s$. Let $\text{Var}(t)$ denote the set of variables occurring in a sorted term t . A sorted term t is called *ground* if $\text{Var}(t) = \emptyset$. $\mathcal{T}_0 = \{t \in \mathcal{T} \mid \text{Var}(t) = \emptyset\}$ 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^Σ , \mathcal{T}_0^Σ , $\mathcal{T}_{s,0}^\Sigma$, and \mathcal{T}^Σ for explicitly representing the sorted signature Σ .

In the following definition, sorted Horn clauses [18, 6] are extended by meta-atoms $\psi(A_1, \dots, A_n)$ that consist of meta-predicates ψ and atoms A_1, \dots, 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 $t_1 \in \mathcal{T}_{s_1}, \dots, t_n \in \mathcal{T}_{s_n}$, $p \in P_n$, and $p: s_1 \times \dots \times s_n \in \Omega$, then $p(t_1, \dots, t_n) \in \mathcal{A}$.
2. If $A_1, \dots, A_n \in \mathcal{A}$ and $\psi \in \Psi_n$, then $\psi(A_1, \dots, A_n) \in \mathcal{MA}$.
3. If $L_1, \dots, L_n \in \mathcal{A} \cup \mathcal{MA}$ ($n \geq 0$), then $\{L_1, \dots, L_n\} \in \mathcal{G}$.
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, \dots, A_n and can appear in the heads and bodies of extended Horn clauses. For example, the atoms $\text{earthquake}(c_1: \text{country})$ and $\text{tsunami}(c_2: \text{coastalArea})$ are used to assert the meta-atom $\text{causes}(\text{earthquake}(c_1: \text{country}), \text{tsunami}(c_2: \text{coastalArea}))$, 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 $x: s$ is replaced with a sorted term in \mathcal{T}_s .

Definition 5 (Sorted Substitutions) A sorted substitution is a partial function $\theta: V \rightarrow \mathcal{T}$ such that $\theta(x: s) \in \mathcal{T}_s - \{x: s\}$ and the domain of θ (denoted $\text{Dom}(\theta)$) is finite. The substituted expression $E\theta$ is defined by the following:

1. $x: s\theta = \theta(x: s)$ if $x: s \in \text{Dom}(\theta)$,
2. $x: s\theta = x: s$ if $x: s \notin \text{Dom}(\theta)$,
3. $(f(t_1, \dots, t_n): s)\theta = f(t_1\theta, \dots, t_n\theta): s$,
4. $(p(t_1, \dots, t_n))\theta = p(t_1\theta, \dots, t_n\theta)$,
5. $(\psi(A_1, \dots, A_n))\theta = \psi(A_1\theta, \dots, A_n\theta)$,
6. $\{L_1, \dots, L_n\}\theta = \{L_1\theta, \dots, L_n\theta\}$,
7. $(L \leftarrow G)\theta = L\theta \leftarrow G\theta$.

Each sorted substitution is represented by $\{x_1: s_1/t_1, \dots, x_n: s_n/t_n\}$. Let θ be a sorted substitution. Then, θ is said to be a sorted ground substitution if for every variable $x: s \in \text{Dom}(\theta)$, $\theta(x: s)$ 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 $\text{Dom}(\theta) = \text{Var}(E)$. The composition $\theta_1\theta_2$ of sorted substitutions θ_1 and θ_2 is defined by $\theta_1\theta_2(x: s) = \theta_2(\theta_1(x: s))$.

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 6 (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 $p: \langle a_1, \dots, a_n \rangle$ where $p \in P_n$, $a_1, \dots, a_n \in AN$, and for any $i \neq j$, $a_i \neq a_j$.*

Given an argument declaration $\Lambda = (AN, \Pi)$, we define an argument function $\text{Arg}: P \rightarrow 2^{AN}$ such that $\text{Arg}(p) = \{a_1, \dots, a_n\}$ for each $p: \langle a_1, \dots, a_n \rangle \in \Pi$. An argument declaration Λ is well arranged in the predicate hierarchy if $\text{Arg}(q) \subseteq \text{Arg}(p)$ 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. In addition, we assume that every sorted signature $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ satisfies the following condition: for any $p, q \in P$ where $p \leq q$, $p: s_1 \times \dots \times s_n \in \Omega$, and $q: s'_1 \times \dots \times s'_m \in \Omega$, if $a_i = a'_j$, $p: \langle a_1, \dots, a_n \rangle$, and $q: \langle a'_1, \dots, a'_m \rangle$ in Λ , then $s_i \leq s'_j$ in Σ .

Definition 7 (Argument Elimination) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \dots, d_n \rangle$ be an n -tuple, and let $p \in P_n$, $q \in P_m$ with $\text{Arg}(q) \subseteq \text{Arg}(p)$. An argument elimination from p to q is a function $\sigma_{p \rightarrow q}^-(\langle d_1, \dots, d_n \rangle) = \langle d'_1, \dots, d'_m \rangle$ such that*

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

where $p: \langle a_1, \dots, a_n \rangle$ and $q: \langle a'_1, \dots, a'_m \rangle$ 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 can be 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 $p \leq q$, $q \leq r$, and Λ is well arranged in Σ , then $\sigma_{q \rightarrow r}^-(\sigma_{p \rightarrow q}^-(\tau)) = \sigma_{p \rightarrow r}^-(\tau)$.

Proof. Suppose that $p \leq q \leq r$ with $\text{arg}(r) \subseteq \text{arg}(q) \subseteq \text{arg}(p)$. Let $\tau = \langle d_1, \dots, d_n \rangle$, $\sigma_{p \rightarrow q}^-(\tau) = \langle d'_1, \dots, d'_m \rangle$, and $\sigma_{p \rightarrow r}^-(\tau) = \langle d''_1, \dots, d''_k \rangle$ with $k \leq m \leq n$. To prove the proposition, we derive $\langle d''_1, \dots, d''_k \rangle = \sigma_{q \rightarrow r}^-(\langle d'_1, \dots, d'_m \rangle)$. By

Definition 7 and $\arg(r) \subseteq \arg(q)$, $\{d''_1, \dots, d''_k\} \subseteq \{d'_1, \dots, d'_m\}$. Therefore, we have that $\sigma_{q \rightarrow r}^-(\langle d'_1, \dots, d'_m \rangle)$ is identical to $\langle d''_1, \dots, d''_k \rangle$. ■

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

3.2 Semantics

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

Definition 8 (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 $s_i \leq s_j$ for $s_i, s_j \in S$, then $I(s_i) \subseteq I(s_j)$,
 - (c) if $f \in F_n$ and $f: s_1 \times \dots \times s_n \rightarrow s \in \Omega$, then $I(f): I(s_1) \times \dots \times I(s_n) \rightarrow I(s)$,
 - (d) if $p \in P_n$ and $p: s_1 \times \dots \times s_n \in \Omega$, then $I(p): I(s_1) \times \dots \times I(s_n) \rightarrow 2^{U_{\mathcal{F}}}$,
 - (e) if $p \leq q$ for $p \in P_n$ and $q \in P_m$, then $I(p)(\tau) \subseteq I(q)(\sigma_{p \rightarrow q}^-(\tau))$,
 - (f) if $\psi \in \Psi_n$, then $I(\psi) \subseteq U_{\mathcal{F}}^n$,
 - (g) if $\psi \leq \phi$ 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 $U_{\mathcal{F}}^n$. In the improved semantics, the implication form $q(t_1, \dots, t_n) \leftarrow \{p(t_1, \dots, t_n)\}$ and the subpredicate relation $p \leq q$ can be interpreted differently, i.e., using the predicate hierarchy, it can be ensured that every $e_i \in I(p)(\tau)$ is included in $I(q)(\tau)$ for $p \leq q$, while the implication form $q(\dots) \leftarrow \{p(\dots)\}$ has a model such that $e_1 \in I(p)(\tau)$ implies $e_2 \in I(q)(\tau)$, even if $e_1 \neq e_2$. Note that the implication form does not indicate a concept hierarchy or subordinate-relation.

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. $p \sqcup q \leq r$ implies $I(p)(\tau_1) \cup I(q)(\tau_2) \subseteq I(r)(\tau)$ with $\tau = \sigma_{p \rightarrow r}^-(\tau_1) = \sigma_{q \rightarrow r}^-(\tau_2)$.
2. $r \leq p \sqcap q$ implies $I(r)(\tau) \subseteq I(p)(\sigma_{r \rightarrow p}^-(\tau)) \cap I(q)(\sigma_{r \rightarrow q}^-(\tau))$.

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

This proposition indicates that any two ground atoms with identical arguments $p(t_1, \dots, t_n)$ and $q(t_1, \dots, t_n)$ 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 $\alpha: V \rightarrow U$ where $\alpha(x: s) \in I(s)$. The variable assignment $\alpha[x: s/d]$ is defined by $(\alpha - \{(x: s, \alpha(x: s))\}) \cup \{(x: s, d)\}$. In other words, if $v = x: s$, then $\alpha[x: s/d](v) = d$, and otherwise $\alpha[x: s/d](v) = \alpha(v)$. 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 $(M, \Delta, \alpha[x: s/d])$ is simply denoted by $\mathcal{I}\alpha[x: s/d]$.

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

Definition 9 *Let $\mathcal{I} = (M, \Delta, \alpha)$ be a Σ -interpretation. The denotation function $\llbracket \cdot \rrbracket_{\alpha}: \mathcal{T} \rightarrow U$ is defined by the following:*

1. $\llbracket x: s \rrbracket_{\alpha} = \alpha(x: s)$,
2. $\llbracket f(t_1, \dots, t_n): s \rrbracket_{\alpha} = I(f)(\llbracket t_1 \rrbracket_{\alpha}, \dots, \llbracket t_n \rrbracket_{\alpha})$ with $f: s_1 \times \dots \times s_n \rightarrow s \in \Omega$,
3. $\llbracket p(t_1, \dots, t_n) \rrbracket_{\alpha} = I(p)(\llbracket t_1 \rrbracket_{\alpha}, \dots, \llbracket t_n \rrbracket_{\alpha})$ with $p: s_1 \times \dots \times s_n \in \Omega$.

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

Definition 10 (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 $\llbracket A \rrbracket_{\alpha} \cap \Delta \neq \emptyset$.
2. $\mathcal{I} \models \psi(A_1, \dots, A_n)$ iff $\mathcal{I} \models A_1, \dots, \mathcal{I} \models A_n$ and $(\llbracket A_1 \rrbracket_{\alpha} \times \dots \times \llbracket A_n \rrbracket_{\alpha}) \cap I(\psi) \neq \emptyset$.
3. $\mathcal{I} \models \{L_1, \dots, L_n\}$ iff $\mathcal{I} \models L_1, \dots, \mathcal{I} \models L_n$.
4. $\mathcal{I} \models L \leftarrow G$ iff for all $d_1 \in I(s_1), \dots, d_n \in I(s_n)$, $\mathcal{I}\alpha[x_1: s_1/d_1, \dots, x_n: s_n/d_n] \models G$ implies $\mathcal{I}\alpha[x_1: s_1/d_1, \dots, x_n: s_n/d_n] \models L$ where $\text{Var}(L \leftarrow G) = \{x_1: s_1, \dots, x_n: s_n\}$.

Let $F \in \mathcal{A} \cup \mathcal{M} \cup \mathcal{G} \cup \mathcal{C}$. 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 $\mathcal{I} \models F$. The following lemma implies that if \mathcal{I} satisfies a clause (an extended Horn clause) $L \leftarrow G$, it also satisfies any sorted ground clause of $L \leftarrow G$.

Lemma 1 *Let $\mathcal{I} = (M, \Delta, \alpha)$ with $M = (U, U_{\mathcal{F}}, I)$ be a Σ -interpretation, $L \leftarrow G$ be a clause in \mathcal{C} , and θ be a sorted substitution for $L \leftarrow G$. If $\mathcal{I} \models L \leftarrow G$, then $\mathcal{I} \models (L \leftarrow G)\theta$.*

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 11 (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 $\mathcal{K} \vdash l: C$) 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, $\mathcal{K} \vdash l: (L \leftarrow G)\theta$ and l is incremented.*
- **Cut rule:** *Let $L \leftarrow G$ and $L' \leftarrow G' \cup \{L\}$ be ground clauses. If $\mathcal{K} \vdash l_1: L \leftarrow G$ and $\mathcal{K} \vdash l_2: L' \leftarrow G' \cup \{L\}$, then $\mathcal{K} \vdash l_2: L' \leftarrow G' \cup G'$.*
- **Predicate hierarchy rule:** *Let $p(t_1, \dots, t_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1: p(t_1, \dots, t_n) \leftarrow G$ and $p \leq q$, then $\mathcal{K} \vdash l_1: q(t'_1, \dots, t'_m) \leftarrow G$ where $\sigma_{p \rightarrow q}^-(t_1, \dots, t_n) = \langle t'_1, \dots, t'_m \rangle$.*
- **Meta-predicate hierarchy rule:** *Let $\psi(A_1, \dots, A_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1: \psi(A_1, \dots, A_n) \leftarrow G$ and $\psi \leq \phi$, then $\mathcal{K} \vdash l_1: \phi(A_1, \dots, A_n) \leftarrow G$.*
- **Fact derivation rule:** *Let $\psi(A_1, \dots, A_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1: \psi(A_1, \dots, A_n) \leftarrow G$, then $\mathcal{K} \vdash l: A_i \leftarrow G$ with $1 \leq i \leq n$ and l is incremented.*

We simply write $\mathcal{K} \vdash l: L$ if $\mathcal{K} \vdash l: L \leftarrow$. 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.

We now show the soundness and completeness of the extended Horn-clause calculus in the class of Σ -models. First, the soundness of the Horn-clause calculus is proved in the usual manner.

Theorem 1 (Soundness of Horn-Clause Calculus) *Let \mathcal{K} be a knowledge base and L be a ground atom or meta-atom. If $\mathcal{K} \vdash l: L$, then $\mathcal{K} \models L$.*

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 $\mathcal{K} \vdash_{\psi(A_1, \dots, A_n)} l: A_i$ if a labeled atom $l: A_i$ is directly derived from a labeled meta-atom $l_1: \psi(A_1, \dots, A_n)$ using the fact derivation rule. Let $L \leftarrow G$ be a clause. We define $\text{ground}(L \leftarrow G)$ as the set of sorted ground clauses for $L \leftarrow G$. We define $\text{ground}(\mathcal{K}) = \bigcup_{L \leftarrow G \in \mathcal{K}} \text{ground}(L \leftarrow G)$ as the set of sorted ground clauses for all $L \leftarrow G$ in \mathcal{K} .

Definition 12 (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. $U_{\mathcal{F}, H} = \mathbb{N} - \{l \in \mathbb{N} \mid \text{ground}(\mathcal{K}) \vdash l: L \leftarrow G \ \& \ L \in \mathcal{MA}\}$,
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 $f: s_1 \times \dots \times s_n \rightarrow s \in \Omega$, then $I_H(f)(t_1, \dots, t_n) = f(t_1, \dots, t_n): s$ where $t_1 \in I_H(s_1), \dots, t_n \in I_H(s_n)$,
 - (c) if $p \in P_n$ and $p: s_1 \times \dots \times s_n \in \Omega$, then $I_H(p)(\tau) = \bigcup_{q \leq p} \{l \in U_{\mathcal{F}, H} \mid \text{ground}(\mathcal{K}) \vdash l: q(\tau')\}$ with $\sigma_{q \rightarrow p}^-(\tau') = \tau$,
 - (d) if $\psi \in \Psi_n$, then $I_H(\psi) = \bigcup_{\phi \leq \psi} \{(l_1, \dots, l_n) \in U_{\mathcal{F}, H}^n \mid \text{for every } 1 \leq i \leq n, \text{ground}(\mathcal{K}) \vdash_{\phi(A_1, \dots, A_n)} l_i: A_i\}$.

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} , $\Delta_H = \bigcup_{\substack{p \in P \\ \tau \in \mathcal{T}_{0, s_1} \times \dots \times \mathcal{T}_{0, s_n}}} I_H(p)(\tau)$ with $p: s_1 \times \dots \times s_n \in \Omega$ is a valuation of facts on M_H , and α is a variable assignment on M_H .

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

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

Proof. (1) (\Rightarrow) Let $L \leftarrow G$ be a clause with $\text{Var}(L \leftarrow G) = \{x_1: s_1, \dots, x_h: s_h\}$. Let $\theta = \{x_1: s_1/t_1, \dots, x_h: s_h/t_h\}$ be a sorted substitution such that $(L \leftarrow G)\theta \in \text{ground}(L \leftarrow G)$. By Definition 12, we have $t_1 \in I_H(s_1), \dots, t_h \in I_H(s_h)$ for \mathcal{I}_H . Hence, $\mathcal{I}_H \models (L \leftarrow G)\theta$. (\Leftarrow) Let θ be a sorted ground substitution for $L \leftarrow G$. So, $(L \leftarrow G)\theta \in \text{ground}(L \leftarrow G)$. By the assumption, $\mathcal{I}_H \models (L \leftarrow G)\theta$. By Definition 12, for all $t_1 \in I_H(s_1), \dots, t_h \in I_H(s_h)$, $\mathcal{I}\alpha[x_1: s_1/t_1, \dots, x_h: s_h/t_h] \models G$ implies $\mathcal{I}\alpha[x_1: s_1/t_1, \dots, x_h: s_h/t_h] \models L$. So, $\mathcal{I}_H \models L \leftarrow G$.

(2) We show that \mathcal{I}_H is a Σ -interpretation. By Definition 12, the conditions 1,2,3-(a), (b), and (c) of Σ -models (in Definition 8) hold. By the conditions 3-(c) and 3-(d) in Definition 12, the conditions 3-(d) and 3-(f) in Definition 8 hold, respectively. Moreover, let us show the conditions 3-(e) and 3-(g) in Definition 8. Let $p \leq q$ and $l \in I_H(p)(\tau)$. Then, by the Herbrand model, there is a sub-predicate r of p ($r \leq p$) such that $\text{ground}(\mathcal{K}) \vdash i: r(\tau'')$ with $\sigma_{r \rightarrow p}(\tau'') = \tau$. Because of $r \leq q$ and $\sigma_{p \rightarrow q}(\sigma_{r \rightarrow p}(\tau'')) = \sigma_{r \rightarrow q}(\tau'')$ in Proposition 1, we have $l \in I_H(q)(\sigma_{p \rightarrow q}(\tau))$ by Definition 12. This satisfies the condition 3-(e). Let $\psi \leq \phi$ and $(l_1, \dots, l_n) \in I_H(\psi)$. Then, by the Herbrand model, there is a sub-meta-predicate ψ_0 of ψ ($\psi_0 \leq \psi$) such that $\text{ground}(\mathcal{K}) \vdash l_1: A_1, \dots, \text{ground}(\mathcal{K}) \vdash l_n: A_n$, and $\text{ground}(\mathcal{K}) \vdash l: \psi_0(A_1, \dots, A_n)$. By $\psi_0 \leq \phi$ and Definition 12, we have $(l_1, \dots, l_n) \in I_H(\phi)$, and so this follows the condition 3-(g).

Next, we prove that \mathcal{I}_H satisfies \mathcal{K} . Let $L \leftarrow G \in \mathcal{K}$. So we want to show $\mathcal{I}_H \models \text{ground}(L \leftarrow G)$. Let $L' \leftarrow G' \in \text{ground}(L \leftarrow G)$. So, there is a sorted ground substitution θ such that $(L \leftarrow G)\theta = L' \leftarrow G'$. Suppose $\mathcal{I}_H \models \{L_1, \dots, L_h\}\theta$ where $G = \{L_1, \dots, L_h\}$. If $L_i\theta$ is of the form $q(t'_1, \dots, t'_m)$, then $\llbracket q(t'_1, \dots, t'_m) \rrbracket_\alpha = I(q)(\llbracket t'_1 \rrbracket_\alpha, \dots, \llbracket t'_m \rrbracket_\alpha) (= I_H(q)(t'_1, \dots, t'_m)) \subseteq \Delta_H$. So, the condition 3-(c) in Definition 12 implies $\text{ground}(\mathcal{K}) \vdash l: p(t_1, \dots, t_n)$ with $\sigma_{p \rightarrow q}(\langle t_1, \dots, t_n \rangle) = \langle t'_1, \dots, t'_m \rangle$ and $p \leq q$. By the sorted substitution rule, we have $\mathcal{K} \vdash l: p(t_1, \dots, t_n)$, and therefore, we obtain $\mathcal{K} \vdash l: q(t'_1, \dots, t'_m)$ by the predicate hierarchy rule. If $L_i\theta$ is of the form $\phi(A_1, \dots, A_n)$, then the condition 3-(d) in Definition 12 implies $\text{ground}(\mathcal{K}) \vdash l: \psi(A_1, \dots, A_n)$ with $\psi \leq \phi$. By the sorted substitution rule, we have $\mathcal{K} \vdash l: \psi(A_1, \dots, A_n)$. Hence, we obtain $\mathcal{K} \vdash l: \phi(A_1, \dots, A_n)$ by the meta-predicate rule. So, $\mathcal{K} \vdash l_1: L_1\theta, \dots, \mathcal{K} \vdash l_h: L_h\theta$. Since $L \leftarrow G \in \mathcal{K}$, the sorted substitution rule derives $\mathcal{K} \vdash l: (L \leftarrow \{L_1, \dots, L_h\})\theta$. By applying the cut rule to them, $\mathcal{K} \vdash l: L\theta$, and therefore, $\text{ground}(\mathcal{K}) \vdash l: L\theta$. If $L\theta$ is an atom, then the condition 3-(c) in Definition 12 implies $\mathcal{I}_H \models L\theta$. If $L\theta$ is a meta-atom $\psi'(A'_1, \dots, A'_k)$, the condition 3-(d) in Definition 12 derives $(l'_1, \dots, l'_k) \in I_H(\psi')$ such that $\psi'' \leq \psi'$ and $\text{ground}(\mathcal{K}) \vdash l: \psi''(A'_1, \dots, A'_k)$. By the fact derivation rule, we have $\text{ground}(\mathcal{K}) \vdash l'_1: A'_1, \dots, \text{ground}(\mathcal{K}) \vdash l'_k: A'_k$. By the condition 3-(c) in Definition 12, we have $\mathcal{I}_H \models A'_1, \dots, \mathcal{I}_H \models A'_k$. Since $(l_1, \dots, l_k) \in (\llbracket A'_1 \rrbracket_\alpha \times \dots \times \llbracket A'_k \rrbracket_\alpha) \cap I_H(\psi')$, we obtain $\mathcal{I}_H \models \psi'(A'_1, \dots, A'_k)$. Therefore, $\mathcal{I}_H \models (L \leftarrow G)\theta$. By the first statement in this lemma, we have $\mathcal{I}_H \models L \leftarrow G$. \blacksquare

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

Theorem 2 (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. If $\mathcal{K} \models L$, then $\mathcal{K} \vdash l: L$.*

Proof. Suppose $\mathcal{K} \models L$. By Lemma 2, there exists a Herbrand interpretation \mathcal{I}_H that satisfies \mathcal{K} . So, we have $\mathcal{I}_H \models L$ by the assumption. According to the conditions 3-(c) and 3-(d) in Definition 12, there exists $\mathcal{K} \vdash l: q(\tau')$ or $\mathcal{K} \vdash l: \psi(A_1, \dots, A_n)$ that further derives $\mathcal{K} \vdash l: L$ by the predicate hierarchy

rule or the meta-predicate hierarchy rule. ■

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

Theorem 3 (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 $Con(\mathcal{K}) = \{L \leftarrow G \mid \mathcal{K} \vdash l: L \leftarrow G\}$ 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 $\mathcal{K} \vdash l: L\theta$ is (single) EXPTIME-complete (w.r.t. the length of \mathcal{K}).

Proof. Let Σ be a function-free sorted signature (i.e., only constants such as 0-ary functions in F_0 are allowed). Suppose $|\mathcal{K}| = m$ and $|\Sigma| = k$. Then, $|Var(\mathcal{K})| \leq m$ and $|T_0^\Sigma| \leq k$. Let us count the number of derivation steps involved when applying inference rules of the Horn-clause calculus. In order to reduce the number of rule applications to a finite value the inference rules are applied in the following two steps: (i) All ground clauses in $ground(\mathcal{K})$ are derived by applying the sorted substitution rule. This computation is bounded by the number of ground clauses in $ground(\mathcal{K})$, i.e., 2^{m^k} . (ii) The cut rule, predicate hierarchy rule, meta-predicate hierarchy rule, and fact derivation rule are applied to the ground clauses derived in step (i) such that the cut rule is applied to $\mathcal{K} \vdash l_1: L_1 \leftarrow, \dots, \mathcal{K} \vdash l_h: L_h \leftarrow$ and $\mathcal{K} \vdash l: L \leftarrow \{L_1, \dots, L_h\}$, and the other rules are applied to $\mathcal{K} \vdash l: L \leftarrow$. These restrictions preserve the completeness of the calculus using the proof of Theorem 2. We set $CS_0 = \{L \mid L \leftarrow \in ground(\mathcal{K})\}$ and operate $CS_{i+1} = CS_i \cup \{L'\}$ for each rule application deriving L' . The CS_i is used to provide the condition where each rule can be applied if the conclusion L is not in CS_i . The cardinality of CS_i is bounded by 2^{m^k} . Therefore, the derivation in (i) and (ii) is computed in 2^{m^k+1} steps.

The hardness of the derivation problem is obtained from the program complexity of DATALOG [5]. ■

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 $\mathcal{A}^\mathcal{V} = \{X: p(t_1, \dots, t_n) \mid X \in \mathcal{V}, p(t_1, \dots, t_n) \in \mathcal{A}\}$. We call the form $X: p(t_1, \dots, t_n)$ a predicate variable atom.

Definition 13 (Queries) Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with a well-arranged argument declaration Λ , and let $\mathcal{MA}^\mathcal{V} = \{\psi(A_1^+, \dots, A_n^+) \mid \psi \in \Psi_n, A_1^+, \dots, A_n^+ \in \mathcal{A} \cup \mathcal{A}^\mathcal{V}\}$ be the set of meta-atoms with predicate variables. The set \mathcal{Q} of queries is defined by that if $L_1, \dots, L_h \in \mathcal{A} \cup \mathcal{A}^\mathcal{V} \cup \mathcal{MA}^\mathcal{V}$, then $\{L_1, \dots, L_h\} \in \mathcal{Q}$.

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

Definition 14 (Substitutions for Predicate Variables)

A substitution for predicate variables is a partial function $\delta: \mathcal{A}^\mathcal{V} \rightarrow \mathcal{A}$ such that $\delta(X: q(t'_1, \dots, t'_m)) \in \mathcal{A}_q$ 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 $X: q(\tau)$. A substitution δ is a most specific substitution for a predicate variable atom $X: q(\tau)$ if $\delta(X: q(\tau)) = p(\tau')$ with $\sigma_{p \rightarrow q}^-(\tau') = \tau$ and there is no other substitution δ' such that $\delta'(X: q(\tau)) = r(\tau'')$ with $\sigma_{r \rightarrow q}^-(\tau'') = \tau$ and $r \leq p$.

Definition 15 (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 $Query: \mathcal{Q} \rightarrow \{yes, no\}$ is defined by the following rule.

- (i) If there exists $\mathcal{K} \vdash l: Q\delta\theta$ 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. In the first step, atoms are substituted for predicate variable atoms in a query Q along with the predicate hierarchy. In the second step, predicate and meta-predicate assertions in the substituted query $Q\delta$ are derived using the Horn-clause calculus.

Theorem 4 (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.

Proof. Suppose $|\mathcal{K}| = m$ and $|\Sigma| = k$. By using the Horn-clause calculus, we can obtain $Con^{atom}(\mathcal{K}) = \{L \mid \mathcal{K} \vdash l: L\}$ such that $|Con^{atom}(\mathcal{K})| \leq 2^{m^k+1}$ according to the proof of Corollary 1. The answer to $Query(Q)$ is computed by searching all the elements in $Con^{atom}(\mathcal{K})$. If the answer is *yes*, then all the substitutions δ and θ that satisfy the query $\mathcal{K} \vdash l: \theta\delta Q$ can be generated by the

search. The search steps are bounded by 2^{m^k+1} . ■

The proof of 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 $\text{Query}(Q)$ 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 16 (Naive Argument Restructuring) *Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \dots, d_n \rangle$ 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 $\sigma_{p \rightarrow q}^+(\langle d_1, \dots, d_n \rangle) = \langle d'_1, \dots, d'_m \rangle$ such that*

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

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

We refine the definition of Σ -models such a way that every argument elimination $\sigma_{p \rightarrow q}^-$ is replaced with an argument restructuring $\sigma_{p \rightarrow q}^+$. The satisfiability relation \models is denoted by \models_{σ^+} 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 17 (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 $\mathcal{K} \vdash_{\sigma^+} l: C$) in the sorted Horn-clause calculus is extended by replacing the predicate hierarchy rule with the following rule:*

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

This extension preserves the soundness of the extended Horn-clause calculus as follows.

Theorem 5 (Soundness of Extended Horn-Clause Calculus) *Let \mathcal{K} be a knowledge base and L be a ground atom or meta-atom. If $\mathcal{K} \vdash_{\sigma^+} L$, then $\mathcal{K} \models_{\sigma^+} L$.*

An atom A_1 is a parent of another atom A_2 if $\mathcal{K} \vdash_{\sigma^+} l: A_2 \leftarrow G$ is derived from $\mathcal{K} \vdash_{\sigma^+} l: A_1 \leftarrow G$ 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 $p(t_1, \dots, t_n)$ with $p: \langle a_1, \dots, a_n \rangle \in \Pi$. 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 $l: A$ is defined by $AL(l: A) = \{(a, t) \mid A[a, t]\} \cup \{(a, t) \mid A'[a, t] \text{ is an ancestor of } A\}$.

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 18 (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 $\langle d_1, \dots, d_n \rangle$ 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 $\sigma_{p \rightarrow q}^(\langle t_1, \dots, t_n \rangle) = \langle t'_1, \dots, t'_m \rangle$ such that*

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

where $p: \langle a_1, \dots, a_n \rangle$ and $q: \langle a'_1, \dots, a'_m \rangle$ in Π , $q: s_1 \times \dots \times s_m$ in Ω , 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, \dots, t_n in a labeled atom $l: p(t_1, \dots, t_n)$ 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 $\sigma_{q \rightarrow r}^*(\sigma_{p \rightarrow q}^*(\tau)) = \sigma_{p \rightarrow r}^*(\tau)$.

Proof. Suppose that $p \leq q \leq r$ with $|\arg(p)| = n$, $|\arg(q)| = m$, and $|\arg(r)| = k$. Let $\tau = \langle d_1, \dots, d_n \rangle$, $\sigma_{p \rightarrow q}^*(\tau) = \langle d'_1, \dots, d'_m \rangle$, and $\sigma_{p \rightarrow r}^*(\tau) = \langle d''_1, \dots, d''_k \rangle$. We show $\langle d'_1, \dots, d'_k \rangle = \langle d'''_1, \dots, d'''_k \rangle$ where $\sigma_{q \rightarrow r}^*(\langle d'_1, \dots, d'_m \rangle) = \langle d'''_1, \dots, d'''_k \rangle$. Let $d'''_i \in \{d'''_1, \dots, d'''_k\}$. If $(a_i, d'''_i) \in AL(l: p(\tau))$, then $d'''_i = d'_i$ due to $(a_i, d'''_i) \in AL(l: q(d'_1, \dots, d'_m))$. This is because $AL(l: p(\tau)) \subseteq AL(l: q(d'_1, \dots, d'_m))$ by Definition 18. If $(a_i, d'''_i) \notin AL(l: p(\tau))$, then $d'''_i = c_{l,a_i} : s_i$. Also, if $(a_i, d'''_i) \notin AL(l: q(d'_1, \dots, d'_m))$, then $d'''_i = c_{l,a_i} : s_i$. Otherwise, the same constant $c_{l,a_i} : s_i$ must be added in the operation $\sigma_{p \rightarrow q}^*(\tau)$, and therefore $d'_i = c_{l,a_i} : s_i$. Hence, we have that $\sigma_{q \rightarrow r}^*(\langle d'_1, \dots, d'_m \rangle) = \langle d'''_1, \dots, d'''_k \rangle$ is identical to $\langle d''_1, \dots, d''_k \rangle$. ■

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

Theorem 6 (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. If $\mathcal{K} \models_{\sigma^+} L$, then $\mathcal{K} \vdash_{\sigma^*} L$.

Similar to the proof of Theorem 2, we can prove Theorem 6 by using Proposition 4. 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 7 (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 .

Proof. We can give the knowledge base that contains the fact $p(c: s_2)$ and the rule $p(y: s_2) \leftarrow \{q(x: s_1, y: s_2)\}$ in the sorted signature $\Sigma = (\{s, \top\}, \{p, q, \top\}, \emptyset, \Omega, \leq^*)$ such that $\Omega = \{c: \rightarrow s, c_1: \rightarrow s, c_2: \rightarrow s, \dots\}$ and

$\leq = \{ p \leq q \}$. This knowledge base generates new constants infinitely, and so the calculus does not terminate. \blacksquare

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 $L = \psi(A_1, \dots, A_m)$ with $A_i = p(\tau)$ for some $1 \leq i \leq m$.

Definition 19 (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, and let a, a' be argument names. Then, \mathcal{K} contains a path from a in predicate p to a' in predicate q if one of the following conditions holds:*

1. $p \leq q$ with $a = a'$ or $a \notin \arg(q)$,
2. $L_q[a', x: s] \leftarrow G \in \mathcal{K}$ where $L_p[a, x: s] \in G$, and
3. \mathcal{K} contains two paths from a in predicate p to a'' in predicate r and from a'' in predicate r to a' in predicate q .

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

Definition 20 (Safe Knowledge Bases) *A knowledge base \mathcal{K} is safe if*

1. $\text{Var}(L) \subseteq \text{Var}(G)$ for every clause $L \leftarrow G$ in \mathcal{K} , and
2. \mathcal{K} contains no path from a in predicate p to a' in predicate q such that $q \leq p$, $a \neq a'$, $a \notin \arg(q)$, and $a \in \arg(p)$.

We give an example of unsafe knowledge bases that lead to undecidable reasoning with argument restructurings.

Example 2 *Given the sorted signature $\Sigma = (S, P, \Psi_n, \Omega, \leq^*)$ such that*

$$\begin{aligned} S &= \{ s_1, s_2, \top \}, \\ P &= \{ p, q, \top \}, \\ \Psi_n &= \emptyset, \\ \Omega &= \{ c: \rightarrow s_1, p: s_1 \times s_2, q: s_1 \}, \\ \leq &= \{ s_1 \leq s_2 \} \cup \{ q \leq p \} \end{aligned}$$

with the argument declaration $\Lambda = (AN, \Pi)$ such that

$$\begin{aligned} AN &= \{ a_1, a_2 \}, \\ \Pi &= \{ p: \langle a_1, a_2 \rangle, q: \langle a_1 \rangle \}, \end{aligned}$$

we can construct the unsafe knowledge base

$$\mathcal{K} = \{ q(c: s_1) \leftarrow, q(y: s_2) \leftarrow \{ p(x: s_1, y: s_2) \} \}.$$

Lemma 3 *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.*

Proof. Let $Fun_0(\mathcal{K})$ is the set of constants occurring in a knowledge base \mathcal{K} . We define $Con_{new}(\mathcal{K}) = \{\mathcal{C} \in Con(\mathcal{K}) \mid Fun_0(\mathcal{C}) \cap F_{0,new} \neq \emptyset\}$. We denote by $Lab(\mathcal{K})$ the set of labels occurring in a knowledge base \mathcal{K} . Let $F' = \{c_{l,a} : s \mid l \in Lab(Con(\mathcal{K}) - Con_{new}(\mathcal{K})), a \in AN\}$. Then, F' is finite because $Lab(Con(\mathcal{K}) - Con_{new}(\mathcal{K}))$ and AN are finite.

Suppose that a new constant c (introduced in a label-based argument restructuring) does not belong to F' . The constant c must be indexed by $c_{l',a} : s$ such that $l' \notin Lab(Con(\mathcal{K}) - Con_{new}(\mathcal{K}))$. If $c_{l',a} : s$ is generated, then there is a clause $l' : L_p[a', c'] \leftarrow G$ in $Con(\mathcal{K})$ such that $p \leq q$, $a' \in \arg(p)$, and $a' \notin \arg(q)$. This is because $c_{l',a} : s$ must be introduced in the predicate hierarchy rule, i.e., for $p \leq q$, $L_p[a', c']$ derives $L_q[a, c_{l',a} : s]$ where $c_{l',a}$ is the new constant. However, every safe knowledge base does not contain any path from $L_p[a', c']$ to derives $L_q[a'', c']$ that derives $L_q[a, c_{l',a} : s]$. This is contradictory to the assumption. So, it must be sure $c \in F'$, and therefore $F_{0,new} = F'$. ■

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

Theorem 8 (Termination of Extended Horn-Clause Calculus)

Let \mathcal{K} be a safe knowledge base in a sorted signature Σ . Then, the extended Horn-clause calculus with label-based argument restructuring terminates if Σ is function-free.

By Lemma 3, we can consider only finite sorted signatures if knowledge bases are safe. So, similar to the proof of Theorem 3, Theorems 8 and 3 can be shown.

Corollary 3 (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 $\mathcal{K} \vdash_{\sigma^} l : L\theta$ is (single) EXPTIME-complete (w.r.t. the length of \mathcal{K}).*

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 9 (Termination of Extended Query System)

Let \mathcal{K} be a safe knowledge base in a sorted signature Σ . Then, the extended query system terminates if Σ is function-free.

Proof. This can be proven by Theorem 8. ■

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

Corollary 4 (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 $\text{Query}(Q)$ is (single) EXPTIME-complete (w.r.t. the length of \mathcal{K}).

7 Conclusions and Future Work

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.

We plan to further extend the meta-predicates in the proposed order-sorted logic in order to represent relationships between objects and events. For example, the assertion $\text{kills}(\text{tsunami}(c_2 : \text{country}), \text{John} : \text{person})$ contains an event and an object in a causal relationship. We shall attempt to operate argument structures of meta-predicates in a meta-predicate hierarchy. Such an expression is intuitively conceivable in natural languages; however, it is not easy to incorporate the expression into logical languages because the binary relation kills contains both first- and second-order entities.

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. Groszof, 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 Edition)*. 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. Groszof, 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. K. Kaneiwa and P.H.P. Nguyen. Decidable order-sorted logic programming for ontologies and rules with argument restructuring. In *Proceedings of the 8th International Semantic Web Conference (ISWC 2009)*, 2009.
17. 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.
18. J. W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.
19. M. Manzano. Introduction to many-sorted logic. In *Many-sorted Logic and its Applications*, pages 3–86. John Wiley and Sons, 1993.
20. B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.
21. Boris Motik. On the Properties of Metamodeling in OWL. *Journal of Logic and Computation*, 17(4):617–637, 2007.
22. 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.
23. 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/>.
 24. R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics*, 3(1):41–60, 2005.
 25. M. Schmidt-Schauss. *Computational Aspects of an Order-Sorted Logic with Term Declarations*. Springer-Verlag, 1989.
 26. R. Socher-Ambrosius and P. Johann. *Deduction Systems*. Springer-Verlag, 1996.
 27. 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.