Knowledge and Information Systems (2012): 1–31
© 2012 Springer-Verlag London Ltd.

An Order-Sorted Query System for Sort, Predicate, and Meta-Predicate Hierarchies

Ken Kaneiwa[1]

Philip H.P. Nguyen[2]

[1]Department of Electrical Engineering and Computer Science, Iwate University
4-3-5 Ueda, Morioka, Iwate 020-8551, Japan
kaneiwa@cis.iwate-u.ac.jp;
[2]Attorney-General's Department, Government of South Australia
30 Wakefield St, Adelaide, SA 5000, Australia
philip.nguyen@sa.gov.au

Abstract. This paper presents a decidable order-sorted query system for reasoning between ontologies and rules. We describe order-sorted logic programming with sort, predicate, and meta-predicate hierarchies (OSL_{3h}) that derives 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 in OSL_{3h} 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.

Keywords: Semantic Web; Query system; Order-sorted logic; Meta-predicate hierarchy; Logic programming

1. Introduction

Received Nov 14, 2011
Revised Mar 24, 2012
Accepted Apr 26, 2012
[1] This paper is an extended version of (Kaneiwa and Nguyen, 2009), containing proofs as well as some additional definitions, theorems, lemmas, and examples.

In the Semantic Web context, conceptual knowledge representation and reasoning (Woods and Schmolze, 1992) have been studied for modeling ontologies in OWL (Web Ontology Language) (Patel-Schneider, Hayes and Horrocks, 2004). 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 (Aït-Kaci and Nasr, 1986; Cohn, 1989; Socher-Ambrosius and Johann, 1996; Kaneiwa and Mizoguchi, 2005; Kaneiwa and Mizoguchi, 2009) (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 (Kaneiwa, 2004), order-sorted logic programming was extended by introducing such a predicate hierarchy. Furthermore, the conceptual structure theory (Nguyen, Kaneiwa, Corbett and Nguyen, 2007) 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 (Motik, 2007). Further, a concept hierarchy can be enhanced by concepts named using natural language words because the words contain higher-order expressions. 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 complexity (Dantsin, Eiter, Gottlob and Voronkov, 1997) has been studied in the area of automated deduction. It is known that full logic programming is undecidable, but function-free logic programming is EXPTIME-complete with respect to the length of a program. In addition, non-recursive logic programming with functions is NEXPTIME-complete.

However, SWRL (Semantic Web Rule Language) (Horrocks, Patel-Schneider, Boley, Tabet, Grosof and Dean, 2004), a combination of OWL and RuleML, leads to undecidable reasoning between ontologies and rules (as shown in (Horrocks and Patel-Schneider, 2004)). Several decidable fragments for combining ontologies and rules, such as DL-safe (Hitzler and Parsia, 2009; Rosati, 2005; Motik, Sattler and Studer, 2005), DLP (Description Logic Programs) (Grosof, Horrocks, Volz and Decker, 2003), and the rule-based language ELP (Krötzsch, Rudolph and Hitzler, 2008) (related to the tractable language profile OWL 2 EL (Motik, Grau, Horrocks, Wu, Fokoue, Lutz, 2009)), 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 (Chen

[1] http://ruleml.org/

and Kifer, 1995), 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 the complexity of HILOG is harder than the EXPTIME complexity of DATALOG. Unfortunately, in most cases, higher-order logic programming (Jouannaud and Okada, 1991) 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 ($\mathrm{OSL}_{3\mathrm{h}}$). We define the syntax and semantics of three kinds of hierarchies (sort hierarchy, predicate hierarchy, and meta-predicate hierarchy) in $\mathrm{OSL}_{3\mathrm{h}}$. We develop the order-sorted Horn-clause calculus (Hanus, 1992), 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.

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 $\mathrm{OSL}_{3\mathrm{h}}$. 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 give a case study example of reasoning on ontologies and rules in the Semantic Web context, and in Section 8, we compare our approach with several studies related to an integration of ontologies and rules. In Section 9, 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

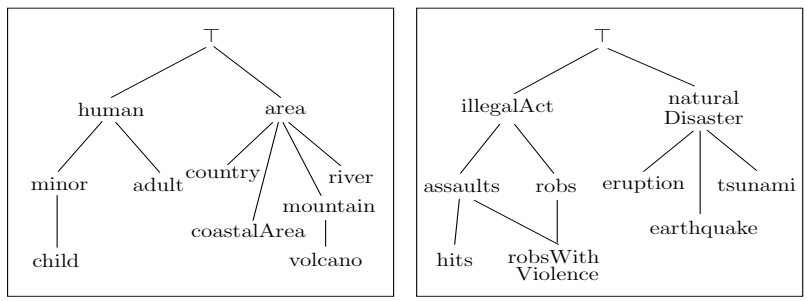

Fig. 1. Sort and predicate hierarchies

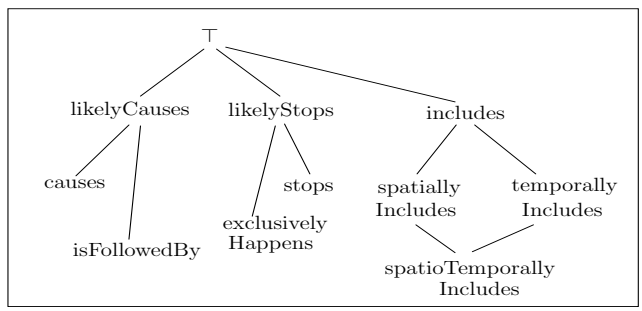

Fig. 2. A meta-predicate hierarchy

`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 order-sorted logic programming.

3. Order-Sorted Logic with Meta-Predicates

First, we define the syntax and semantics of $\mathrm{OSL}_{3\mathrm{h}}$ with sort, predicate, and meta-predicate hierarchies (an extension of (Socher-Ambrosius and Johann, 1996; Schmidt-Schauss, 1989; Manzano, 1993)).

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 for each natural number n
3. P_n: a countable set of n-ary predicate symbols for each natural number n
4. Ψ_n: a countable set of n-ary meta-predicate symbols for each natural number n
5. $\leftarrow, \{, \}$: the connective and auxiliary symbols
6. V_s: an infinite set of variables $x{:}\, s, y{:}\, s, z{:}\, s, \ldots$ 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 with bounded arity such that

(a) if $f \in F_n$ is an n-ary function symbol, then there is a unique function declaration of the form $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$ where $s_1, \ldots, s_n, s$ are sort symbols, and
(b) if $p \in P_n$ is an n-ary predicate symbol, then there is a unique predicate declaration of the form $p\colon s_1 \times \cdots \times s_n \in \Omega$ where $s_1, \ldots, s_n$ are sort symbols.

The predicate hierarchy includes predicates with different argument structures, e.g., a binary predicate can be a subpredicate of a unary predicate. In contrast to that, the meta-predicate hierarchy only contains meta-predicates with a fixed arity. The meta-predicates with various arities are useful for describing ontologies but we do not introduce them in this work in order to avoid to increase the complexity of reasoning on meta-predicate hierarchies. 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 $c\colon \to s$.

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

$$
\begin{aligned}
S = \{\ & \mathit{child},\ \mathit{minor},\ \mathit{adult},\ \mathit{human},\ \mathit{bank},\ \mathit{mountain},\ \mathit{volcano},\ \mathit{river},\\
& \mathit{area},\ \mathit{coastalArea},\ \mathit{country},\ \top\ \},\\
P = \{\ & \mathit{hits},\ \mathit{violates},\ \mathit{robs},\ \mathit{robsWithViolence},\ \mathit{illegalAct},\ \mathit{eruption},\\
& \mathit{earthquake},\ \mathit{tsunami},\ \mathit{naturalDisaster},\ \top\ \},\\
\Psi_2 = \{\ & \mathit{causes},\ \mathit{isFollowedBy},\ \mathit{likelyCauses},\ \mathit{stops},\\
& \mathit{exclusivelyHappens},\ \mathit{likelyStops},\ \mathit{temporallyIncludes},\\
& \mathit{spatiallyIncludes},\ \mathit{spatioTemporallyIncludes},\ \mathit{includes},\ \top\ \},\\
\Omega = \{\ & \mathit{tom}\colon \to \mathit{minor},\ \mathit{john}\colon \to \mathit{adult},\ c_1\colon \to \mathit{country},\ c_2\colon \to \mathit{coastalArea},\\
& \mathit{hits}\colon \mathit{human}\times\mathit{human},\ \mathit{violates}\colon \mathit{human}\times\mathit{human},\ \mathit{illegalAct}\colon \mathit{human},\\
& \mathit{robsWithViolence}\colon \mathit{human} \times \mathit{bank} \times \mathit{human},\ \mathit{robs}\colon \mathit{human} \times \mathit{bank},\\
& \mathit{earthquake}\colon \mathit{area},\ \mathit{tsunami}\colon \mathit{area},\ \mathit{naturalDisaster}\colon \mathit{area}\ \ \},\\
\leq = \{\ & \mathit{child} \leq \mathit{minor},\ \mathit{minor} \leq \mathit{human},\ \mathit{adult} \leq \mathit{human},\ \mathit{river} \leq \mathit{area},\\
& \mathit{country} \leq \mathit{area},\ \mathit{volcano} \leq \mathit{mountain},\ \mathit{mountain} \leq \mathit{area},\\
& \mathit{coastalArea} \leq \mathit{area}\ \} \cup\\
& \{\ \mathit{hits} \leq \mathit{violates},\ \mathit{violates} \leq \mathit{illegalAct},\ \mathit{robs} \leq \mathit{illegalAct},\\
& \mathit{robsWithViolence} \leq \mathit{robs},\ \mathit{robsWithViolence} \leq \mathit{violates},\\
& \mathit{eruption} \leq \mathit{naturalDisaster},\ \mathit{tsunami} \leq \mathit{naturalDisaster},\\
& \mathit{earthquake} \leq \mathit{naturalDisaster}\ \} \cup\\
& \{\ \mathit{causes} \leq \mathit{likelyCauses},\ \mathit{isFollowedBy} \leq \mathit{likelyCauses},\\
& \mathit{stops} \leq \mathit{likelyStops},\ \mathit{exclusivelyHappens} \leq \mathit{likelyStops},\\
& \mathit{spatioTemporallyIncludes} \leq \mathit{temporallyIncludes},\\
& \mathit{spatioTemporallyIncludes} \leq \mathit{spatiallyIncludes},\\
& \mathit{temporallyIncludes} \leq \mathit{includes},\ \mathit{spatiallyIncludes} \leq \mathit{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 an 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 an 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 (Lloyd, 1987; Doets, 1994) are extended by meta-atoms $\psi(A_1, \ldots, A_n)$ that consist of meta-predicates ψ and atoms $A_1, \ldots, A_n$. Atoms are defined by sorted terms and predicates, and then meta-atoms are defined by atoms and meta-predicates.

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 $L \leftarrow G \in \mathcal{C}$.

For example, if $mike\colon person$, $mary\colon person \in \mathcal{T}_{person}$, $helps \in P_2$, and $helps\colon person \times person \in \Omega$, then $helps(mike\colon person, mary\colon person)$ is an atom in $\mathcal{A}$. We can use atoms as arguments of meta-predicates to assert n-ary relations ψ over atoms $A_1, \ldots, A_n$. 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 in Ψ_2. Both atoms (in $\mathcal{A}$) and meta-atoms (in $\mathcal{MA}$) can appear in the heads and bodies of extended Horn clauses. Let

$smolking(x\colon person), heavySmolker(x\colon person)$ be atoms in $\mathcal{A}$ and $likelyCauses(smolking(x\colon person), suffersFromLungCancer(x\colon person))$ be a meta-atom in $\mathcal{MA}$. Then, we have the following clause in $\mathcal{C}$:

$$likelyCauses(smoking(x\colon person), suffersFromLungCancer(x\colon person)) \leftarrow \{heavySmoker(x\colon person)\}$$

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 $\mathcal{T}_s$.

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

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

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 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 such that there is a unique argument structure of the form $p\colon \langle a_1, \ldots, a_n\rangle$ for each $p \in P_n$ where $a_1, \ldots, a_n \in AN$ and for every $i \neq j$, it holds $a_i \neq a_j$.

In addition to the sorted signature, the argument declaration $\Lambda = (AN, \Pi)$ determines the roles and structure of arguments in each predicate. In Λ, each argument name implies the role of an argument and the tuple of argument names $a\ (\in AN)$ in the form $p\colon \langle a_1, \ldots, a_n\rangle$ indicates the argument structure of predicate p.

Example 2. Let us consider the argument declaration $\Lambda = (\{sbj, obj_1, obj_2\}, \{prescribes\colon \langle sbj, obj_1, obj_2\rangle\})$ for a sorted signature Σ that contains the following predicate declaration:

$$prescribes\colon person \times medicine \times disease$$

where *prescribes* is a ternary predicate and *doctor*, *medicine*, and *disease* are sorts. Then, we have the atom $prescribes(nick\colon doctor, m_1\colon medicine, d_1\colon disease)$.

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. 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 $p \leq q$, $p\colon s_1 \times \cdots \times s_n \in \Omega$, and $q\colon s'_1 \times \cdots \times s'_m \in \Omega$, if $a_i = a'_j$ for $p\colon \langle a_1, \ldots, a_n\rangle$ and $q\colon \langle a'_1, \ldots, a'_m\rangle$ in Λ, then $s_i \leq s'_j$ in Σ. We denote an n-tuple of elements $d_1, \ldots, d_n$ by $\langle d_1, \ldots, d_n\rangle$.

Definition 7 (Argument Elimination). Let $\Sigma = (S, P, \Psi_n, \Omega, \leq)$ be a sorted signature with an argument declaration $\Lambda = (AN, \Pi)$, let $\langle d_1, \ldots, d_n\rangle$ be an n-tuple, and let $p \in P_n$, $q \in P_m$ with $Arg(q) \subseteq Arg(p)$ and $m \leq n$ for natural numbers n, m. 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

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

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

The argument elimination from predicate p to predicate q simply deletes some arguments of p if q does not have their names in the argument declaration.

Example 3. Let us consider the argument declaration $\Lambda = (\{sbj, obj_1, obj_2\}, \{prescribes\colon \langle sbj, obj_1, obj_2\rangle,\ gives\colon \langle sbj, obj_1\rangle\})$ for a sorted signature Σ that contains the following predicate declarations:

$$prescribes\colon person \times medicine \times disease$$

and

$$gives\colon person \times thing$$

where *prescribes* is a ternary predicate, *gives* is a binary predicates, and *doctor*, *medicine*, *thing*, and *disease* are sorts.

According to $Arg(gives) \subseteq Arg(prescribes)$, we have the argument elimination from *prescribes* to *gives* with

$$\sigma^-_{prescribes\to gives}(\langle nick\colon doctor, m_1\colon medicine, d_1\colon disease\rangle) = \langle nick\colon doctor, m_1\colon medicine\rangle$$

The argument eliminations will be used in the semantics and inference system of $\mathrm{OSL}_{3\mathrm{h}}$. 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 $\sigma^-_{q\to r}(\sigma^-_{p\to q}(\tau)) = \sigma^-_{p\to r}(\tau)$.

Proof. Suppose that $p \leq q \leq r$ with $\arg(r) \subseteq \arg(q) \subseteq \arg(p)$. Let $\tau = \langle d_1, \ldots, d_n\rangle$, $\sigma^-_{p\to q}(\tau) = \langle d'_1, \ldots, d'_m\rangle$, and $\sigma^-_{p\to r}(\tau) = \langle d''_1, \ldots, d''_k\rangle$ with $k \leq m \leq n$. To prove the proposition, we derive $\langle d''_1, \ldots, d''_k\rangle = \sigma^-_{q\to r}(\langle d'_1, \ldots, d'_m\rangle)$. By Definition 7 and $\arg(r) \subseteq \arg(q)$, $\{d''_1, \ldots, d''_k\} \subseteq \{d'_1, \ldots, d'_m\}$. Therefore, we

have that $\sigma^{-}_{q\to r}(\langle d'_1,\ldots,d'_m\rangle)$ is identical to $\langle d''_1,\ldots,d''_k\rangle$. ∎

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

Example 4. Let us consider that $eatsWith \leq eats \leq dines$ with $\arg(eatsWith) = \{sbj, obj, tool\}$, $\arg(eats) = \{sbj, obj\}$, and $\arg(dines) = \{sbj\}$. For the following two argument eliminations:

$$\langle tom\colon person, c_1\colon pasta\rangle = \sigma^{-}_{eatsWith\to eats}(\langle tom\colon person, c_1\colon pasta, c_2\colon fork\rangle)$$

and $\langle tom\colon person\rangle = \sigma^{-}_{eats\to dines}(\langle tom\colon person, c_1\colon pasta\rangle)$, we obtain

$$\langle tom\colon person\rangle = \sigma^{-}_{eatsWith\to dines}(\langle tom\colon person, c_1\colon pasta, c_2\colon fork\rangle)$$

3.2. Semantics

We define the semantics of $\mathrm{OSL}_{3\mathrm{h}}$ 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[2];
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 $f\colon s_1 \times \cdots \times s_n \to s \in \Omega$, 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 $p \leq q$ 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 $\psi \leq \phi$ 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 $U^n_{\mathcal{F}}$. In the improved semantics, the implication form $q(t_1,\ldots,t_n) \leftarrow \{p(t_1,\ldots,t_n)\}$ (in Definition 4) 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 $q(..) \leftarrow \{p(..)\}$ 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

[2] Each fact is a statement that implies an event or action.

$r \in P_k$ 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 $\tau = \sigma^-_{p\to r}(\tau_1) = \sigma^-_{q\to r}(\tau_2)$.
2. $r \leq p \sqcap q$ implies $I(r)(\tau) \subseteq I(p)(\sigma^-_{r\to p}(\tau)) \cap I(q)(\sigma^-_{r\to q}(\tau))$.

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

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 9. 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 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* $[\![A]\!]_\alpha \cap \Delta \neq \emptyset$.
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 $\mathcal{I}\alpha[x_1{:}\,s_1/d_1, \ldots, x_n{:}\,s_n/d_n] \models L$ where $Var\ (L \leftarrow G) = \{x_1{:}\,s_1, \ldots, x_n{:}\,s_n\}$.

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 $\mathcal{I} \models F$. The following lemma implies that if $\mathcal{I}$ satisfies a clause (an extended Horn clause) $L \leftarrow G$, it also satisfies any substituted expression $(L \leftarrow G)\theta$.

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 $\mathcal{I} \models (L \leftarrow G)\theta$.

Proof. Let $Var(L \leftarrow G) = \{x_1{:}\,s_1, \ldots, x_n{:}\,s_n\}$. By Definition 10, for all $d_1 \in I(s_1), \ldots, d_n \in I(s_n)$, $\mathcal{I}\alpha[x_1{:}\,s_1/d_1, \ldots, x_n{:}\,s_n/d_n] \models G$ implies $\mathcal{I}\alpha[x_1{:}\,s_1/d_1, \ldots, x_n{:}\,s_n/d_n] \models L$. Let $d'_1 \in I(s'_1), \ldots, d'_m \in I(s'_m)$ where $Var((L \leftarrow G)\theta) = \{y_1{:}\,s'_1, \ldots, y_m{:}\,s'_m\}$. Then, we have $[\![\theta(x_1{:}\,s_1)]\!]_{\alpha'} \in I(s_1), \ldots, [\![\theta(x_n{:}\,s_n)]\!]_{\alpha'} \in I(s_n)$ for $\alpha' = \alpha[y_1{:}\,s'_1/d'_1, \ldots, y_m{:}\,s'_m/d'_m]$ because $\theta(x_1{:}\,s_1), \ldots, \theta(x_n{:}\,s_n)$ are well-sorted terms of sorts $s_1, \ldots, s_n$. This derives $\mathcal{I} \models (L \leftarrow G)\theta$. ∎

4. Horn-Clause Calculus for Predicate Hierarchies

In this section, we extend the order-sorted Horn-clause calculus 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. We use $L[A] \leftarrow G$ to denote that an atom A occurs in L, e.g., $\psi(A, A')[A] \leftarrow G$ and $A[A] \leftarrow G$.

Definition 11 (Sorted Horn-Clause Calculus). Let C be a ground clause, $\mathcal{K}$ be a knowledge base, l be a non-negative integer variable, and l_1, l_2 be non-negative integers. 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 $L[p(t_1, \ldots, t_n)] \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1{:}\,L[p(t_1, \ldots, t_n)] \leftarrow G$ and $p \leq q$, then $\mathcal{K} \vdash l_1{:}\,L[q(t'_1, \ldots, t'_m)] \leftarrow G$ 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 $\mathcal{K} \vdash l_1{:}\,\psi(A_1, \ldots, A_n) \leftarrow G$ and $\psi \leq \phi$, then $\mathcal{K} \vdash l_1{:}\,\phi(A_1, \ldots, A_n) \leftarrow G$.

Fact derivation rule: Let $\psi(A_1, \ldots, A_n) \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash l_1{:}\,\psi(A_1, \ldots, A_n) \leftarrow G$, then $\mathcal{K} \vdash l{:}\,A_i \leftarrow G$ with $1 \leq i \leq n$ and l is incremented.

Initially, we set the variable $l = 0$. After a ground clause is derived by the sorted substitution rule or fact derivation rule (i.e., we obtain $\mathcal{K} \vdash 0{:}\,L \leftarrow G$ by the first rule application), the value of l is incremented. For example, consider the

following derivation process:

$$\dfrac{\dfrac{\dfrac{p(x\colon s_1, y\colon s_2) \leftarrow G \in \mathcal{K}}{0\colon p(t_1,t_2) \leftarrow G}\ \text{(Sorted sub.)}}{0\colon q(t_1) \leftarrow G}\ \text{(Predicate hie.)} \quad \dfrac{r(v\colon s_3) \leftarrow \{q(w\colon s_1)\} \in \mathcal{K}}{1\colon r(t_3) \leftarrow \{q(t_1)\}}\ \text{(Sorted sub.)}}{1\colon r(t_3) \leftarrow G}\ \text{(Cut)}$$

Note that the variable l is not changed by the predicate hierarchy rule, meta-predicate hierarchy rule, or cut rule. In order to identify each conclusion, it is incremented when a new fact (in the head of a clause) is derived, e.g., $p(t_1, t_2)$ and $r(t_3)$ are new facts but $q(t_1)$ is not in the derivation above. This is because $q(t_1)$ is an abstract expression of the fact $p(t_1, t_2)$. 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.

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 $\mathcal{K} \vdash l\colon L$, then $\mathcal{K} \models L$.

Proof. This is proven by induction on the height n of a derivation tree of $\mathcal{K} \vdash l\colon L$. Let $\mathcal{I} = (M, \Delta, \alpha)$ be a Σ-interpretation.

Base case: $n = 0$. We have $\mathcal{K} \vdash l\colon L$ if $L \leftarrow \in \mathcal{K}$. So, $\mathcal{I} \models \mathcal{K}$ implies $\mathcal{I} \models L$.

Induction step: $n > 0$. The ground atom or meta-atom L is derived by some applications of inference rules.

- Sorted substitution rule. We have $L \leftarrow G \in \mathcal{K}$ and θ is a sorted ground substitution for $L \leftarrow G$. So, $\mathcal{I} \models L \leftarrow G$. By Lemma 1, $\mathcal{I} \models (L \leftarrow G)\theta$.
- Cut rule. Because $\mathcal{K} \vdash l_1\colon L \leftarrow G$ and $\mathcal{K} \vdash l_2\colon L' \leftarrow G' \cup \{L\}$, by induction hypothesis, $\mathcal{I} \models L \leftarrow G$ and $\mathcal{I} \models L' \leftarrow G' \cup \{L\}$. Hence, $\mathcal{I} \models L \leftarrow G \cup G'$.
- Predicate hierarchy rule. Because $\mathcal{K} \vdash l_1\colon L[p(t_1, \ldots, t_n)] \leftarrow G$ and $p \leq q$, by induction hypothesis, $\mathcal{I} \models L[p(t_1, \ldots, t_n)] \leftarrow G$ where $I(p)(t_1, \ldots, t_n) \subseteq I(q)(\sigma^{-}_{p \to q}(t_1, \ldots, t_n))$, and if $\mathcal{I} \models G$, then $[\![q(t'_1, \ldots, t'_m)]\!]_\alpha \cap \Delta \neq \emptyset$ where $\sigma^{-}_{p \to q}(\langle t_1, \ldots, t_n \rangle) = \langle t'_1, \ldots, t'_m \rangle$. Thus, $\mathcal{I} \models L[q(t'_1, \ldots, t'_m)] \leftarrow G$.
- Meta-predicate hierarchy rule. Because $\mathcal{K} \vdash l_1\colon \psi(A_1, \ldots, A_n) \leftarrow G$ and $\psi \leq \phi$, by induction hypothesis, $\mathcal{I} \models \psi(A_1, \ldots, A_n) \leftarrow G$ where $I(\psi) \subseteq I(\phi)$, and if $\mathcal{I} \models G$, then $\mathcal{I} \models A_1, \ldots, \mathcal{I} \models A_n$ and $([\![A_1]\!]_\alpha \times \cdots \times [\![A_n]\!]_\alpha) \cap I(\phi) \neq \emptyset$. So, we have $\mathcal{I} \models \phi(A_1, \ldots, A_n) \leftarrow G$.
- Fact derivation rule. Because $\mathcal{K} \vdash l_1\colon \psi(A_1, \ldots, A_n) \leftarrow G$, by induction hypothesis, $\mathcal{I} \models \psi(A_1, \ldots, A_n) \leftarrow G$. By Definition 10, $\mathcal{I} \models \psi(A_1, \ldots, A_n)$ if and only if $\mathcal{I} \models A_1, \ldots, \mathcal{I} \models A_n$ and $([\![A_1]\!]_\alpha \times \cdots \times [\![A_n]\!]_\alpha) \cap I(\psi) \neq \emptyset$. Hence, $\mathcal{I} \models A_i \leftarrow G$ for every $1 \leq i \leq n$. ∎

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 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. $U_{\mathcal{F},H} = \mathbb{N} - \{l \in \mathbb{N} \mid \mathit{ground}(\mathcal{K}) \vdash l\colon L \leftarrow G \ \&\ L \in \mathcal{MA}\}$,
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 $I_H(p)(\tau) = \bigcup_{q \leq p}\{l \in U_{\mathcal{F},H} \mid \mathit{ground}(\mathcal{K}) \vdash l\colon q(\tau')\}$ with $\sigma^{-}_{q \to p}(\tau') = \tau$,
   (d) if $\psi \in \Psi_n$, then $I_H(\psi) = \bigcup_{\phi \leq \psi}\{(l_1, \ldots, l_n) \in U^n_{\mathcal{F},H} \mid \text{for every } 1 \leq i \leq n,\ \mathit{ground}(\mathcal{K}) \vdash_{\phi(A_1,\ldots,A_n)} l_i\colon A_i\}$.

A Herbrand model is a model such that ground terms and clauses are interpreted by themselves. For example, the ground term $\mathit{tom}\colon \mathit{person} \in \mathcal{T}_0$ is interpreted by $\mathit{tom}\colon \mathit{person} \in U_H$. For this reason, we define $U_H = \mathcal{T}_0$ in the above definition of Herbrand models. The Herbrand model is extended for interpreting predicate and meta-predicate hierarchies. We define the set $U_{\mathcal{F},H}$ of facts as a set of non-negative integers l introduced in $\mathit{ground}(\mathcal{K}) \vdash l\colon L \leftarrow G$ with $L \in \mathcal{A}$ in order to identify all the derived facts in the interpretation of predicates in a hierarchy.

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

Proof. (1) ($\Rightarrow$) Let $L \leftarrow G$ be a clause with $\mathit{Var}(L \leftarrow G) = \{x_1\colon s_1, \ldots, x_h\colon s_h\}$. Let $\theta = \{x_1\colon s_1/t_1, \ldots, x_h\colon s_h/t_h\}$ be a sorted substitution such that $(L \leftarrow G)\theta \in \mathit{ground}(L \leftarrow G)$. By Definition 12, we have $t_1 \in I_H(s_1), \ldots, t_h \in I_H(s_h)$ 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, $(L \leftarrow G)\theta \in \mathit{ground}(L \leftarrow G)$. By the assumption, $\mathcal{I}_H \models (L \leftarrow G)\theta$. By Definition 12, for all $t_1 \in I_H(s_1), \ldots, t_h \in I_H(s_h)$, $\mathcal{I}\alpha[x_1\colon s_1/t_1, \ldots, x_h\colon s_h/t_h] \models G$ implies $\mathcal{I}\alpha[x_1\colon s_1/t_1, \ldots, x_h\colon s_h/t_h] \models L$. 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 $l_0 \in I_H(p)(\tau)$. Then, by the Herbrand model, there is a sub-predicate r of p ($r \leq p$) such that $\mathit{ground}(\mathcal{K}) \vdash l_0 : r(\tau'')$ with $\sigma^-_{r\to p}(\tau'') = \tau$. Because of $r \leq q$ and $\sigma^-_{p\to q}(\sigma^-_{r\to p}(\tau'')) = \sigma^-_{r\to q}(\tau'')$ in Proposition 1, we have $l_0 \in I_H(q)(\sigma^-_{p\to q}(\tau))$ by Definition 12. This satisfies the condition 3-(e). Let $\psi \leq \phi$ and $(l_1, \ldots, l_n) \in I_H(\psi)$. Then, by the Herbrand model, there is a sub-meta-predicate ψ_0 of ψ ($\psi_0 \leq \psi$) such that $\mathit{ground}(\mathcal{K}) \vdash l_1 : A_1, \ldots, \mathit{ground}(\mathcal{K}) \vdash l_n : A_n$, and $\mathit{ground}(\mathcal{K}) \vdash l'_0 : \psi_0(A_1, \ldots, A_n)$. By $\psi_0 \leq \phi$ and Definition 12, we have $(l_1, \ldots, l_n) \in I_H(\phi)$, 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 $\mathcal{I}_H \models \mathit{ground}(L \leftarrow G)$. Let $L' \leftarrow G' \in \mathit{ground}(L \leftarrow G)$. So, there is a sorted ground substitution θ such that $(L \leftarrow G)\theta = L' \leftarrow G'$. Suppose $\mathcal{I}_H \models \{L_1, \ldots, L_h\}\theta$ where $G = \{L_1, \ldots, L_h\}$. If $L_i\theta$ is of the form $q(t'_1, \ldots, t'_m)$, then there is $l_i \in [\![q(t'_1, \ldots, t'_m)]\!]_\alpha \cap \Delta_H = I(q)([\![t'_1]\!]_\alpha, \ldots, [\![t'_m]\!]_\alpha) \cap \Delta_H (= I_H(q)(t'_1, \ldots, t'_m) \cap \Delta_H)$. So, the condition 3-(c) in Definition 12 implies $\mathit{ground}(\mathcal{K}) \vdash l_i : p(t_1, \ldots, t_n)$ with $\sigma^-_{p\to q}(\langle t_1, \ldots, t_n\rangle) = \langle t'_1, \ldots, t'_m\rangle$ and $p \leq q$. By the sorted substitution rule, we have $\mathcal{K} \vdash l_i : p(t_1, \ldots, t_n)$, and therefore, we obtain $\mathcal{K} \vdash l_i : q(t'_1, \ldots, t'_m)$ by the predicate hierarchy rule. If $L_i\theta$ is of the form $\psi(A_1, \ldots, A_n)$, then there is an n-tuple in $([\![A_1]\!]_\alpha \times \cdots \times [\![A_n]\!]_\alpha) \cap \mathcal{I}_H(\psi)$. So, the condition 3-(d) in Definition 12 implies $\mathit{ground}(\mathcal{K}) \vdash l_i : \phi(A_1, \ldots, A_n)$ because of $\mathit{ground}(\mathcal{K}) \vdash_{\phi(A_1,\ldots,A_n)} l'_j : A_j$ and $\phi \leq \psi$. By the sorted substitution rule, we have $\mathcal{K} \vdash l_i : \phi(A_1, \ldots, A_n)$. Hence, we obtain $\mathcal{K} \vdash l_i : \psi(A_1, \ldots, A_n)$ by the meta-predicate rule. So, $\mathcal{K} \vdash l_1 : L_1\theta, \ldots, \mathcal{K} \vdash l_h : L_h\theta$. Since $L \leftarrow G \in \mathcal{K}$, the sorted substitution rule derives $\mathcal{K} \vdash l : (L \leftarrow \{L_1, \ldots, L_h\})\theta$. By applying the cut rule to them, $\mathcal{K} \vdash l : L\theta$, and therefore, $\mathit{ground}(\mathcal{K}) \vdash l : L\theta$. 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 $\psi'(A'_1, \ldots, A'_k)$, the condition 3-(d) in Definition 12 derives $(l''_1, \ldots, l''_k) \in I_H(\psi'')$ such that $\psi'' \leq \psi'$ and $\mathit{ground}(\mathcal{K}) \vdash l : \psi''(A'_1, \ldots, A'_k)$. By the fact derivation rule, we have $\mathit{ground}(\mathcal{K}) \vdash l''_1 : A'_1, \ldots, \mathit{ground}(\mathcal{K}) \vdash l''_k : A'_k$. By the condition 3-(c) in Definition 12, we have $\mathcal{I}_H \models A'_1, \ldots, \mathcal{I}_H \models A'_k$. Since $(l''_1, \ldots, l''_k) \in ([\![A'_1]\!]_\alpha \times \cdots \times [\![A'_k]\!]_\alpha) \cap I_H(\psi')$, we obtain $\mathcal{I}_H \models \psi'(A'_1, \ldots, A'_k)$. Therefore, $\mathcal{I}_H \models (L \leftarrow G)\theta$. By the first statement in this lemma, we have $\mathcal{I}_H \models L \leftarrow G$. ∎

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 $\mathcal{K} \vdash l : \psi(A_1, \ldots, A_n)$ 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 finite and function-free.

Proof. We show that every inference rule in the Horn-clause calculus cannot be applied infinitely. The set of sorted ground terms in the sorted signature Σ is limited to finite because Σ is finite and function-free. So, all ground clauses in $ground(\mathcal{K})$ can be derived by a finite number of applications of the sorted substitution rule. Moreover, the other inference rules are applied only to ground clauses. For each ground clause, the numbers of applications of these rules are bounded by the numbers of predicate and meta-predicates in their hierarchies in Σ, respectively. The number of applications of the fact derivation rule is bounded by the number of arguments in meta-atoms appearing in the heads of ground clauses. We set CS_0 as the set of ground clauses that are derived from $ground(\mathcal{K})$ using the predicate hierarchy rule, meta-predicate hierarchy rule, and fact derivation rule, and operate $CS_{i+1} = CS_i \cup \{L' \leftarrow G'\}$ for each application of the cut rule deriving $L' \leftarrow G'$. Let CS^* be the set of ground clauses $L \leftarrow G$ such that $L \in \{L' | L' \leftarrow G' \in CS_0\}$ and $G \subseteq \bigcup_{L' \leftarrow G' \in CS_0} G'$. Then, there is a finite sequence $CS_0, CS_1, \ldots, CS_k$ where $CS_0 \subset CS_1 \subset \cdots \subset CS_k$. This is because CS^* is finite and for every CS_i, $CS_i \subseteq CS^*$ holds. ∎

The termination of the calculus indicates 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 EXPTIME-complete (w.r.t. the size 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 that $|\mathcal{K}| = m$, $|\Sigma| = k$, and d is the maximum number of arities of predicates (i.e., bounded arity). Then, $|Var(\mathcal{K})| \leq m$ and $|\mathcal{T}_0^\Sigma| \leq k$. 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 three 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., k^m. (ii) The predicate hierarchy rule, meta-predicate hierarchy rule, and fact derivation rule are applied to the ground clauses derived in step (i). Each ground clause $L \leftarrow G$ in $ground(\mathcal{K})$ can further derive at most $k^{d+1} = k^d \times k$ clauses by applying these rules. This is because the combinations of ground atoms and meta-atoms derived from $L = q(t_1, \ldots, t_d)$ or

$$\phi(q_1(t_1, \ldots, t_d), q_2(t'_1, \ldots, t'_d), \ldots, q_d(t''_1, \ldots, t''_d))$$

are decided by $|\{\psi | \phi \leq \psi\}| \times |\{q'_1 | q_1 \leq q'_1\}| \times \cdots \times |\{q'_d | q_d \leq q'_d\}|$, i.e., $k \times k^d$. This computation is bounded by $k^{m+d+1} = |ground(\mathcal{K})| \times k^{d+1}$. We write $ground^+(\mathcal{K})$

for the set of ground clauses derived in steps (i) and (ii). (iii) The cut rule is applied to the ground clauses in $ground^+(\mathcal{K})$ such that it is applied to $\mathcal{K} \vdash l_1{:}\, L_1, \ldots, \mathcal{K} \vdash l_h{:}\, L_h$ and $\mathcal{K} \vdash l{:}\, L \leftarrow \{L_1, \ldots, L_h\}$. These steps 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 $k^{m+d+1} = |ground^+(\mathcal{K})|$. Therefore, the derivation in (i), (ii), and (iii) is computed in $k^m + k^{m+d+1} + k^{m+d+1}$ steps, (i.e., $2^{m \log k} + 2^{(m+d+1) \log k+1} \leq 2^{(m+d+1) \log k+2}$).

The hardness of the derivation problem is obtained from the program complexity of DATALOG (Dantsin et al., 1997). ∎

5. Query System

We describe a query-answering system for $\mathrm{OSL}_{3\mathrm{h}}$. 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 $\mathcal{A}^{\mathcal{V}} = \{X{:}\, p(t_1, \ldots, t_n) \mid X \in \mathcal{V} \& p(t_1, \ldots, t_n) \in \mathcal{A}\}$. We call the form $X{:}\, p(t_1, \ldots, t_n)$ 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 $\mathcal{MA}^{\mathcal{V}} = \{\psi(A_1^+, \ldots, A_n^+) \mid \psi \in \Psi_n \& A_1^+, \ldots, A_n^+ \in \mathcal{A} \cup \mathcal{A}^{\mathcal{V}}\}$ 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 $X{:}\, q(t'_1, \ldots, t'_m)$ 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 14 (Substitutions for Predicate Variables). A substitution for predicate variables is a partial function $\delta{:}\, \mathcal{A}^{\mathcal{V}} \to \mathcal{A}$ such that $\delta(X{:}\, q(t'_1, \ldots, t'_m)) \in \mathcal{A}_q$ 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 $X{:}\, q(\tau)$. A substitution δ is a most specific substitution for a predicate variable atom $X{:}\, q(\tau)$ if $\delta(X{:}\, q(\tau)) = p(\tau')$ with $\sigma^-_{p \to q}(\tau') = \tau$ and there is no other substitution δ' such that $\delta'(X{:}\, q(\tau)) = r(\tau'')$ with $\sigma^-_{r \to q}(\tau'') = \tau$ 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 $Query{:}\, \mathcal{Q} \to \{yes, no\}$ is defined by the following rule.

1. 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 $Query(Q) = yes$.
2. 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 Queries). Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ. Then, the query system terminates if Σ is function-free.

Proof. Suppose that $|\mathcal{K}| = m$, $|\Sigma| = k$, and d is the maximum number of arities of predicates (i.e., bounded arity). By using the Horn-clause calculus, we can obtain $Con^{atom}(\mathcal{K}) = \{L \mid \mathcal{K} \vdash l{:}\, L\}$ such that $|Con^{atom}(\mathcal{K})| \leq 2^{(m+d+1)\log k+2}$ 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 $\mathcal{K} \vdash l{:}\, Q\delta\theta$ can be generated by the search. The search steps are bounded by $2^{(m+d+1)\log k+2}$. ∎

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 Queries). 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 EXPTIME-complete (w.r.t. the size 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 them 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 $\langle d_1, \ldots, d_n \rangle$ 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.

The naive argument restructuring from predicate p to predicate q deletes some arguments of p and adds some arguments of q as new constants in order to fit a tuple of arguments of p into the argument structure of q.

Example 5. Let us consider the argument declaration $\Lambda = (\{sbj, obj_1, obj_2\}, \{tells\colon \langle sbj, obj_1 \rangle, informs\colon \langle sbj, obj_1, obj_2 \rangle\})$ for a sorted signature Σ that contains the following predicate declarations:

$$tells\colon person \times information$$

and

$$informs\colon person \times information \times person$$

where *tells* is a binary predicate, *informs* is a ternary predicates, and *person* and *information* are sorts. By the naive argument restructuring, we have the following result:

$$\sigma^+_{tells \to informs}(\langle sarry\colon person, i_1\colon information \rangle) = \langle sarry\colon person, i_1\colon information, c \rangle$$

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 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 $\mathcal{K} \vdash_{\sigma^+} l\colon C$) in the sorted Horn-clause calculus is extended by replacing the predicate hierarchy rule with the following rule:

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

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 $\mathcal{K} \vdash_{\sigma^+} l\colon A_2 \leftarrow G$ 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 this 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 $\langle d_1, \ldots, d_n \rangle$ 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

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

where $p\colon \langle a_1, \ldots, a_n \rangle$ and $q\colon \langle a'_1, \ldots, a'_m \rangle$ in Π, $q\colon s_1 \times \cdots \times s_m$ 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.

The label-based argument restructuring from predicate p to predicate q adds the terms t_j that have been derived or new constants indexed by the pairs of labels l and argument names a'_i as arguments of q.

Example 6. Let us consider the argument declaration Λ for Σ in Example 5. By the label-based argument restructuring, we have the following result:

$$\sigma^*_{tells \to informs}(\langle sarry\colon person, i_1\colon information \rangle) = \langle sarry\colon person, i_1\colon information, c_{3,obj_2}\colon person \rangle$$

if $\mathcal{K} \vdash 3\colon tells(sarry\colon person, i_1\colon information) \leftarrow G$.

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 $\sigma^*_{q \to r}(\sigma^*_{p \to q}(\tau)) = \sigma^*_{p \to r}(\tau)$.

Proof. Suppose that $p \leq q \leq r$ with $|\arg(p)| = n$, $|\arg(q)| = m$, and $|\arg(r)| = k$. Let $\tau = \langle d_1, \ldots, d_n \rangle$, $\sigma^*_{p \to q}(\tau) = \langle d'_1, \ldots, d'_m \rangle$, and $\sigma^*_{p \to r}(\tau) = \langle d''_1, \ldots, d''_k \rangle$. We show $\langle d''_1, \ldots, d''_k \rangle = \langle d'''_1, \ldots, d'''_k \rangle$ where $\sigma^*_{q \to r}(\langle d'_1, \ldots, d'_m \rangle) = \langle d'''_1, \ldots, d'''_k \rangle$. Let $d'''_i \in \{d'''_1, \ldots, d'''_k\}$. If $(a_i, d'''_i) \in AL(l\colon p(\tau))$, then $d'''_i = d''_i$ due to $(a_i, d'''_i) \in$

$AL(l\colon q(d'_1,\ldots,d'_m))$. This is because $AL(l\colon p(\tau)) \subseteq AL(l\colon q(d'_1,\ldots,d'_m))$ by Definition 18. If $(a_i, d'''_i) \notin AL(l\colon p(\tau))$, then $d'''_i = c_{l,a_i}\colon s_i$. Also, if $(a_i, d'''_i) \notin AL(l\colon q(d'_1,\ldots,d'_m))$, then $d'_i = c_{l,a_i}\colon s_i$. Otherwise, the same constant $c_{l,a_i}\colon s_i$ must be added in the operation $\sigma^*_{p\to q}(\tau)$, and therefore $d'_i = c_{l,a_i}\colon s_i$. Hence, we have that $\sigma^*_{q\to r}(\langle d'_1,\ldots,d'_m\rangle) = \langle d'''_1,\ldots,d'''_k\rangle$ is identical to $\langle d''_1,\ldots,d''_k\rangle$. ∎

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 $p(c\colon s_2)$ and the rule $p(y\colon s_2) \leftarrow \{q(x\colon s_1, y\colon s_2)\}$ in the sorted signature $\Sigma = (\{\ s,\ \top\ \}, \{\ p,\ q,\ \top\ \}, \emptyset, \Omega, \leq^*)$ such that $\Omega = \{\ c\colon\to s,\ c_1\colon\to s,\ c_2\colon\to s,\ \ldots\ \}$ and $\leq = \{\ p \leq q\ \}$. In the knowledge base, importantly, q has the second argument, whereas p does not have the second argument. First of all, the fact $p(c\colon s_2)$ is used to derive another fact $q(c\colon s_2, c_1\colon s)$ along with the subpredicate relation between p and q. After that, it further deducts the fact $p(c_1\colon s)$. Then, this fact can derive another fact $q(c_1\colon s, c_2\colon s)$ by introducing a new argument c_2. As you see, this reasoning process is cyclic. So, it cannot terminate because new arguments are generated infinitely in this unsafe combination of ontologies and rules. Therefore, the calculus does not terminate. ∎

We will show that the consequence problem for $\mathrm{OSL}_{3\mathrm{h}}$ is undecidable. We use the following undecidability of Horn-clauses in first-order logic.

Theorem 8 (Undecidability of Horn-Clauses with Functions). The satisfiability problem for Horn-clauses with one binary predicate and two unary functions is undecidable (Börger, Grädel and Gurevich, 1997).

We define the transformation from Horn-clauses with functions into sorted Horn-clauses with a predicate-hierarchy as follows. Note that unary functions have to be transformed into predicates and variables in $\mathrm{OSL}_{3\mathrm{h}}$ because function-free sorted signatures are considered.

Definition 19 (Transformation). Let $\mathcal{K}$ be a finite set of Horn-clauses with one binary predicate p and two unary functions f_1, f_2. Then, we define the function-free sorted signature $\Sigma = (S, P, \Psi_n, \Omega, \leq^*)$ such that

$$S = \{\ \top\ \},$$

$$
\begin{aligned}
P &= \{\ p\ \},\\
\Psi_n &= \emptyset,\\
\Omega &= \{\ p{:}\top \times \top,\ \delta_{f_1}{:}\top,\ \delta'_{f_1}{:}\top \times \top,\ \delta_{f_2}{:}\top,\ \delta'_{f_2}{:}\top \times \top\ \},\\
\leq &= \{\ \delta_{f_1} \leq \delta'_{f_1},\ \delta_{f_2} \leq \delta'_{f_2}\ \}
\end{aligned}
$$

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

$$
\begin{aligned}
AN &= \{\ a_1,\ a_2,\ a_3,\ a_4\ \},\\
\Pi &= \{\ p_1{:}\langle a_1, a_2\rangle,\ \delta_{f_1}{:}\langle a_3\rangle,\ \delta'_{f_1}{:}\langle a_3, a_4\rangle,\ \delta_{f_2}{:}\langle a_3\rangle,\ \delta'_{f_2}{:}\langle a_3, a_4\rangle\ \}.
\end{aligned}
$$

Let $\mathcal{K}_0 = \mathcal{K}'$ with $\mathcal{K}' = \{(L \leftarrow G)\{x_1/x_1{:}\top, \ldots, x_n/x_n{:}\top\} \mid L \leftarrow G \in \mathcal{K}\ \&\ Var(L \leftarrow G) = \{x_1, \ldots, x_n\}\}$. By applying the following operation to each Horn-clause $L \leftarrow G$ in $\mathcal{K}_i$ and each function $f \in \{f_1, f_2\}$, $\mathcal{K}_i$ $(i \geq 0)$ is transformed into $\mathcal{K}_{i+1}$ (denoted by $\mathcal{K}_i \rightarrow_{tr} \mathcal{K}_{i+1}$):

– If $L \leftarrow G$ contains $f(x{:}\top)$, then $\mathcal{K}_{i+1} = (\mathcal{K}_i - \{L \leftarrow G\}) \cup \{\delta_f(x{:}\top) \leftarrow, L' \leftarrow G' \cup \{\delta'_f(x{:}\top,\ v{:}\top)\}\}$ where $v{:}\top$ is a new variable and $L' \leftarrow G'$ is obtained by replacing $f(x{:}\top)$ (in $L \leftarrow G$) with $v{:}\top$.

First, every variable x in $\mathcal{K}$ is replaced by the sorted variable $x{:}\top$. In the transformation, the two unary functions f_1, f_2 are replaced by the two unary predicates $\delta_{f_1}, \delta_{f_2}$ with argument structures $\delta_{f_1}{:}\langle a_3\rangle$ and $\delta_{f_2}{:}\langle a_3\rangle$ and two binary predicates $\delta'_{f_1}, \delta'_{f_2}$ with argument structures $\delta'_{f_1}{:}\langle a_3, a_4\rangle$ and $\delta'_{f_2}{:}\langle a_3, a_4\rangle$. Using the predicates, the functional terms $f(t)$ in each clause are transformed into two clauses without functions. For example, the Horn clause

$$p(f_1(x), y) \leftarrow \{p(x, y)\}$$

is transformed into the two clauses

$$
\begin{aligned}
&\delta_{f_1}(x{:}\top) \leftarrow\\
&p(v{:}\top, y{:}\top) \leftarrow \{p(x{:}\top, y{:}\top),\ \delta'_{f_1}(x{:}\top, v{:}\top)\}
\end{aligned}
$$

where $v{:}\top$ is a new variable. Intuitively, the subpredicate relation $\delta_{f_1} \leq \delta'_{f_1}$ leads to a mapping f_1 from $x{:}\top$ into $v{:}\top$.

A knowledge base $\mathcal{K}_m$ in $\mathrm{OSL}_{3\mathrm{h}}$ is obtained from $\mathcal{K}$ when no more transformation can be applied to $\mathcal{K}_m$, i.e., there is a finite sequence $\mathcal{K}_0 \rightarrow_{tr} \mathcal{K}_1 \rightarrow_{tr} \cdots \rightarrow_{tr} \mathcal{K}_{m-1} \rightarrow_{tr} \mathcal{K}_m$. We denote the transformed knowledge base by $trans(\mathcal{K}) = \mathcal{K}_m$. The undecidability for Horn-clauses with one binary predicate and two unary functions implies that there is a finite set of Horn-clauses the satisfiability problem of which is undecidable. So, we show that this set can be transformed into a finite set of extended Horn-clauses in $\mathrm{OSL}_{3\mathrm{h}}$. The existence of $trans(\mathcal{K})$ for each $\mathcal{K}$ is guaranteed by the following lemma.

Lemma 3. For every finite set $\mathcal{K}$ of Horn-clauses with one binary predicate p and two unary functions f_1, f_2, there exists $trans(\mathcal{K})$.

Proof. This lemma is shown by the fact that each transformation deletes function symbols and the number of function symbols in $\mathcal{K}$ is finite. ∎

Lemma 4. Let $\mathcal{K}$ be a finite set of Horn-clauses with one binary predicate p and two unary functions f_1, f_2. Then, $\mathcal{K}$ is satisfiable if and only if $trans(\mathcal{K})$ is Σ-satisfiable.

Proof. ($\Rightarrow$) Let $M_0 = (U_0, I_0)$ be a model and let $\mathcal{I}_0 = (M_0, \alpha_0)$ be an interpretation of $\mathcal{K}$, i.e., $\mathcal{I}_0 \models \mathcal{K}$ in the semantics of first-order logic. From $\mathcal{I}_0$, we construct a Σ-model $M = (U, U_\mathcal{F}, I)$ such that

1. $U = U_0$,
2. $U_\mathcal{F} = \mathbb{N} \cup (\{f_1, f_2\} \times \mathbb{N})$,
3. I is a function with the following conditions:
   (a) $I(\top) = U_0$,
   (b) $I(p)\colon I(\top) \times I(\top) \to 2^{U_\mathcal{F}}$ where for every $(u, u') \in I_0(p)$, $I(p)(u, u') = \{i\}$ with $i \in \mathbb{N}$, and $I(p)(u, u') \neq I(p)(u'', u''')$ if $(u, u') \neq (u'', u''')$,
   (c) if $\delta_f \in \{\delta_{f_1}, \delta_{f_2}\}$, then $I(\delta_f)\colon I(\top) \to 2^{U_\mathcal{F}}$ where for every $u \in U$, $I(\delta_f)(u) = \{(f, i)\}$ with $(f, i) \in \{f\} \times \mathbb{N}$, and $I(\delta_f)(u) \neq I(\delta_f)(u')$ if $u \neq u'$,
   (d) if $\delta'_f \in \{\delta'_{f_1}, \delta'_{f_2}\}$, then $I(\delta'_f)\colon I(\top) \times I(\top) \to 2^{U_\mathcal{F}}$ where for every $u \in U$, $I(\delta'_f)(u, I_0(f)(u)) = I(\delta_f)(u)$,
   (e) for each $f \in \{f_1, f_2\}$, $I(\delta_f)(u) \subseteq I(\delta'_f)(\sigma^+_{\delta_f \to \delta'_f}(u))$ where $\sigma^+_{\delta_f \to \delta'_f}(u) = \langle u, u' \rangle$ if $I_0(f)(u) = u'$.

This Σ-model $M = (U, U_\mathcal{F}, I)$ is used to construct a Σ-interpretation $\mathcal{I} = (M, \Delta, \alpha)$ such that

1. $\Delta \subseteq U_\mathcal{F}$ with the following conditions:
   (a) $i \in \Delta$ if $I(p)(u, u') = \{i\}$,
   (b) for each $\delta_f \in \{\delta_{f_1}, \delta_{f_2}\}$, $(f, i) \in \Delta$ if $I(\delta_f)(u) = (f, i)$,
   (c) for each $\delta'_f \in \{\delta'_{f_1}, \delta'_{f_2}\}$, $(f, i) \in \Delta$ if $I(\delta'_f)(u, u') = (f, i)$,
2. for every variable $x\colon \top$, $\alpha(x\colon \top) = \alpha_0(x)$.

The Σ-interpretation $\mathcal{I}$ satisfies every clause in $trans(\mathcal{K})$. Hence, $trans(\mathcal{K})$ is satisfiable.

($\Leftarrow$) Let $M' = (U', U'_\mathcal{F}, I')$ be a Σ-model and let $\mathcal{I}' = (M', \Delta', \alpha')$ be a Σ-interpretation of $trans(\mathcal{K})$, i.e., $\mathcal{I}' \models trans(\mathcal{K})$ in the semantics of $\mathrm{OSL}_{3\mathrm{h}}$. From $\mathcal{I}'$, we construct a model $M = (U, I)$ such that

1. $U = U'$,
2. I is a function with the following conditions:
   (a) if $f \in \{f_1, f_2\}$ and $u \in U$, then $I(f)\colon U \to U$ where $I(f)(u) = u'$ with $\sigma^+_{\delta_f \to \delta'_f}(u) = \langle u, u' \rangle$,
   (b) $I(p) \subseteq U \times U$ where $(u, u') \in I(p)$ if $I'(p)(u, u') \cap \Delta' \neq \emptyset$.

The interpretation $\mathcal{I}$ satisfies every clause in $\mathcal{K}$. Therefore, $\mathcal{K}$ is satisfiable. ∎

Theorem 9 (Undecidability of Extended Horn-Clauses). Let $\mathcal{K}$ be a knowledge base in a sorted signature Σ and L be an atom or meta-atom. Then, the consequence problem $\mathcal{K} \models L$ is undecidable.

Proof. It can be shown by Lemmas 3 and 4. ∎

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 20 (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 $a' \notin \arg(p)$,
2. $L_q[a', x\colon s] \leftarrow G \in \mathcal{K}$ where $L_p[a, x\colon s] \in G$, 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 21 (Safe Knowledge Bases). A knowledge base $\mathcal{K}$ is safe if

1. $Var(L) \subseteq Var(G)$ for every clause $L \leftarrow G$ in $\mathcal{K}$,
2. $\mathcal{K}$ contains
   (a) 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 $a \in \arg(p)$, and
   (b) for each a in predicate p with $r \leq p$ and $a \notin \arg(r)$, at most one path from a in predicate p to a' in predicate q with $q \leq p'$ and $\arg(p') \not\subseteq \arg(q)$.

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

Example 7. Given the sorted signature $\Sigma = (S, P, \Psi_n, \Omega, \leq^*)$ such that

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

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

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

we can construct the unsafe knowledge base

$$
\begin{aligned}
\mathcal{K} = \{\ & q(c\colon s_1) \leftarrow,\\
& q(y\colon s_2) \leftarrow \{p(x\colon s_1, y\colon s_2)\}\ \}.
\end{aligned}
$$

Lemma 5. 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 $Fun_0(Con(\mathcal{K}))$ be the set of constants occurring in a set of clauses $Con(\mathcal{K})$ and let $|P|$ be the number of predicates. We define $Fun_{0,new}(Con(\mathcal{K})) = \{c \in Fun_0(Con(\mathcal{K})) \cap F_{0,new} \mid c \text{ is generated from new constants more than } |P|\}$. Let $F' = Fun_0(Con(\mathcal{K})) - Fun_{0,new}(Con(\mathcal{K}))$. Then, we have that F' is finite because it is bounded by the condition of $Fun_{0,new}(Con(\mathcal{K}))$.

Suppose that a new constant $c \in F_{0,new}$ (introduced in a label-based argument restructuring) does not belong to F'. The constant c must be indexed

by $c_{l',a}\colon s \in Fun_{0,new}(Con(\mathcal{K}))$. If $c_{l',a}\colon s$ is generated, then there is a clause $l'\colon L_p[a', c'] \leftarrow G$ in $Con^+(\mathcal{K}) = \{l\colon L \leftarrow G \mid \mathcal{K} \vdash l\colon L \leftarrow G\}$ such that $p \leq q$, $a \notin \arg(p)$, and $a \in \arg(q)$. This is because $c_{l',a}\colon s$ must be introduced in the predicate hierarchy rule, i.e., for $p \leq q$, $L_p[a', c']$ derives $L_q[a, c_{l',a}\colon s]$ where $c_{l',a}$ is the new constant. However, every safe knowledge base contains no more than $|P|$-length path to derive $L_q[a, c_{l',a}\colon s]$ from $L_p[a', c']$. 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 10 (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.

Proof. By Lemma 5, it is sufficient to consider only finite sorted signatures if knowledge bases are safe. So, similar to the proof of Theorem 3, Theorem 10 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 $\mathcal{K} \vdash_{\sigma^*} l\colon L\theta$ is EXPTIME-complete (w.r.t. the size 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 that $|\mathcal{K}| = m$, $|\Sigma| = k$, and d is the maximum number of arities of predicates (i.e., bounded arity). Then, Σ is extended into Σ^+ by supplementing new constants in the predicate hierarchy rule. Each ground atom can generate at most $d \cdot k^2 (= d \times k \times k)$ new constants because the number of predicates occurring in a predicate hierarchy and the length of a path are both equal to or less than k. The set of ground atoms $p(t_1, \ldots, t_d)$ is bounded by $|P| \times |F_0|^d \leq k^{d+1}$. So, the total number of new constants is bounded by $d \cdot k^{d+3} = d \times k^2 \times k^{d+1}$. Hence, $|\Sigma^+| \leq |\Sigma| + d \cdot k^{d+3} = k + d \cdot k^{d+3}$. According to $|Var(\mathcal{K})| \leq m$ and $|\mathcal{T}_0^\Sigma| \leq k + d \cdot k^{d+3}$, the derivation is computed in $2^{(m+d+1)\log k+2} + 2^{(m+d+1)(d+3)\log k+(m+d+1)\log d+2}$ steps (similar to the proof of Corollary 1). ∎

Table 1 lists the complexity 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 11 (Termination of Extended Queries). 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 10. ∎

Corollary 4 (Complexity of Extended Queries). 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 EXPTIME-complete (w.r.t. the size of $\mathcal{K}$).

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

Horn-clause calculus	complexity	completeness
argument elimination	EXPTIME	yes
naive argument restructuring	undecidable	no
label-based argument restructuring	undecidable	yes
label-based argument restructuring for safe knowledge bases	EXPTIME	yes

7. Case Study Example

We consider how to reason on ontologies and rules in $\mathrm{OSL}_{3\mathrm{h}}$ for a Semantic Web application. On the Semantic Web, ontologies and rules play an important role in making information machine-readable.

As the most common Web application, Web search engines are useful tools for searching for information on the Web. However, they currently only return Web sites matching some keywords and often not an accurate or comprehensive answer to a query. We provide a case study example of a Semantic Web application that semantically combines a Web search engine service and a reasoning service.

For the reasoning service, the ontology designers construct the following predicate hierarchy:

$$\begin{aligned} &smokes \leq inhales \\ &inhales \leq \top \end{aligned}$$

and the following meta-predicate hierarchy:

$$\begin{aligned} &indirectlyCauses \leq causes \\ &\quad directlyCauses \leq causes \\ &\qquad causes \leq happensBefore \\ &\qquad causes \leq connectedTo \end{aligned}$$

where $smokes$ and $\top$ are unary predicates, $inhales$ is a binary predicate, and $indirectlyCauses$, $causes$, $directlyCauses$, $happensBefore$, and $connectedTo$ are binary meta-predicates.

In addition, the ontology designers can specify the following expressive rule (an extended Horn clause):

$$\begin{aligned} &indirectlyCauses(inhales(x{:}\,person, v{:}\,substance), hasCancer(y{:}\,lung)) \\ &\leftarrow \{\ hasPart(x{:}\,person, y{:}\,lung), \\ &\qquad causes(inhales(x{:}\,person, v{:}\,substance), becomes(y{:}\,lung, z{:}\,color)), \\ &\qquad causes(becomes(y{:}\,lung, z{:}\,color), hasCancer(y{:}\,lung))\ \} \end{aligned}$$

The above rule expresses that the predicate "inhaling a substance indirectly causes lung cancer" can be inferred from the three predicates: "the lungs belong to the inhaling person," "inhaling a substance causes the lungs to become black," and "a lung becoming black causes cancer." This rule enables us to infer indirect causality that indicates semantic links between event descriptions on the Web.

We assume that the following facts are extracted or mined from semantic

tags and texts in some Web pages (as discussed in (Sánchez, Isern and Millan, 2011; Vongdoiwang and Batanov, 2006; Senkul and Salin, 2012)).

$$smokes(jack\colon person) \leftarrow$$
$$hasPart(jack\colon person, c_1\colon lung) \leftarrow$$
$$becomes(c_1\colon lung, black\colon color) \leftarrow$$
$$causes(smokes(jack\colon person), becomes(c_1\colon lung, black\colon color)) \leftarrow$$
$$causes(becomes(c_1\colon lung, black\colon color), hasCancer(c_1\colon lung)) \leftarrow$$

When a user tries to seek an answer to the query "what causes lung cancer to a person?" If the search engine does not know the answer but it can find some facts, then the following query could be formulated to the reasoning service designed with ontologies and rules.

$$Query(happensBefore(X\colon \top(y\colon person), hasCancer(c_1\colon lung)))$$

Using the extended Horn-clause calculus, the reasoning service would return the following conclusion ("Jack inhaling a substance" is a cause of lung cancer in Jack):

$$X = inhales(jack\colon person, c_{a,l}\colon substance)$$

In the reasoning steps, first the atom $smokes(jack\colon person)$ derives the following atom:

$$inhales(jack\colon person, c_{a,l}\colon substance)$$

with argument restructuring for $smokes \leq inhales$. Next, the above rule derives the following causality:

$$indirectlyCauses(inhales(jack\colon person, c_{a,l}\colon substance), hasCancer(c_1\colon lung))$$

with $indirectlyCauses \leq causes$ and $causes \leq happensBefore$. This meta-atom further derives the following upper meta-predicate in the hierarchy:

$$happensBefore(inhales(jack\colon person, c_{a,l}\colon substance), hasCancer(c_1\colon lung))$$

Moreover, the query "what is the specific nature of (or what are the details concerning) Jack inhaling a substance?" i.e., $Query(Y\colon inhales(jack\colon person, c_{a,l}\colon substance))$, would return the following specific answer ("Jack is a smoker"):

$$Y = smokes(jack\colon person)$$

From this result of the reasoning service with ontologies and rules, the search engine can return an accurate answer to the user.

This example of reasoning can be implemented in a safe knowledge base using label-based argument restructuring (listed in Table 1). To implement it in a safe or unsafe knowledge base using argument elimination, the binary predicate *inhales* has to be replaced by a unary predicate. This stops to generate any new constant for the subpredicate relation $smokes \leq inhales$ of unary and binary predicates.

However, the knowledge base becomes unsafe if it contains a cyclic path, or the following rule with unary predicate *absorbed*, binary predicates *mildlyAffects* and *severelyAffects*, and two subpredicate relations between them:

$$absorbed(w_2\colon substance) \leftarrow \{inhales(w_1\colon person, w_2\colon substance)\}$$

$$absorbed \leq mildlyAffects$$
$$absorbed \leq severelyAffects$$

Intuitively, unsafety is caused by the fact that two new constants (i.e., persons in *mildlyAffects* and in *severelyAffects*) are generated from one constant (i.e., substance) in the atom $inhales(jack\colon person, c_{a,l}\colon substance)$.

8. Related Work

In related studies, several approaches for combining ontologies and rules have been proposed such as in (Krisnadhi, Maier and Hitzler, 2011). It is required that reasoning services in the Semantics Web must be decidable. So, decidable languages for ontologies and rules have been designed. In fact, Description Logics and DATALOG are both decidable if they are separated. However, it is not so easy to guarantee decidability if ontologies and rules are combined in reasoning services.

SWRL (Semantic Web Rule Language) (Horrocks et al., 2004) is a combination of OWL and RuleML that leads to undecidable reasoning between ontologies and rules. To overcome this undecidability, several decidable fragments for combining ontologies and rules have been proposed.

DLP (Description Logic Programs) (Grosof et al., 2003) is a subset of both Description Logic and DATALOG, and thus is decidable. OWL 2 RL (Motik et al., 2009) is a recent formalism based on DLP and related to $\mathrm{OSL}_{3\mathrm{h}}$. It can represent both rules and ontologies. Its reasoning is tractable. Compared with DLP and OWL 2 RL, $\mathrm{OSL}_{3\mathrm{h}}$ can describe rules and OWL 2 RL axioms using full expressions in DATALOG, while DLP and OWL 2 RL cannot handle some useful rule expressions because of their lack of full expressions in DATALOG. $\mathrm{OSL}_{3\mathrm{h}}$ could be considered a superset of DLP and OWL 2 RL.

DL-safe rules (Hitzler and Parsia, 2009; Rosati, 2005; Motik et al., 2005) are an approach to integration of rules and Description Logics. But in order to preserve the decidability, the domain of each variable in DL-safe rules is limited to named individuals in DL ABoxes and logic programs. For example, the following DL-rule

$$C(x), R(x, y), p(x), q(y) \rightarrow r(x, y)$$

is safe because the variables x and y of concept C and role R have their domains limited by the variables of predicates p and q in logic programs.

By generalizing the notion of DL-safe rules, extended DL-rules have been proposed. ELP (Krötzsch et al., 2008) is a rule language based on the tractable Description Logic $\mathcal{EL}^{++}$ and is formalized with extended DL-rules for $\mathcal{EL}^{++}$ (called OWL 2 EL). Its rule expressions can represent role inclusion, local reflexivity, role disjointness, and the universal role. $\mathrm{OSL}_{3\mathrm{h}}$ and ELP can express the following rules (presented in (Krötzsch et al., 2008)) which OWL 2 EL cannot handle:

$$NutAllergic(x), NutProduct(y) \rightarrow dislikes(x, y)$$
$$orderedDish(x, y), dislikes(x, y) \rightarrow Unhappy(x)$$
$$dislikes(x, z), Dish(y), contains(y, z) \rightarrow dislikes(x, y)$$

Compared with ELP, $\mathrm{OSL}_{3\mathrm{h}}$ can describe n-ary predicates and meta-predicates

in full rule expressions with sort, predicate, and meta-predicate hierarchies which ELP lacks, e.g., see the case study example in Section 7.

9. Conclusions and Future Work

We developed an order-sorted logic programming language equipped with concept hierarchies of sorts, predicates, and meta-predicates. In $\mathrm{OSL}_{3\mathrm{h}}$, predicates with differently structured arguments are conceptually interpreted in the semantics. According to the semantics, predicate-hierarchy reasoning was realized in the hierarchies of predicates and meta-predicates in which predicate assertions can be used as arguments of meta-level predicates. To achieve such enhanced reasoning, we designed inference rules for predicate and meta-predicate hierarchies in the order-sorted Horn-clause calculus. We employed the calculus to develop a query-answering system for generalized queries containing predicate variables. We showed that the complexity of our expressive query-answering system is identical to that of DATALOG. We proved 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 further plan to extend the formalization of meta-predicates in $\mathrm{OSL}_{3\mathrm{h}}$ in order to represent relationships between objects and events. For example, the assertion $kills(tsunami(c_2 : country), John{:}\,person)$ contains an event and an object in a causal relationship. Because of the limitation to meta-predicates in $\mathrm{OSL}_{3\mathrm{h}}$, this assertion cannot be described in the current formalization. In addition, in order to derive upper and lower meta-predicates flexibly, we plan to restructure arguments of meta-predicates in a meta-predicate hierarchy. The operation of argument restructuring is useful to semantically connect event descriptions in the Semantic Web, and contributes toward the building of the future Linked Data Web (Groza, Grimnes, Handschuh and Decker, 2011). This is an important step in the design of an efficient reasoning system on complex expressions.

Acknowledgements. This research has been partially supported by the Japanese Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (14780311). The authors wish to thank Michaël Thomazo for useful comments and discussions.

References

Aït-Kaci, H. and Nasr, R. (1986), 'LOGIN: A logic programming language with built-in inheritance', *Journal of Logic Programming* **3**(3), 185–215.

Börger, E., Grädel, E. and Gurevich, Y. (1997), *The Classical Decision Problem*, Springer.

Chen, W. and Kifer, M. (1995), Sorted HiLog: Sorts in higher-order logic data languages, *in* 'Proc. of the 5th International Conference on Database Theory (ICDT'95)', LNCS 893, Springer, pp. 252–265.

Cohn, A. G. (1989), 'Taxonomic reasoning with many sorted logics', *Artificial Intelligence Review* **3**, 89–128.

Dantsin, E., Eiter, T., Gottlob, G. and Voronkov, A. (1997), Complexity and expressive power of logic programming, *in* 'IEEE Conference on Computational Complexity', pp. 82–101.

Doets, K. (1994), *From Logic to Logic Programming*, The MIT Press.

Grosof, B., Horrocks, I., Volz, R. and Decker, S. (2003), Description Logic Programs: Combining

Logic Programs with Description Logics, *in* 'Proc. of the Twelfth International World Wide Web Conference (WWW 2003)', Budapest, Hungary.

Groza, T., Grimnes, G. A., Handschuh, S. and Decker, S. (Online First, 28 Dec 2011), 'From raw publications to linked data', *Knowledge and Information Systems* pp. 1–21.

Hanus, M. (1992), Logic programming with type specifications, *in* F. Pfenning, ed., 'Types in Logic Programming', The MIT Press.

Hitzler, P. and Parsia, B. (2009), Ontologies and rules, *in* S. Staab and R. Studer, eds, 'Handbook on Ontologies (2nd Edition)'.

Horrocks, I. and Patel-Schneider, P. F. (2004), A proposal for an owl rules language, *in* 'Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)', ACM, pp. 723–731.
URL: *download/2004/HoPa04a.pdf*

Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B. and Dean, M. (2004), 'SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Recommendation, http://www.w3.org/submission/swrl/'.

Jouannaud, J.-P. and Okada, M. (1991), 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, pp. 455–468.

Kaneiwa, K. (2004), 'Order-sorted logic programming with predicate hierarchy', *Artificial Intelligence* **158**(2), 155–188.

Kaneiwa, K. and Mizoguchi, R. (2005), 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)', LNCS 3702, Springer–Verlag, pp. 169–184.

Kaneiwa, K. and Mizoguchi, R. (2009), 'Distributed reasoning with ontologies and rules in order-sorted logic programming', *Journal of Web Semantics* **7**(3), 252–270.

Kaneiwa, K. and Nguyen, P. (2009), Decidable order-sorted logic programming for ontologies and rules with argument restructuring, *in* 'Proceedings of the 8th International Semantic Web Conference (ISWC 2009)', LNCS 5823, Springer, pp. 328–343.

Kaneiwa, K. and Satoh, K. (2010), 'On the complexities of consistency checking for restricted UML class diagrams', *Theoretical Computer Science* **411**(2), 301–323.

Krisnadhi, A., Maier, F. and Hitzler, P. (2011), OWL and Rules, *in* 'Reasoning Web. Semantic Technologies for the Web of Data - 7th International Summer School 2011, Galway, Ireland, August 23-27, Tutorial Lectures', LNCS 6848, pp. 382–415.

Krötzsch, M., Rudolph, S. and Hitzler, P. (2008), ELP: Tractable rules for OWL 2, *in* 'Proceedings of the 7th International Semantic Web Conference (ISWC 2008)', LNCS 5318, pp. 649–664.

Lloyd, J. W. (1987), *Foundations of Logic Programming*, Springer-Verlag.

Manzano, M. (1993), Introduction to many-sorted logic, *in* 'Many-sorted Logic and its Applications', John Wiley and Sons, pp. 3–86.

Motik, B. (2007), 'On the Properties of Metamodeling in OWL', *Journal of Logic and Computation* **17**(4), 617–637.

Motik, B., Grau, B. C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (2009), 'OWL 2 Web Ontology Language Profiles, W3C Recommendation, http://www.w3.org/tr/owl2-profiles/'.

Motik, B., Sattler, U. and Studer, R. (2005), 'Query Answering for OWL-DL with Rules', *Journal of Web Semantics: Science, Services and Agents on the World Wide Web* **3**(1), 41–60.

Nguyen, P., Kaneiwa, K., Corbett, D. and Nguyen, M.-Q. (2007), An ontology formalization of relation type hierarchy in conceptual structure theory, *in* 'Proceedings of the 21th Australian Joint Conference on Artificial Intelligence (AI2008)', LNCS 5360, Springer–Verlag, pp. 79–85.

Patel-Schneider, P. F., Hayes, P. and Horrocks, I. (2004), 'OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation, http://www.w3.org/tr/2004/rec-owl-semantics-20040210/'.

Rosati, R. (2005), 'On the decidability and complexity of integrating ontologies and rules', *Journal of Web Semantics* **3**(1), 41–60.

Sánchez, S., Isern, D. and Millan, M. (2011), 'Content annotation for the semantic web: an automatic web-based approach', *Knowledge and Information Systems* **27**, 393–418.

Schmidt-Schauss, M. (1989), *Computational Aspects of an Order-Sorted Logic with Term Declarations*, Springer-Verlag.

Senkul, P. and Salin, S. (2012), 'Improving pattern quality in web usage mining by using semantic information', *Knowledge and Information Systems* **30**, 527–541.
Socher-Ambrosius, R. and Johann, P. (1996), *Deduction Systems*, Springer-Verlag.
Vongdoiwang, W. and Batanov, D. (2006), 'An ontology-based procedure for generating object model from text description', *Knowledge and Information Systems* **10**, 93–108.
Woods, W. and Schmolze, J. (1992), 'The KL-ONE family', *Computers and Mathematics with Applications, Special Issue on Semantic Networks in Artificial Intelligence, Part 1* **23**(2–5), 133–178.

Author Biographies

insert photo

Ken Kaneiwa is Associate Professor at Department of Electrical Engineering and Computer Science, Iwate University, Japan. He worked for Fujitsu Ltd. from 1993 to 1996 and received his MS and PhD degrees from Japan Advanced Institute of Science and Technology in 1998 and 2001, respectively, majoring in Information Science in both cases. His research interests include knowledge representation and reasoning, order-sorted logic and semantic web. He received Best Paper Award from Japanese Society for Artificial Intelligence in 2006 and is a member of Japan Society for Software Science and Technology, Institute of Electronics, Information and Communication Engineers, Information Processing Society of Japan, Japanese Society for Artificial Intelligence and Association for Logic Programming.

insert photo

Philip H.P. Nguyen received the MS degree in pure mathematics from the University of Grenoble, France, the postgraduate degree in economic mathematics, the postgraduate degree in industry management, and the PhD degree in information technology, respectively, from the University of Paris, France. His research interests include knowledge representation, ontology, human-computer interface and multi-agent system. He is currently a principal technical specialist for the Attorney-General's Department of the Government of South Australia in Adelaide (Australia).

Correspondence and offprint requests to: Ken Kaneiwa, Department of Electrical Engineering and Computer Science, Iwate University, 4-3-5 Ueda, Morioka, Iwate 020-8551, Japan, Email: kaneiwa@cis.iwate-u.ac.jp