Distributed Reasoning with Ontologies and Rules in Order-Sorted Logic Programming[1]

Ken Kaneiwa
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika, Soraku, Kyoto 619-0289, Japan
kaneiwa@nict.go.jp

Riichiro Mizoguchi
Osaka University
8-1 Mihogaoka, Ibaraki, Osaka 567-0047, Japan
miz@ei.sanken.osaka-u.ac.jp

Abstract

Integrating ontologies and rules on the Semantic Web enables software agents to interoperate between them; however, this leads to two problems. First, reasoning services in SWRL (a combination of OWL and RuleML) are not decidable. Second, no studies have focused on distributed reasoning services for integrating ontologies and rules in multiple knowledge bases. In order to address these problems, we consider distributed reasoning services for ontologies and rules with decidable and effective computation. In this paper, we describe multiple order-sorted logic programming that transfers rigid properties from knowledge bases. Our order-sorted logic contains types (rigid sorts), non-rigid sorts, and unary predicates that distinctly express essential sorts, non-essential sorts, and non-sortal properties. We formalize the order-sorted Horn-clause calculus for such expressions in a single knowledge base. This calculus is extended by embedding rigid-property derivation for multiple knowledge bases, each of which can transfer rigid-property information from other knowledge bases. In order to enable the reasoning to be effective and decidable, we design a query-answering system that combines order-sorted linear resolution and rigid-property resolution as top-down algorithms.

keywords: distributed reasoning, ontologies and rules, order-sorted logic, rigidity, linear resolution system

1 Introduction

Ontologies and rules play an important role in the Semantic Web; they ensure that information available on the World Wide Web is machine-readable. Markup languages such as RDF (Resource Description Framework [33]), OWL (Web Ontology Language [41]), and RuleML (Rule Markup Language [1]) have been developed to model ontologies and rules on the Semantic Web. Description Logics [5] and DATALOG provide formal semantics and decidable reasoning services for OWL and RuleML. In particular, SWRL (Semantic Web Rule

[1]This paper is a significantly extended version of [29].

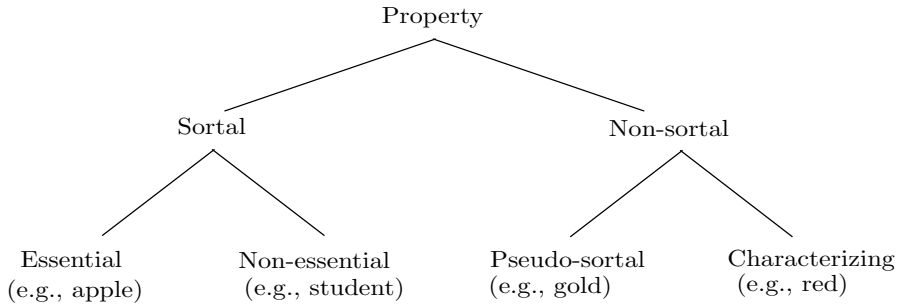


Figure 1: Ontological property classification

Language [26]), a combination of OWL and RuleML has been proposed to enable software agents to interoperate between ontologies and rules. However, because of the expressive power of SWRL, its reasoning services tend to become undecidable, as shown by Horrocks and Patel-Schneider [25].

In the Semantic Web context, knowledge bases consisting of ontologies and rules are widely distributed across multiple sites, contexts, and domains. To address distributed ontologies, C-OWL (Context OWL [8]) is used; C-OWL localizes ontologies and allows explicit mappings between two ontologies. However, to the best of our knowledge, distributed reasoning services for integrating ontologies and rules have not been developed, although software agents have their own facts and rules in knowledge bases. To address these problems, it is necessary to study distributed reasoning services for ontologies and rules to achieve decidable reasoning through effective computation. In other words, decidability must be retained even if reasoning for integrated ontologies and rules is further extended by distributed reasoning services. This is because a combination of different reasoning services often gives rise to undecidability or difficulties similar to those observed when using SWRL.

As an alternative, order-sorted logic [39, 12, 53, 54, 3, 31, 46], which is an extension of first-order predicate logic, involves many sorts and their hierarchy (called sort hierarchy). Order-sorted logic has three advantages: (i) reduction in search space through restriction on the domains and ranges of functions, predicates, and variables [51, 52, 53], (ii) structural knowledge representation by means of partially ordered sorts [13], and (iii) detection of sort errors in well-sorted formulas [40]. General logical languages represent all formulas, even if some of them contain expressions not relevant to ontology and knowledge representation. We believe that well-sorted formulas based on formal ontology ensure that knowledge bases are reliable similar to type-checked programs.

In formal ontology, entities are essentially classified into properties, events, processes, objects, and parts; entity relationships are also formally defined (parthood, dependence, connection, etc.) [7, 45, 44]. Upper ontologies that classify properties on the basis of sortality, rigidity, and identity have been developed by Guarino and Welty [22]. In Figure 1 [20], properties are subdivided into two categories: sortal and non-sortal [48, 37]. Sortal properties are further subdivided into essential and non-essential properties[2]. Essential properties are sortal and rigid. Rigidity implies that if an entity has an essential property, it must have that property in any possible world. For example, if John has the essential property

[2]Guarino and Welty called them *substantial properties* and *non-substantial properties* [21]; we use *essential* instead of *substantial* in order to avoid evoking the property possessed by substances.

Table 1: An overview of our algorithms

	bottom-up algorithm	top-down algorithm
single knowledge base	sorted Horn clause calculus	linear resolution
multiple knowledge bases	rigid-property derivation	rigid-property resolution
decidable strategy for multiple knowledge bases		query-answering of rigid-property resolution

person, he is always a person. Therefore, every individual with the property *person* continues to have the essential property. On the other hand, non-essential properties are sortal and non-rigid. Thus, if an entity has a non-essential property, it may not have the property at certain times or in certain situations. For example, if John has the non-essential property *student*, he may cease to be a student at any time in the future even if he is a student at present; i.e., the non-essential property can be regarded as the role of an individual. Such property classifications [10, 30, 18] take into account the differences between sorts and unary predicates in logic. The rigidity of essential and non-essential properties can be interpreted in terms of possible worlds, i.e., a property is characterized by whether or not it holds in a possible world. In this context, we can model properties for different knowledge bases. If a knowledge base is updated, another property model can exist at that given point of time. Moreover, if two software agents possess different knowledge bases through their interoperation between ontologies and rules, there can be two property models for different beliefs.

In this study, we propose a framework for multiple order-sorted logic programming where a sort hierarchy corresponds to an ontology, and we use property classification (as in formal ontology) to distinguish rigid properties among knowledge bases. We refine the syntax and semantics of order-sorted logic by combining it with the notions of essential sorts, non-essential sorts, and non-sortal properties [21]. We describe the order-sorted Horn-clause calculus and a linear resolution system for such properties in a single knowledge base. For multiple knowledge bases, each knowledge base can extract rigid-property information from other knowledge bases; this phenomenon is called rigid property derivation.

Table 1 lists two reasons for the development of the Horn-clause calculus and the linear resolution system. First, we need to design both bottom-up and top-down algorithms because each has a distinct advantage. The Horn-clause calculus follows the bottom-up algorithm, which is easy to design and implement because it uses intuitive inference rules without unification. However, it is not efficient because the truth of a goal is verified by deriving all the ground formulas for the worst case. Second, we prove the completeness of the linear resolution system on the basis of the completeness of the bottom-up algorithm. The Horn-clause calculus easily generates a canonical model that proves its completeness; that is, all derivable formulas correspond to the model. However, the linear resolution system does not easily generate a canonical model because the resolution steps skip some reasoning redundancies through unification of terms, i.e., all the ground formulas are not necessary as long as a goal is derived. As a result, the linear resolution system is more effective than the Horn-clause calculus.

For the same reasons, a rigid-property resolution system is designed as an effective top-

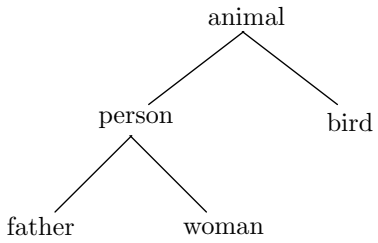


Figure 2: A sort hierarchy

down algorithm; this algorithm enables rigid-property derivation for multiple knowledge bases. To implement the algorithm on a computer, we design a query-answering algorithm that uses a decidable strategy. This algorithm terminates under SLD-resolution (selection-rule driven linear resolution for definite clauses) when the Horn clauses are function-free (although they can be recursive[3]).

This paper is organized as follows. In Section 2, we list the examples that motivated us to present the syntax and semantics of refined order-sorted logic and reasoning services among multiple knowledge bases. In Section 3, we formalize order-sorted logic with types, non-rigid sorts, and sort predicates. In Section 4, we describe the development of the Horn-clause calculus, which we extend using sorted and unsorted substitutions and inference rules for sort predicates. We then present a system for the derivation of rigid properties for multiple knowledge bases. In Section 5, we describe a decidable query-answering system that incorporates the resolution systems of order-sorted reasoning and rigid-property derivation in multiple knowledge bases. In Section 6, we discuss related research and the contributions of our study to the Semantic Web. Finally, we conclude with a short summary and our future prospects in Section 7.

2 Motivating Examples

2.1 Rigidity of Types, Sorts, and Unary Predicates

In a sorted first-order language, each sort hierarchy is constructed as a pair $(S, \leq)$ of a set S of sorts and a subsort relation $\leq$ over S. For example, we can define the subsort declarations $person \leq animal$, $bird \leq animal$, $father \leq person$, and $woman \leq person$, which describe the sort hierarchy in Figure 2. By using sorts in a hierarchy, each variable can be restricted by sort s as a subset of the universe. For example, the domain of the sorted variable $x\colon person$ is the set of persons. In sort theory [6, 17] and constraint logic [9], the sorted constant $c\colon s$ is equivalent to the sort predicate formula $s(c)$, and the subsort relation $s \leq s'$ in a sort hierarchy is represented by its equivalent implication form $\forall x(s(x) \Rightarrow s'(x))$ in first-order logic. This translation motivated us to consider whether sorts and unary predicates are logically and semantically identical.

Table 2 shows that essential sorts, non-essential sorts, and non-sortal properties correspond to types τ [21], non-rigid sorts σ, and unary predicates p, respectively. This distinction between properties changes the meaning of *instantiation* of properties (i.e., term t is

[3]As defined in [35], a predicate p is said to depend on a predicate q if q appears in the body of a clause whose head is p. A set of clauses is recursive if there is a cycle in the dependency relation among predicates.

Table 2: Rigidity of types, sorts, and unary predicates

property	expression		instance_of	subsumption
essential sort	sort s	type τ	$\tau(t)$ rigid	$\tau_1 < \tau_2$
non-essential sort		non-rigid sort σ	$\sigma(t)$ non-rigid	$\sigma_1 < \tau_2$
				$\sigma_1 < \sigma_2$
non-sortal property	unary predicate p		$p(t)$	$\forall x(p_1(x) \Rightarrow p_2(x))$

an instance of a property if $p(t)$ is true where p denotes the property). From the semantic viewpoint, the instantiation of type $\tau(t)$ implies that t forever belongs to τ since all types are rigid. In contrast, the instantiation of a non-rigid sort $\sigma(t)$ and unary predicate $p(t)$ is not always true since non-rigid sorts and unary predicates are not rigid. For example, let *person* be a type, *student* be a non-rigid sort, and *happy* be a unary predicate. Then, $person(john)$ can be true at all times, but the truth of $student(john)$ and $happy(john)$ is variable and depends on the situation. By using our logic, *subsumption* between properties (i.e., a property subsumes another property) can be expressed using either of two forms: *subsort* and *implication*. The subsort relations $\tau_1 < \tau_2$, $\sigma_1 < \tau_2$, and $\sigma_1 < \sigma_2$[4] are declared to be true in all situations, while some of the implication forms $\forall x(p_1(x) \Rightarrow p_2(x))$ hold only in some situations. For example, $student < person$ is true in any situation, but the truth of $\forall x(rich(x) \Rightarrow happy(x))$ depends on the situation. Hence, we should select a subsort relation $s_1 < s_2$ if a sort s_1 always subsumes another sort s_2. In addition, any relation of the form $\tau_1 < \sigma_2$ (i.e., a type is a subsort of a non-rigid sort) is not allowed as a subsumption due to the consideration of rigid and non-rigid properties [22].

To appropriately formalize these notions in logic, we use order-sorted logic with sort predicates [6, 28] as a basic language. The sorted logic contains sort predicates in addition to sorted terms and formulas, which are useful for expressing the unary predicates p_τ and p_σ of types τ and non-rigid sorts σ[5]. In standard order-sorted logic, a variable x of sort s is denoted by $x\colon s$ and a constant c of sort s is denoted by $c\colon s$. Furthermore, we carefully address the rigidity of types, non-rigid sorts, and unary predicates in sorted expressions of the logic. In particular, the sorted expressions $x\colon \tau$, $x\colon \sigma$, and $c\colon \tau$ are allowed, while the sorted expression $c\colon \sigma$ is not. For example, $x_1\colon person$ and $x_2\colon customer$ are variables of the type (rigid sort) *person* and the non-rigid sort *customer*, and $john\colon person$ is a constant of the type *person*. However, constants of non-rigid sorts $c\colon \sigma$, such as $john\colon customer$ and $john\colon US\text{-}citizen$ are meaningless since *customer* and *US-citizen* are not rigid for some humans. In other words, since the elements of non-rigid sorts are not fixed, an element may not belong to a non-rigid sort in a particular situation. Thus, we are required to use such a non-rigid sort as a sort of variables $x\colon \sigma$ and a sort predicate $\sigma(t)$ (or $p_\sigma(t)$). For example, we can describe the formulas as

$$(\forall x\colon customer)(excellent(x\colon customer) \Rightarrow obtaining\_a\_discount(x\colon customer))$$

and $customer(john)$.

[4]Let s_1, s_2 be sorts. $s_1 < s_2$ holds if $s_1 \neq s_2$ and s_1 is a subsort of s_2.

[5]A sort predicate p_s is simply denoted by s when this will not cause confusion.

2.2 Reasoning among Multiple Knowledge Bases

Let us consider multi-agent reasoning [16] where multiple knowledge bases can be assigned to agents. For example, in Figure 3, each agent has its own knowledge base as a set of sorted formulas; all agents also possess and access common taxonomic knowledge as a conceptual hierarchy of sorts and types. In other words, facts and rules in each knowledge base represent assertional knowledge in an agent, context, or application domain; on the other hand, a sort hierarchy expresses taxonomic knowledge commonly used in different situations.

In multi-agent reasoning, each agent may be able to transfer rigid-property information from other agents. As an illustrative example, we consider the following knowledge bases with the sort and type hierarchies shown in Figure 4:

Knowledge base 1:

(1a) $male\_customer(john\colon person)$,

(1b) $excellent(john\colon person)$,

(1c) $(\forall x\colon customer)(excellent(x\colon customer) \Rightarrow obtaining\_a\_discount(x\colon customer))$

Knowledge base 2:

(2a) $pet\_seller(mary\colon person)$,

(2b) $(\forall x\colon customer)(cares\_about(mary\colon person, x\colon customer))$,

(2c) $(\forall x\colon bird)(cares\_about(mary\colon person, x\colon bird))$

Knowledge base 3:

(3a) $canary(peter\colon animal)$,

(3b) $(\forall x\colon animal)(bird(x\colon animal) \Rightarrow canfly(x\colon animal))$

Knowledge base 4:

(4a) $father(tony\colon animal, peter\colon animal)$,

(4b) $(\forall y\colon animal)(\forall x\colon animal)(father(y\colon animal, x\colon animal) \wedge bird(x\colon animal) \Rightarrow bird(y\colon animal))$

Fact (3a) is true in knowledge base 3, and if the subsumption $canary < bird$ holds in the sort and type hierarchies, then $bird(peter\colon animal)$ can be derived in knowledge base 3. This is also true in any other knowledge base since the evaluation of the type *bird* does not depend on a particular situation, i.e., it is rigid. In other words, the elements of the type *bird* can be propagated (Peter is forever a bird), unlike the elements of the non-rigid sort *customer*. Hence, since instantiation as rigid-property information must be true in all situations, we conclude that the fact $bird(peter\colon animal)$ is true in knowledge bases 1, 2, and 4. On the basis of this additional information, knowledge base 2 derives the new fact $cares\_about(mary\colon person, peter\colon animal)$ from (2c), which cannot be derived without importing the rigid-property information. Moreover, from the fact $father(tony\colon animal, peter\colon animal)$ in knowledge base 4, we cannot normally derive whether Tony is a bird. By transferring the fact $bird(peter\colon animal)$, we can use knowledge base 4 to derive $bird(tony\colon animal)$ (Tony is a bird) from fact (4a) and rule (4b).[6] Therefore,

[6] In order to guarantee safe reasoning, we assume that rules whose conclusion is a rigid assertion only contain rigid assertions in their conditions. This holds for unary and n-ary predicates.

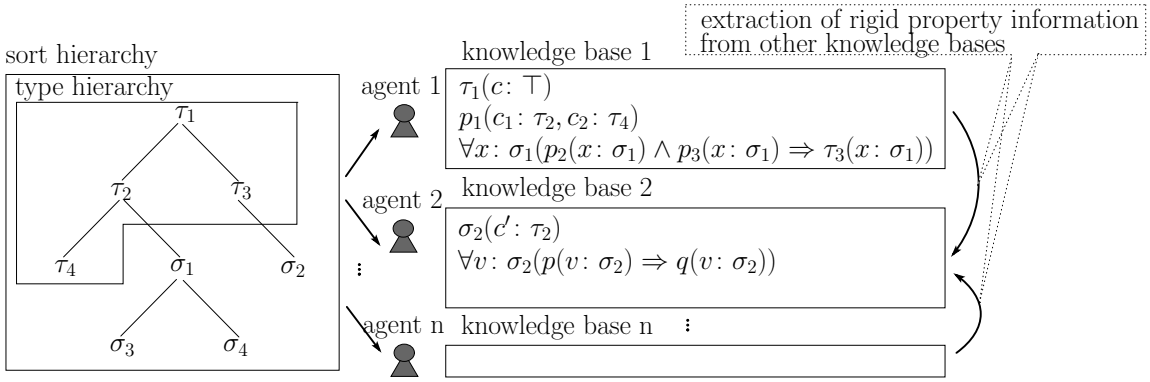


Figure 3: Multiple knowledge bases with sort and type hierarchies

the rigid fact *bird*(*tony*: *animal*) recursively results in *canfly*(*tony*: *animal*) in knowledge base 3 and *cares_about*(*mary*: *person*, *tony*: *animal*) in knowledge base 2[7]. However, when fact (1a) and the subsumption *male_customer* $<$ *customer* hold, *customer*(*john*: *person*) is true only in knowledge base 1 because *customer* is not rigid. Hence, (2b) does not lead to *cares_about*(*mary*: *person*, *john*: *customer*) in knowledge base 3.

3 Order-Sorted Logic with Rigidity and Sort Predicates

First, we introduce three classified property expressions: types, non-rigid sorts, and unary predicates in the syntax and semantics of order-sorted logic with sort predicates based on [47, 43, 38].

3.1 Syntax

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

1. *T: a countable set of type symbols $\tau_1, \tau_2, \ldots$ including the greatest type $\top$*
2. *N: a countable set of non-rigid sort symbols $\sigma_1, \sigma_2, \ldots$ with $T \cap N = \emptyset$*
3. *C: a countable set of constant symbols*
4. *F_n: a countable set of n-ary function symbols*
5. *P_n: a countable set of n-ary predicate symbols*
6. *$\rightarrow, \leftarrow, (,)$: the connective and auxiliary symbols*
7. *V_s: an infinite set of variables $x\colon s, y\colon s, z\colon s, \ldots$ of sort s.*

[7]First, knowledge base 3 exports rigid-property information to knowledge base 4. A new fact can then be derived in knowledge base 4; knowledge base 3 then imports the new fact as rigid-property information from knowledge base 4.

We generally call type symbols (denoted $\tau, \tau_1, \tau_2, \ldots$) or non-rigid sort symbols (denoted $\sigma, \sigma_1, \sigma_2, \ldots$) *sort symbols* (denoted $s, s_1, s_2, \ldots$). Namely, $T \cup N$ is the set of sort symbols. The set of variables of all sorts is denoted by $V = \bigcup_{s \in T \cup N} V_s$. For all sorts $s \in T \cup N$, the unary predicates $p_s \in P_1$ indexed by the sorts s (called *sort predicates*) are introduced, and the set of sort predicates is denoted by $P_{T \cup N} = \{p_s \mid s \in T \cup N\}$. In particular, the predicate p_τ indexed by a type τ is called a *type predicate*, and the predicate p_σ indexed by a non-rigid sort σ is called an *non-rigid sort predicate*. In what follows, we assume that the language $\mathcal{L}$ contains all the sort predicates in $P_{T \cup N}$.

Definition 2 (Sorted Signatures with Rigidity) *A signature of a sorted first-order language $\mathcal{L}$ with rigidity and sort predicates (called sorted signature) is a tuple $\Sigma = (T, N, \Omega, \leq)$ such that:*

1. *$(T \cup N, \leq)$ is a partially ordered set of sorts (called a sort hierarchy) where*
   (a) *$T \cup N$ is the union of the set of type symbols and the set of non-rigid sort symbols in $\mathcal{L}$, and*
   (b) *each ordered pair is not of the form $\tau \leq \sigma$ (i.e., it is of the form $\sigma_i \leq \sigma_j$, $\sigma \leq \tau$, or $\tau_k \leq \tau_l$);*
2. *Ω is a finite set of constant, function, and predicate declarations such that*
   (a) *if $c \in C$, then there is a unique constant declaration of the form $c\colon \ \rightarrow \tau \in \Omega$ (which means $c\colon \emptyset \rightarrow \tau \in \Omega$);*
   (b) *if $f \in F_n$, then there is a unique function declaration of the form $f\colon \tau_1 \times \cdots \times \tau_n \rightarrow \tau \in \Omega$;*
   (c) *if $p \in P_n$, then there is a unique predicate declaration of the form $p\colon s_1 \times \cdots \times s_n \in \Omega$ (in particular, for all sort predicates $p_s \in P_{T \cup N}$, $p_s\colon \top \in \Omega$).*

The declarations in Ω determine the domains and ranges of constants, functions, and predicates. The domains and ranges of constants and functions are required to be rigidly sorted. Specifically, constant and function declarations are defined by types τ_i in order to avoid the non-rigid domains and ranges of constants and functions. In contrast, predicate declarations are defined by any sorts s_i (types or non-rigid sorts) since the domains of predicates can be non-rigid. In the following example, we will show constant and function declarations that are defined by types and non-rigid sorts.

Example 1 (Rigidity of Constants and Functions) *Consider a sorted signature that contains the subsort relation student $\leq$ person, the constant and function declarations:*

$$\begin{aligned} &\mathit{john}\colon \ \rightarrow \mathit{person} \in \Omega \\ &\mathit{father}\colon \mathit{person} \rightarrow \mathit{person} \in \Omega \end{aligned}$$

for $\mathit{john} \in C$, $\mathit{father} \in F_1$, and person $\in T$, and the predicate declaration:

$$\mathit{getting\_a\_scholarship}\colon \mathit{student} \in \Omega$$

for $getting\_a\_scholarship \in P_1$ *and* $student \in N$*. The sorted signatures exclude the following declarations of unrigidly sorted constants and functions as sort errors:*

$$john\colon \rightarrow student \in \Omega$$
$$father\colon person \rightarrow teacher \in \Omega$$

for $student, teacher \in N$ *give rise to the sorted terms:*

$$john\colon student$$
$$father(john\colon student)\colon teacher.$$[8]

The first expresses the constant john of the non-rigid sort student, and the second represents the function father with the argument $john\colon student$ *the range of which is the non-rigid sort teacher. These expressions contain ill-sorted errors because the sorts student and teacher are not rigid for some humans.*

The restriction of a sort hierarchy to types is a partially ordered set of types and called a type hierarchy. The type hierarchy in a sorted signature corresponds to the backbone taxonomy consisting only of rigid properties [55]. Based on the definition of subsumption relation in [22], any relation of the form $\tau \leq \sigma$ is not allowed in the sort hierarchy, since although types τ are rigid, non-rigid sorts σ are not rigid. It should be noted that Kaplan [30] pointed out that Guarino and Welty's formulation of subsumption was incorrect or inconsistent. More precisely, he proved that 'it is possible for a non-rigid property to subsume a rigid one, given Guarino and Welty's definitions of rigid, non-rigid, and subsumption.' If 'p subsumes q' is formalized by $\forall x(q(x) \Rightarrow p(x))$, this undesired analysis is derived. However, as mentioned by Kaplan, if the subsumption is formalized by $\Box(\forall x)(q(x) \Rightarrow p(x))$, the condition that an non-rigid property cannot subsume a rigid one *necessarily* is correct. Therefore, we support the necessary subsumption in the semantics that will be defined latter (in Definition 9).

Next, we define terms, atoms (atomic formulas), goals, and clauses of a sorted first-order language with rigidity and sort predicates.

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

1. *If* $x\colon s' \in V_{s'}$ *and* $s' \leq s$*, then* $x\colon s \in \mathcal{T}_s$*.*
2. *If* $c \in C$*,* $c\colon \rightarrow \tau \in \Omega$*, and* $\tau \leq s$*, then* $c\colon \tau \in \mathcal{T}_s$*.*
3. *If* $t_1 \in \mathcal{T}_{\tau_1}, \ldots, t_n \in \mathcal{T}_{\tau_n}$*,* $f \in F_n$*,* $f\colon \tau_1 \times \cdots \times \tau_n \rightarrow \tau \in \Omega$*, and* $\tau \leq s$*, then* $f(t_1, \ldots, t_n)\colon \tau \in \mathcal{T}_s$*.*

Note that $\mathcal{T}_s$ contains not only terms of sort s but also terms of subsorts s' of sort s if $s' \leq s$. For example, let $person, animal$ be types and let $person \leq animal \in \Omega$. Then, $john\colon person$ belongs to the set $\mathcal{T}_{animal}$ of terms of the sort *animal* and its subsorts. The set of terms of all sorts is denoted by $\mathcal{T} = \bigcup_{s \in T \cup N} \mathcal{T}_s$. The function *sort* is a mapping from sorted terms to their sorts, defined by (i) $sort(x\colon s) = s$, (ii) $sort(c\colon \tau) = \tau$, and (iii) $sort(f(t_1, \ldots, t_n)\colon \tau) = \tau$.

[8] Note that if $john\colon \rightarrow student$ is declared in Ω, $john\colon student$ is a term of the sort *person* because *student* is a subsort of *person* (i.e., $student \leq person$).

Definition 4 *The function* $Var\colon \mathcal{T} \to 2^V$ *is defined by the following:*

1. $Var(x\colon s) = \{x\colon s\}$,
2. $Var(c\colon \tau) = \emptyset$ *for* $c \in C$,
3. $Var(f(t_1, \ldots, t_n)\colon \tau) = Var(t_1) \cup \cdots \cup Var(t_n)$ *for* $f \in F_n$.

A sorted term is called *ground* if its set of variables is empty. $\mathcal{T}_0 = \{t \in \mathcal{T} \mid Var(t) = \emptyset\}$ is the set of sorted ground terms. The set of ground terms of sort s is denoted by $\mathcal{T}_{0,s} = \mathcal{T}_0 \cap \mathcal{T}_s$. In order to provide a rule language, we restrict the set of possible formulas to Horn-clauses [36, 14] but it can be extended by full formulas.

Definition 5 (Sorted Formulas) *Let* $\Sigma = (T, N, \Omega, \leq)$ *be a sorted signature. The set* $\mathcal{A}$ *of atoms, the set* $\mathcal{G}$ *of goals, and the set* $\mathcal{C}$ *of clauses are defined by the following:*

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* $L_1, \ldots, L_n \in \mathcal{A}$ $(n \geq 0)$, *then* $\{L_1, \ldots, L_n\} \in \mathcal{G}$.
3. *If* $G \in \mathcal{G}$ *and* $L \in \mathcal{A}$, *then* $L \leftarrow G \in \mathcal{C}$.

An atom $p_s(t)$ with a sort predicate is simply denoted by $s(t)$ when this will not cause confusion. A clause $L \leftarrow G$ is denoted by $L \leftarrow$ if $G = \emptyset$. The function Var is extended to atoms, goals, and clauses, i.e., for each $e \in \mathcal{A} \cup \mathcal{G} \cup \mathcal{C}$, $Var(e)$ denotes the set of variables occurring in e.

Example 2 *Let us consider the sorted signature* $\Sigma = (T, N, \Omega, \leq^*)$ *such that*

$$
\begin{array}{rcl}
T & = & \{\ male,\ person,\ canary,\ bird,\ animal,\ \top\ \}, \\
N & = & \{\ pet\_seller,\ customer,\ male\_customer\ \}, \\
\Omega & = & \{\ john\colon \to person,\ mary\colon \to person, \\
 & & \quad peter\colon \to animal,\ tony\colon \to animal, \\
 & & \quad excellent\colon person,\ cares\_about\colon animal \times \top, \\
 & & \quad obtaining\_a\_discount\colon customer, \\
 & & \quad father\colon \top \times \top,\ canfly\colon \top\ \} \cup \\
 & & \{\ p_s\colon \top \mid s \in T \cup N\ \}, \\
< & = & \{\ (canary, bird), (bird, animal), (animal, \top), \\
 & & \quad (person, animal), (male, animal), (male\_customer, male), \\
 & & \quad (pet\_seller, person), (customer, person), \\
 & & \quad (male\_customer, customer)\ \},
\end{array}
$$

and $\leq^*$ *is the reflexive and transitive closure of* $<$. *This sorted signature declares the sort and type hierarchies in Figure 4. The expressions:*

$$
\begin{array}{l}
customer(john\colon person) \leftarrow, \\
canfly(x\colon animal) \leftarrow \{bird(x\colon animal)\}
\end{array}
$$

are clauses in the sorted first-order language.

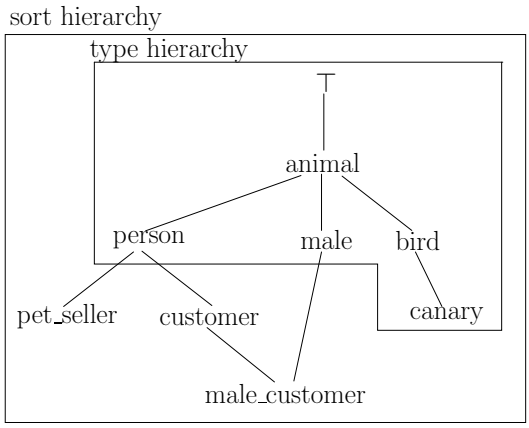


Figure 4: Sort and type hierarchies in a sorted signature

We define a sorted substitution such that each sorted variable $x\colon s$ is replaced with a well-sorted term in $\mathcal{T}_s$.

Definition 6 (Sorted Substitution) *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 $Dom(\theta)$ is finite.*

Moreover, an unsorted substitution is defined as a substitution that operationally ignores the sort of each variable, which may lead to ill-sorted terms.

Definition 7 (Unsorted Substitution) *An unsorted substitution is a partial function $\theta^u\colon V \to \mathcal{T}$ such that if $\theta^u(x\colon s)$ is defined then it belongs to $\mathcal{T}_s - \{x\colon s\}$, and $Dom(\theta)$ is finite.*

For example, for $customer \leq person$, $\theta^u(x\colon customer) = john\colon person$ is an unsorted substitution, but not a sorted substitution. Each of the sorted and unsorted substitutions can be represented by $\{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$. Let θ be a sorted substitution. 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. Similarly, these notions are defined for unsorted substitutions θ^u.

Definition 8 *Let γ be a sorted substitution θ or an unsorted substitution θ^u. The term $t\gamma$ is defined by the following:*

1. $x\colon s\gamma = \gamma(x\colon s)$ *if* $x\colon s \in Dom(\gamma)$,
2. $x\colon s\gamma = x\colon s$ *if* $x\colon s \notin Dom(\gamma)$,
3. $c\colon \tau\gamma = c\colon \tau$,
4. $(f(t_1, \ldots, t_n)\colon \tau)\gamma = f(t_1\gamma, \ldots, t_n\gamma)\colon \tau$,

Note that the substitution γ in statements 2 and 3 does not anything for the term t because the sorted variable $x\colon s$ does not belong to the domain $Dom(\gamma)$ or the term t is a sorted constant. Moreover, the substitution γ in statement 4 is inductively defined for the sorted

variables occurring in the functional term $f(t_1, \ldots, t_n)\colon \tau$. For instance, let γ be a sorted substitution with $\gamma(x\colon s) = c_2\colon s_1$ and $s_1 \leq s$. The functional term $(f(x\colon s, c_2\colon s_2)\colon \tau_1)\gamma$ can be substituted with $f(c_1\colon s_1, c_2\colon s_2)\colon \tau_1$ if $x\colon s \in Dom(\gamma)$ (but there is no condition for τ_1). In the usual manner of first-order logic, sorted and unsorted substitutions are extended to atoms, goals, and clauses. Let E be an expression, θ be a sorted substitution, and θ^u be an unsorted substitution. The application of a sorted (unsorted) substitution to E is denoted by $E\theta$ ($E\theta^u$).

Let t be a sorted term. The substituted term $t\theta^u$ is an ill-sorted term if $t\theta^u \notin \mathcal{T}$. If t' is an ill-sorted term and $t'\theta^u \notin \mathcal{T}$, then $t'\theta^u$ is also an ill-sorted term. The ill-sorted expressions are more generally defined as follows. Let $E \in \mathcal{T} \cup \mathcal{A} \cup \mathcal{G} \cup \mathcal{C}$. The substituted expression $E\theta^u$ is an ill-sorted expression if $E\theta^u \notin \mathcal{T} \cup \mathcal{A} \cup \mathcal{G} \cup \mathcal{C}$. If E' is an ill-sorted expression and $E'\theta^u \notin \mathcal{T} \cup \mathcal{A} \cup \mathcal{G} \cup \mathcal{C}$, then $E'\theta^u$ is an ill-sorted expression. The set of sorted and ill-sorted terms, the set of sorted and ill-sorted atoms, the set of sorted and ill-sorted goals, and the set of sorted and ill-sorted clauses are denoted by $\mathcal{T}^+$, $\mathcal{A}^+$, $\mathcal{G}^+$, and $\mathcal{C}^+$, respectively.

Let E be a sorted or ill-sorted expression. The substitution θ (resp. θ^u) is a sorted ground substitution (resp. unsorted ground substitution) for E if $E\theta$ (resp. $E\theta^u$) is ground and $dom(\theta) = Var(E)$ (resp. $dom(\theta^u) = Var(E)$). The composition $\theta_1\theta_2$ of sorted substitutions θ_1 and θ_2 (resp. unsorted substitutions θ_1^u and θ_2^u) is defined by $(x\colon s)\theta_1\theta_2 = ((x\colon s)\theta_1)\theta_2$ (resp. $(x\colon s)\theta_1^u\theta_2^u = ((x\colon s)\theta_1^u)\theta_2^u$).

3.2 Semantics

The semantics of order-sorted logic with rigidity and sort predicates is defined over possible worlds. This characterizes the rigidity of types and non-rigid sorts by interpreting them in each world.

Definition 9 (Σ-Model) *Let Σ be a sorted signature. A Σ-model M is a tuple (W, U, I) such that*

1. *W is a non-empty set of worlds,*
2. *U is a non-empty set of individuals,*
3. *$I = \{I_w \mid w \in W\}$ is the set of functions I_w for all worlds $w \in W$ with the following conditions:*
   - *(a) if $s \in T \cup N$, then $I_w(s) \subseteq U$ (in particular, $I_w(\top) = U$),*
   - *(b) if $s_i \leq s_j$ for $s_i, s_j \in T \cup N$, then $I_w(s_i) \subseteq I_w(s_j)$,*
   - *(c) if $c \in C$ and $c\colon \to \tau \in \Omega$, then $I_w(c) \in I_w(\tau)$,*
   - *(d) if $f \in F_n$ and $f\colon \tau_1 \times \cdots \times \tau_n \to \tau \in \Omega$, then $I_w(f)\colon I_w(\tau_1) \times \cdots \times I_w(\tau_n) \to I_w(\tau)$,*
   - *(e) if $p \in P_n$ and $p\colon s_1 \times \cdots \times s_n \in \Omega$, then $I_w(p) \subseteq I_w(s_1) \times \cdots \times I_w(s_n)$.*

In the semantics, statement 3-(b) (i.e., $I_w(s_i) \subseteq I_w(s_j)$ holds for all worlds $w \in W$) corresponds to the necessary subsumption $\Box(\forall x)(q(x) \Rightarrow p(x))$, whose validity was proven by Kaplan [30].

Moreover, by restricting Σ-models, we give the class of rigid Σ-models as follows.

Definition 10 (Rigid Σ-Model) *Let Σ be a sorted signature. A rigid Σ-model is a Σ-model $M = (W, U, I)$ such that for all $w_i, w_j \in W$, $\tau \in T$, $c \in C$, and $f \in F$, $I_{w_i}(\tau) = I_{w_j}(\tau)$, $I_{w_i}(c) = I_{w_j}(c)$, and $I_{w_i}(f) = I_{w_j}(f)$.*

In what follows, we will deal with rigid Σ-models $M = (W, U, I)$ that satisfy the following condition:

- If $s \in T \cup N$, then $I_w(s) = I_w(p_s)$.

This condition drives that if $s_i \leq s_j$, then $I_w(p_{s_i}) \subseteq I_w(p_{s_j})$. In the Σ-models, the interpretation of sorts s is equivalent to the interpretation of the sort predicates p_s.

Definition 11 (Variable Assignment) *Let Σ be a sorted signature. A variable assignment on a rigid Σ-model $M = (W, U, I)$ is a function $\alpha_w \colon V \to U$ where $\alpha_w(x \colon s) \in I_w(s)$.*

We define $\alpha_w[x \colon s/d]$ by $(\alpha_w - \{(x \colon s, \alpha_w(x \colon s))\}) \cup \{(x \colon s, d)\}$. In other words, if $v = x \colon s$, then $\alpha_w[x \colon s/d](v) = d$, and otherwise $\alpha_w[x \colon s/d](v) = \alpha_w(v)$. A rigid Σ-interpretation $\mathcal{I}$ is a pair (M, α) of a rigid Σ-model M and a set of variable assignments $\alpha = \{\alpha_w \mid w \in W\}$ on M. Let $\mathcal{I} = (M, \alpha)$. The rigid Σ-interpretation $(M, \alpha - \{\alpha_w\} \cup \{\alpha_w[x \colon s/d]\})$ is denoted by $\mathcal{I}\alpha_w[x \colon s/d]$.

We define an interpretation of sorted and ill-sorted terms (in $\mathcal{T}^+$) as follows.

Definition 12 *Let $\mathcal{I} = (M, \alpha)$ be a rigid Σ-interpretation. The denotation function $[\![\ ]\!]_{w,\alpha} \colon \mathcal{T}^+ \to U$ is defined by the following:*

1. $[\![x \colon s]\!]_{w,\alpha} = \alpha_w(x \colon s)$,
2. $[\![c \colon \tau]\!]_{w,\alpha} = I_w(c)$,
3. $$[\![f(t_1, \ldots, t_n) \colon \tau]\!]_{w,\alpha} = \begin{cases} I_w(f)([\![t_1]\!]_{w,\alpha}, \ldots, [\![t_n]\!]_{w,\alpha}) & \text{if } [\![t_1]\!]_{w,\alpha} \in I_w(\tau_1), \ldots, [\![t_n]\!]_{w,\alpha} \\ & \in I_w(\tau_n) \text{ and } f \colon \tau_1 \times \cdots \times \tau_n \\ & \to \tau \in \Omega \\ \bot & \text{otherwise.} \end{cases}$$

The satisfiability of sorted and ill-sorted atoms, goals, and clauses are defined for a rigid Σ-interpretation $\mathcal{I}$ and a world $w \in W$.

Definition 13 (Rigid Σ-Satisfaction Relation) *Let $\mathcal{I} = (M, \alpha)$ with $M = (W, U, I)$ be a rigid Σ-interpretation, let $F \in \mathcal{A}^+ \cup \mathcal{G}^+ \cup \mathcal{C}^+$, and let $w \in W$. The rigid Σ-satisfaction relation $\mathcal{I}, w \models F$ is defined inductively as follows:*

1. $\mathcal{I}, w \models p(t_1, \ldots, t_n)$ *iff* $([\![t_1]\!]_{w,\alpha}, \ldots, [\![t_n]\!]_{w,\alpha}) \in I_w(p)$.
2. $\mathcal{I}, w \models \{L_1, \ldots, L_n\}$ *iff* $\mathcal{I}, w \models L_1, \ldots, \mathcal{I}, w \models L_n$.
3. $\mathcal{I}, w \models L \leftarrow G$ *iff for all* $d_1 \in I_w(s_1), \ldots, d_n \in I_w(s_n)$, $\mathcal{I}\alpha_w[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n]$, $w \models G$ *implies* $\mathcal{I}\alpha_w[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n], w \models L$ *where* $Var\ (L \leftarrow G) = \{x_1 \colon s_1, \ldots, x_n \colon s_n\}$.

Let Γ be a set of formulas in $\mathcal{A}^+ \cup \mathcal{G}^+ \cup \mathcal{C}^+$. We write $\mathcal{I}, w \models \Gamma$ if, for every formula $F \in \Gamma$, $\mathcal{I}, w \models F$. A formula F is said to be rigid Σ-satisfiable if for some rigid Σ-interpretation $\mathcal{I}$ and world w, $\mathcal{I}, w \models F$. Otherwise, it is rigid Σ-unsatisfiable. F is a consequence of Γ in the class of rigid Σ-interpretations (denoted $\Gamma \models F$) if for each rigid Σ-interpretation $\mathcal{I}$ and $w \in W$, $\mathcal{I}, w \models \Gamma$ implies $\mathcal{I}, w \models F$.

Lemma 1 *Let $\mathcal{I} = (M, \alpha)$ with $M = (W, U, I)$ be a rigid Σ-interpretation, $L \leftarrow G$ be a clause in $\mathcal{C}^+$, θ be a sorted substitution, and let $w \in W$. If $\mathcal{I}, w \models L \leftarrow G$, then $\mathcal{I}, w \models (L \leftarrow G)\theta$.*

Proof. Let $Var(L \leftarrow G) = \{x_1 \colon s_1, \ldots, x_n \colon s_n\}$. By Definition 13, for all $d_1 \in I_w(s_1), \ldots, d_n \in I_w(s_n)$, $\mathcal{I}\alpha_w[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n], w \models G$ implies $\mathcal{I}\alpha_w[x_1 \colon s_1/d_1, \ldots, x_n \colon s_n/d_n]$, $w \models L$. Let $d'_1 \in I_w(s'_1), \ldots, d'_m \in I_w(s'_m)$ where $Var((L \leftarrow G)\theta) = \{y_1 \colon s'_1, \ldots, y_m \colon s'_m\}$. Then, we have $[\![\theta(x_1 \colon s_1)]\!]_{w,\alpha'} \in I_w(s_1)$, ..., $[\![\theta(x_n \colon s_n)]\!]_{w,\alpha'} \in I_w(s_n)$ for $\alpha' = \alpha - \{\alpha_w\} \cup \{\alpha_w[y_1 \colon s'_1/d'_1, \ldots, y_m \colon s'_m/d'_m]\}$ because $\theta(x_1 \colon s_1)$, ..., $\theta(x_n \colon s_n)$ are well-sorted terms of sorts $s_1, \ldots, s_n$. This derives $\mathcal{I}, w \models (L \leftarrow G)\theta$. ∎

Example 3 *Given the sorted signature $\Sigma = (T, N, \Omega, \leq^*)$ of Example 2, we consider a rigid Σ-model $M = (W, U, I)$ such that*

$$\begin{aligned} W &= \{\ w_1, w_2, w_3, w_4\ \}, \\ U &= \{\ j, m, t, p\ \}, \\ I &= \{\ I_{w_1}, I_{w_2}, I_{w_3}, I_{w_4}\ \} \end{aligned}$$

where for $i \leq i \leq 4$, $I_{w_i}(john) = j$, $I_{w_i}(mary) = m$, $I_{w_i}(tony) = t$, $I_{w_i}(peter) = j$, $I_{w_i}(person) = \{j, m\}$, $I_{w_i}(animal) = \{j, m, t, p\}$, for each $s_j \leq s_k$, $I_{w_i}(s_j) \subseteq I_{w_i}(s_k)$, and $I_{w_1}(male\_customer) = I_{w_1}(excellent) = I_{w_1}(customer) = I_{w_1}(obtaining\_a\_discount) = \{j\}$.

Then, we have the following rigid Σ-satisfaction relation.

$$\begin{aligned} \mathcal{I}, w_1 &\models obtaining\_a\_discount(x \colon customer) \leftarrow \{excellent(x \colon customer)\} \\ \mathcal{I}, w_1 &\models obtaining\_a\_discount(john \colon person) \\ \mathcal{I}, w_2 &\models animal(john \colon person) \end{aligned}$$

4 Knowledge Base Reasoning with Rigid Properties

This section develops two knowledge base reasoning systems for our order-sorted logic. We first extend the Horn-clause calculus [23] for a single knowledge base by incorporating sorted and unsorted substitutions and inference rules of type predicates. Second, using the calculus, we develop a rigid-property derivation system for many separated knowledge bases.

4.1 Extended Horn-Clause Calculus

Each knowledge base is constructed by a finite set of *sorted* clauses (i.e., ill-sorted expressions are excluded) as follows.

Definition 14 (Knowledge Base) *Let $\Sigma = (T, N, \Omega, \leq)$ be a sorted signature. A knowledge base $\mathcal{K}$ is a finite set of sorted clauses in Σ.*

Let θ^u be an unsorted substitution. We define the function $\iota(\theta^u) = \{x\colon s/t \in \theta^u \mid t \notin \mathcal{T}_s\}$ in order to obtain the ill-sorted operations from θ^u.

In the following, we define inference rules for the Horn-clause calculus with sorted and unsorted substitutions. This extended calculus can derive syntactically ill-sorted expressions by applying unsorted substitutions, but the derived ill-sorted expressions are semantically well-sorted in the rigid Σ-models.

Definition 15 (Sorted Horn-Clause Calculus) *Let C be a sorted or ill-sorted ground clause and $\mathcal{K}$ be a knowledge base. A derivation of C from $\mathcal{K}$ (denoted $\mathcal{K} \vdash C$) 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, $\mathcal{K} \vdash (L \leftarrow G)\theta$.*
- **Cut rule:** *Let $L \leftarrow G$ and $L' \leftarrow G' \cup \{L\}$ be ground clauses. If $\mathcal{K} \vdash L \leftarrow G$ and $\mathcal{K} \vdash L' \leftarrow G' \cup \{L\}$, then $\mathcal{K} \vdash L' \leftarrow G \cup G'$.*
- **Subsort rule:** *Let $p_s(t) \leftarrow G$ and $p_{s'}(t) \leftarrow G$ be ground clauses. If $\mathcal{K} \vdash p_s(t) \leftarrow G$ and $s \leq s'$, then $\mathcal{K} \vdash p_{s'}(t) \leftarrow G$.*
- **Type predicate rule:** *Let t be a sorted ground term in $\mathcal{T}$. If $sort(t) \leq \tau$, then $\mathcal{K} \vdash p_\tau(t) \leftarrow$.*
- **Unsorted type predicate rule:** *Let t be a sorted term in $\mathcal{T}$, let θ^u be an unsorted ground substitution for t where $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$, and let $p_{s_i}(t_i) \leftarrow G_i$ be a ground clause. If $sort(t) \leq \tau$ and $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$, then $\mathcal{K} \vdash p_\tau(t\theta^u) \leftarrow G_1 \cup \cdots \cup G_n$.*
- **Unsorted substitution rule:** *Let $L \leftarrow G \in \mathcal{K}$, let θ^u be an unsorted ground substitution for $L \leftarrow G$ where $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$, and let $p_{s_i}(t_i) \leftarrow G_i$ be a ground clause. If $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$, then $\mathcal{K} \vdash (L \leftarrow G \cup G_1 \cup \cdots \cup G_n)\theta^u$.*

We write $\mathcal{K} \vdash L$ if $\mathcal{K} \vdash L \leftarrow$. The sorted substitution rule and the cut rule serve as sorted inference rules in ordinary order-sorted logic. The subsort rule actualizes an inference of the implication form $p_s(t) \Rightarrow p_{s'}(t)$ with respect to the subsort relation $s < s'$. The type predicate rule derives axioms $p_\tau(t)$ of type predicates with $sort(t) \leq \tau$. For example, $animal(john\colon person)$ is valid if $person \leq animal$. Furthermore, the unsorted type predicate and unsorted substitution rules are unsorted variants of the type predicate and sorted substitution rules, respectively. These inference rules are based on the sorted resolution system with sort predicates in [6]. In order to deal with the rigidity of sorts, we distinguish types and non-rigid sorts in the calculus. In particular, the unsorted type predicate and unsorted substitution rules are necessary for reasoning over non-rigid sorted terms. For a non-rigid sort $customer \in N$, if $customer(john\colon person)$ is true, then $x\colon customer$ can be unsortedly substituted with a sorted term $john\colon person$, while the sort of $john\colon person$ is not a subsort of $customer$.

$\mathcal{K}_1 \vdash obtaining\_a\_discount(john\colon person)$:

$$\dfrac{excellent(john\colon person) \leftarrow \quad \dfrac{\dfrac{male\_customer(john\colon person) \leftarrow}{customer(john\colon person) \leftarrow}\,(subsort)}{obtaining\_a\_discount(john\colon person) \leftarrow \{excellent(john\colon person)\}}\,(unsorted\ substitution)}{obtaining\_a\_discount(john\colon person) \leftarrow}\,(cut)$$

$\mathcal{K}_3 \vdash canfly(peter\colon animal)$:

$$\dfrac{canfly(peter\colon animal) \leftarrow \{bird(peter\colon animal)\} \quad \dfrac{canary(peter\colon animal) \leftarrow}{bird(peter\colon animal) \leftarrow}\,(subsort)}{canfly(peter\colon animal) \leftarrow}\,(cut)$$

Figure 5: An example of derivations

Example 4 *Suppose we have the sorted signature $\Sigma = (T, N, \Omega, \leq^*)$ of Example 2. Consider the following knowledge bases $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$:*

$$\begin{aligned}
\mathcal{K}_1 \;=\; & \{\ male_customer(john\colon person) \leftarrow, \\
& \quad excellent(john\colon person) \leftarrow, \\
& \quad obtaining_a_discount(x\colon customer) \leftarrow \{excellent(x\colon customer)\}\ \} \\
\mathcal{K}_2 \;=\; & \{\ pet_seller(mary\colon person) \leftarrow, \\
& \quad cares_about(mary\colon person, x\colon customer) \leftarrow, \\
& \quad cares_about(mary\colon person, x\colon bird) \leftarrow\ \} \\
\mathcal{K}_3 \;=\; & \{\ canary(peter\colon animal) \leftarrow, \\
& \quad canfly(x\colon animal) \leftarrow \{bird(x\colon animal)\}\ \} \\
\mathcal{K}_4 \;=\; & \{\ father(tony\colon animal, peter\colon animal) \leftarrow, \\
& \quad bird(y\colon animal) \leftarrow \{bird(x\colon animal), \\
& \quad father(y\colon animal, x\colon animal)\}\ \}
\end{aligned}$$

Figure 5 shows derivations from $\mathcal{K}_1$ and $\mathcal{K}_3$ in the sorted Horn-clause calculus. In the first, we can derive the ill-sorted expression

$$obtaining\_a\_discount(john\colon person)$$

where for person $\in T$, customer $\in N$, person $\not\leq$ customer, and obtaining_a_discount: customer $\in \Omega$. This conclusion is obtained by an application of the unsorted substitution rule to the clauses

$$\begin{aligned}
& customer(john\colon person) \leftarrow, \\
& obtaining_a_discount(x\colon customer) \leftarrow \{excellent(x\colon customer)\}
\end{aligned}$$

with the unsorted substitution

$$\theta^u = \{x\colon customer/john\colon person\}.$$

We show the soundness and completeness of the Horn-clause calculus as follows.

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

Proof. This is proven by induction on the height n of a derivation tree of $\mathcal{K} \vdash L$. Let $\mathcal{I} = (M, \alpha)$ with $M = (W, U, I)$ be a rigid Σ-interpretation and let w be a world in W.

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

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

- Sorted substitution rule. We have $L \leftarrow G \in \mathcal{K}$ and θ be a sorted ground substitution for $L \leftarrow G$. So, $\mathcal{I}, w \models L \leftarrow G$. By Lemma 1, $\mathcal{I}, w \models (L \leftarrow G)\theta$.

- Type predicate rule. We have $sort(t) \leq \tau$ where t is a sorted ground term. So, $[\![t]\!]_{w,\alpha} \subseteq I_w(\tau)$ because t is a well-sorted term in $\mathcal{T}_\tau$. So, $\mathcal{I}, w \models p_\tau(t)$.

- Cut rule. Because $\mathcal{K} \vdash L \leftarrow G$ and $\mathcal{K} \vdash L' \leftarrow G' \cup \{L\}$, by induction hypothesis, $\mathcal{I}, w \models L \leftarrow G$ and $\mathcal{I}, w \models L' \leftarrow G' \cup \{L\}$. Hence, $\mathcal{I}, w \models L \leftarrow G \cup G'$.

- Subsort rule. Because $\mathcal{K} \vdash p_s(t) \leftarrow G$ and $s < s'$, by induction hypothesis, $\mathcal{I}, w \models p_s(t) \leftarrow G$ where $I_w(s) \subseteq I_w(s')$. Thus, $\mathcal{I}, w \models p_{s'}(t) \leftarrow G$.

- Unsorted type predicate rule. Since $sort(t) \leq \tau$ and $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$, by induction hypothesis, $\mathcal{I}, w \models p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{I}, w \models p_{s_n}(t_n) \leftarrow G_n$. If $\mathcal{I}, w \models G_1 \cup \cdots \cup G_n$, then $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_n}(t_n)$. Now we want to show $\mathcal{I}, w \models p_\tau(t\theta^u)$ where θ^u is an unsorted ground substitution for t and $\iota(\theta^u) = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$. If $t = c \colon s$, then $[\![t]\!]_{w,\alpha} \in I_w(\tau)$. This is because $sort(t) \leq \tau$. If $t = x \colon s$ and $x \colon s \notin \{x_1 \colon s_1, \ldots, x_n \colon s_n\}$, then $[\![t\theta^u]\!]_{w,\alpha} \in I_w(\tau)$. If $t = x_i \colon s_i$, then $sort(t) = s \leq \tau$ and so $I_w(s) \subseteq I_w(\tau)$. Then, $[\![\theta^u(x_i \colon s_i)]\!]_{w,\alpha} = [\![t_i]\!]_{w,\alpha} \in I_w(s) \subseteq I_w(\tau) \subseteq I_w(p_\tau)$. If $t = f(a_1, \ldots, a_r) \colon \tau'$ where $f \colon \tau_1 \times \cdots \times \tau_r \to \tau'$, by Definition 12, $[\![a_1]\!]_{w,\alpha} \in I_w(\tau_1), \ldots, [\![a_r]\!]_{w,\alpha} \in I_w(\tau_r)$. By Definition 9-3-(d), $[\![f(a_1, \ldots, a_r) \colon \tau']\!]_{w,\alpha} \in I_w(\tau') \subseteq I_w(\tau) \subseteq I_w(p_\tau)$. Hence, $\mathcal{I}, w \models p_\tau(t\theta^u)$. Therefore, $\mathcal{I}, w \models p_\tau(t\theta^u) \leftarrow G_1 \cup \cdots \cup G_n$.

- Unsorted substitution rule. We have $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$. By induction hypothesis, $\mathcal{I}, w \models p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{I}, w \models p_{s_n}(t_n) \leftarrow G_n$. If $\mathcal{I}, w \models (G \cup G_1 \cup \cdots \cup G_n)\theta^u$, then $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_n}(t_n)$. $L \leftarrow G \in \mathcal{K}$ implies $\mathcal{I}, w \models L \leftarrow G$. So, for all $d_1 \in I_w(s'_1), \ldots, d_k \in I_w(s'_k)$, $\mathcal{I}\alpha_w[y_1 \colon s'_1/d_1, \ldots, y_k \colon s'_k/d_k]$, $w \models G$ implies $\mathcal{I}\alpha_w[y_1 \colon s'_1/d_1, \ldots, y_k \colon s'_k/d_k], w \models L$ where $Var\ (L \leftarrow G) = \{y_1 \colon s'_1, \ldots, y_k \colon s'_k\}$. Similar to the proof of the unsorted type predicate rule, $\mathcal{I}, w \models (L \leftarrow G \cup G_1 \cup \cdots \cup G_n)\theta^u$. ∎

Let θ be a sorted substitution and C be a sorted clause. $C\theta$ is called a sorted instance of C. The set of all sorted ground instances of C is denoted by $ground(C)$. We write $ground(\mathcal{K})$ for $\bigcup_{C \in \mathcal{K}} ground(C)$. The set of derivable sorted and ill-sorted terms for sort s in $\mathcal{K}$ is defined as $\{t \mid \mathcal{K} \vdash p_s(t)\}$. We define the set of derivable terms for all sorts in $\mathcal{K}$ by $\bigcup_{s \in T \cup N} \{t \mid \mathcal{K} \vdash p_s(t)\}$. The set of rigid derivable sorted and ill-sorted terms for sort s

in $\mathcal{K}$ is defined as $\{t \mid \mathcal{K} \vdash_{\mathcal{S}} p_s(t)\}$. We define the set of rigid derivable terms for all sorts in $\mathcal{K}$ by $\bigcup_{s \in T \cup N}\{t \mid \mathcal{K} \vdash_{\mathcal{S}} p_s(t)\}$. To prove the completeness of the Horn-clause calculus and the rigid-property derivation system, we construct a Herbrand model for a finite set of knowledge bases.

Definition 16 (Herbrand Model) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_m\}$ be a finite set of knowledge bases. A Herbrand model (resp. rigid Herbrand model) M_H for $\mathcal{S}$ is a tuple $(W_{\mathcal{S}}, U_H, I_H)$ such that*

1. *$W_{\mathcal{S}} = \{w_{\mathcal{K}_1}, \ldots, w_{\mathcal{K}_m}\}$,*
2. *$U_H = \{t \mid \mathcal{K}_i \vdash p_s(t)$ for $\mathcal{K}_i \in \mathcal{S},\ s \in T \cup N\}$ (resp. $U_H = \{t \mid \mathcal{K}_i \vdash_{\mathcal{S}} p_s(t)$ for $\mathcal{K}_i \in \mathcal{S},\ s \in T \cup N\}$),*
3. *$I_H = \{I_{w_{\mathcal{K}_i}} \mid w_{\mathcal{K}_i} \in W_{\mathcal{S}}\}$ is the set of functions for all $w_{\mathcal{K}_i} \in W_{\mathcal{S}}$ with the following conditions:*
   (a) *$I_{w_{\mathcal{K}_i}}(s) = \{t \mid \mathcal{K}_i \vdash p_s(t)\}$ (resp. $I_{w_{\mathcal{K}_i}}(s) = \{t \mid \mathcal{K} \vdash_{\mathcal{S}} p_s(t)\}$),*
   (b) *if $c \in C$ and $c\colon \to \tau \in \Omega$, then $I_{w_{\mathcal{K}_i}}(c) = c\colon \tau$,*
   (c) *if $f \in F_n$ and $f\colon \tau_1 \times \cdots \times \tau_n \to \tau \in \Omega$, then $I_{w_{\mathcal{K}_i}}(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)\colon \tau$ where $t_k \in I_{w_{\mathcal{K}_i}}(\tau_k)$ for $1 \leq k \leq n$,*
   (d) *if $p \in P_n$ and $p\colon s_1 \times \cdots \times s_n \in \Omega$, then $I_{w_{\mathcal{K}_i}}(p) = \{(t_1, \ldots, t_n) \in I_{w_{\mathcal{K}_i}}(s_1) \times \cdots \times I_{w_{\mathcal{K}_i}}(s_n) \mid \mathcal{K}_i \vdash p(t_1, \ldots, t_n)$ (resp. $\mathcal{K}_i \vdash_{\mathcal{S}} p(t_1, \ldots, t_n)$)$\}$.*

A (rigid) Herbrand interpretation $\mathcal{I}_H$ for $\mathcal{S}$ is a pair (M_H, α) such that M_H is a (rigid) Herbrand model for $\mathcal{S}$ and α is a set of variable assignments on M_H. Let $L \leftarrow G$ be a clause with $Var(L \leftarrow G) = \{x_1\colon s_1, \ldots, x_n\colon s_n\}$. We define $ground^+_{\mathcal{K}_i}(L \leftarrow G)$ as the set of sorted and ill-sorted ground clauses $(L \leftarrow G)\theta^u$ with $\theta^u = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$ where $\mathcal{K}_i \vdash p_{s_k}(t_k)$ for $1 \leq k \leq n$.

Note that many worlds $w_{\mathcal{K}_1}, \ldots, w_{\mathcal{K}_m}$ in the Herbrand model are not necessary to show the completeness of the Horn-clause calculus for a single knowledge base $\mathcal{K}$, but they are used to prove the completeness of rigid-property derivation for a finite set $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ of knowledge bases. To avoid introducing additional semantics, the Herbrand model with many worlds is employed as a unified model in the proofs of completeness for both. We show that a rigid Herbrand interpretation is a rigid Σ-interpretation for each knowledge base $\mathcal{K}_i$ in $\mathcal{S}$.

Lemma 2 *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases. Let $\mathcal{I}_{\mathcal{S}}$ be a Herbrand interpretation for $\mathcal{S}$ and $L \leftarrow G$ be a clause. Then, $\mathcal{I}_{\mathcal{S}}$ has the following properties:*

1. *$\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models L \leftarrow G$ if and only if $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models ground^+_{\mathcal{K}_i}(L \leftarrow G)$.*
2. *$\mathcal{I}_{\mathcal{S}}$ is a rigid Σ-interpretation such that $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models \mathcal{K}_i$.*

Proof. 1. ($\Rightarrow$) Let $L \leftarrow G$ be a clause with $Var(L \leftarrow G) = \{x_1\colon s_1, \ldots, x_n\colon s_n\}$. Let $\theta^u = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$ be a sorted or ill-sorted substitution such that $(L \leftarrow G)\theta^u \in ground^+_{\mathcal{K}_i}(L \leftarrow G)$. By Definition 16, we have $t_1 \in I_{w_{\mathcal{K}_i}}(s_1)$, $\ldots$, $t_n \in I_{w_{\mathcal{K}_i}}(s_n)$. Hence,

$\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models (L \leftarrow G)\theta^u$. ($\Leftarrow$) Let θ be a sorted ground substitution for $Var(L \rightarrow G)$. So, $(L \rightarrow G)\theta \in ground(L \leftarrow G)$. By the assumption and $ground(L \rightarrow G) \subseteq ground^+_{\mathcal{K}_i}(L \leftarrow G)$, $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models (L \leftarrow G)\theta$. By Definition 16, $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models L \leftarrow G$.

2. We show that $\mathcal{I}_\mathcal{S}$ is a rigid Σ-interpretation. By Definition 16, the conditions 1,2,3-(a) and 3-(e) of rigid Σ-models (Definition 9) hold. The condition 3-(c) is shown by the following. If $c \in C$ and $c\colon \rightarrow \tau \in \Omega$, then we have $I_{w_{\mathcal{K}_i}}(c) = c\colon \tau$ (by Definition 16 3-(b)). By the type predicate rule in the Horn-clause calculus, $\mathcal{K}_i \vdash p_\tau(c\colon \tau)$. So, by Definition 16 3-(a), $c\colon \tau \in I_{w_{\mathcal{K}_i}}(\tau)$. Thus, $I_{w_{\mathcal{K}_i}}(c) \in I_{w_{\mathcal{K}_i}}(\tau)$. The conditions 3-(b) and 3-(d) can be shown by the subsort rule and unsorted type predicate rule. Furthermore, by the definition of $\mathcal{T}_{0,s}$ and Definition 16, we can derive that $\mathcal{I}_\mathcal{S}$ is a rigid Σ-interpretation.

Next, we prove that $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models \mathcal{K}_i$. Let $L \leftarrow G \in \mathcal{K}_i$ where $Var(L \leftarrow G) = \{x_1\colon s_1, \ldots, x_n\colon s_n\}$. So we want to show $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models ground^+_{\mathcal{K}_i}(L \leftarrow G)$. Let $L' \leftarrow G' \in ground^+_{\mathcal{K}_i}(L \leftarrow G)$. So, there is an unsorted ground substitution $\theta^u = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$ such that $(L \leftarrow G)\theta^u = L' \leftarrow G'$. By definition of $ground^+_{\mathcal{K}_i}(L \leftarrow G)$, $\mathcal{K}_i \vdash p_{s_k}(t_k)$ for $1 \leq k \leq n$. Suppose $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models \{L_1, \ldots, L_n\}\theta^u$ where $G = \{L_1, \ldots, L_n\}$. By definition 16, $\mathcal{K}_i \vdash L_1\theta^u$, $\ldots$, $\mathcal{K}_i \vdash L_n\theta^u$. Then, $\mathcal{K}_i \vdash (L \leftarrow \{L_1, \ldots, L_n\})\theta^u$ (by the unsorted substitution rule and $\mathcal{K}_i \vdash p_{s_k}(t_k)$ for $1 \leq k \leq n$). By the cut rule, $\mathcal{K}_i \vdash L\theta^u$. Hence, by Definition 16, $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models L\theta^u$. Therefore, $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models (L \leftarrow G)\theta^u$. By the first statement in Lemma 2, we obtain $\mathcal{I}_\mathcal{S}, w_{\mathcal{K}_i} \models L \leftarrow G$. ∎

The Herbrand model and the above lemma will be used to show the completeness of the Horn-clause calculus as follows.

Theorem 2 (Completeness of Horn-Clause Calculus) *Let $\mathcal{K}$ be a knowledge base and L be a sorted or ill-sorted ground atom. If $\mathcal{K} \models L$, then $\mathcal{K} \vdash L$.*

Proof. Assume that $\mathcal{K}$ is a particular knowledge base in a finite set $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ of knowledge bases. So, by Definition 16, a Herbrand Model can be constructed for $\mathcal{S}$ in order to obtain a rigid Σ-interpretation such that $\mathcal{I}_\mathcal{S}, w_\mathcal{K} \models \mathcal{K}$. By Lemma 2, $\mathcal{I}_\mathcal{S}, w_\mathcal{K} \models \mathcal{K}$. Hence, we have $\mathcal{I}_\mathcal{S}, w_\mathcal{K} \models L$. If $L = p(t_1, \ldots, t_n)$, then $(t_1, \ldots, t_n) \in I_{w_{\mathcal{K}_i}}(p)$. Therefore, $\mathcal{K} \vdash p(t_1, \ldots, t_n)$ by the condition 3-(a) of Definition 16. ∎

4.2 Rigid-Property Derivation in Knowledge Bases

In general, the conclusions in each knowledge base are not derivable from another knowledge base since knowledge bases are separately constructed for their respective situations. Nevertheless, each knowledge base can derive something from the rigid-property information in other knowledge bases. In order to elaborate on this idea, we present a derivation system of rigid properties in a finite set of knowledge bases called *a rigid-property derivation system.*

Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases, where $\mathcal{K}_1, \ldots, \mathcal{K}_n$ have the same sorted signature Σ. A rigid Σ-model $M = (W_\mathcal{S}, U, I)$ is said to be a rigid Σ-model for $\mathcal{S}$ if $W_\mathcal{S} = \{w_{\mathcal{K}_1}, \ldots, w_{\mathcal{K}_n}\}$ is the set of knowledge base worlds of $\mathcal{K}_1, \ldots, \mathcal{K}_n$. When we determine the consequence $\mathcal{K}_i \models L$, we will consider only rigid Σ-models having one world for each $\mathcal{K}_i$ in $\mathcal{S}$. We denote $\mathcal{K}_i \models_\mathcal{S} F$ if for every rigid Σ-interpretation $\mathcal{I}$ for $\mathcal{S}$, $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1$, $\ldots$, $\mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n$ imply $\mathcal{I}, w_{\mathcal{K}_i} \models F$. Let us denote the set of sorted and ill-sorted ground atoms by $\mathcal{A}^+_0$. We define the theory of $\mathcal{K}$ as $Th(\mathcal{K}) = \{L \in \mathcal{A}^+_0 \mid \mathcal{K} \vdash L\}$.

In order to define rigid-property derivation, an expansion of knowledge bases is introduced. The expansion of each knowledge base is obtained from other knowledge bases by propagating rigid-property information.

Definition 17 (Expansion of Knowledge Bases) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases. The expanded knowledge bases $\mathcal{K}_i^m$ of $\mathcal{K}_i$ in $\mathcal{S}$ are defined by the following:*

$$\begin{aligned} \mathcal{K}_i^0 &= \mathcal{K}_i \\ \mathcal{K}_i^{m+1} &= \mathcal{K}_i^m \cup \Delta(Th(\mathcal{K}_1^m) \cup \cdots \cup Th(\mathcal{K}_n^m)) \end{aligned}$$

where $\Delta(X) = \{p_\tau(t) \in \mathcal{A}_0^+ \mid \tau \in T \text{ and } p_\tau(t) \in X\}$

Each knowledge base $\mathcal{K}_i$ is expanded to $\mathcal{K}_i^0, \mathcal{K}_i^1, \ldots$ by adding rigid atomic formulas $p_\tau(t)$ derivable in one of the knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n \in \mathcal{S}$. Using this expansion, we define rigid-property derivation in a finite set of knowledge bases.

Definition 18 (Rigid-Property Derivation in $\mathcal{S}$) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases. A ground clause $L \leftarrow G$ is derivable from $\mathcal{K}_i$ in $\mathcal{S}$ (denoted $\mathcal{K}_i \vdash_{\mathcal{S}} L \leftarrow G$) if there exists an expanded knowledge base $\mathcal{K}_i^m$ of $\mathcal{K}_i$ such that $\mathcal{K}_i^m \vdash L \leftarrow G$.*

We will show the soundness and completeness of the rigid-property derivation system as follows.

Lemma 3 *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and $\mathcal{I}$ be a rigid Σ-interpretation for $\mathcal{S}$. If $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1^m, \ldots, \mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n^m$, then $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1^{m+1}, \ldots, \mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n^{m+1}$.*

Proof. Let $\mathcal{I} = (M, \alpha)$ be a rigid Σ-interpretation for $\mathcal{S}$. Let $x \in \{1, \ldots, n\}$. By Definition 17, $\mathcal{K}_x^{m+1} = \mathcal{K}_x^m \cup \Delta(Th(\mathcal{K}_1^m) \cup \cdots \cup Th(\mathcal{K}_n^m))$. By Theorem 1 and $Th(\mathcal{K}_x^m) = \{L \mid \mathcal{K}_x^m \vdash L\}$, $\mathcal{I}, w_{\mathcal{K}_1} \models Th(\mathcal{K}_1^m), \ldots, \mathcal{I}, w_{\mathcal{K}_n} \models Th(\mathcal{K}_n^m)$. Let $p_\tau(t) \in \Delta(Th(\mathcal{K}_1^m) \cup \cdots \cup Th(\mathcal{K}_n^m))$. Then, there exists $\mathcal{K}_l^m$ such that $p_\tau(t) \in Th(\mathcal{K}_l^m)$. By assumption and the soundness of the Horn-clause calculus, $\mathcal{I}, w_{\mathcal{K}_l} \models p_\tau(t)$. So, $[\![t]\!]_{w_{\mathcal{K}_l}\alpha} \in I_{w_{\mathcal{K}_l}}(\tau)$ because $\mathcal{I}$ is a rigid Σ-interpretation. Then, by Definition 10 (saying that for all $w_i, w_j \in W$, $I_{w_i}(\tau) = I_{w_j}(\tau)$, $I_{w_i}(c) = I_{w_j}(c)$, and $I_{w_i}(f) = I_{w_j}(f)$), $[\![t]\!]_{w_{\mathcal{K}_x}\alpha} \in I_{w_{\mathcal{K}_x}}(\tau)$. Hence, $\mathcal{I}, w_{\mathcal{K}_x} \models p_\tau(t)$. Therefore, we have $\mathcal{I}, w_{\mathcal{K}_x} \models \mathcal{K}_x^{m+1}$. ∎

The soundness of the rigid-property derivation system can be derived by Lemma 3.

Theorem 3 (Soundness of Rigid-Property Derivation) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and L be a sorted or ill-sorted ground atom. If $\mathcal{K}_i \vdash_{\mathcal{S}} L$, then $\mathcal{K}_i \models_{\mathcal{S}} L$.*

Proof. By Definition 18, there exists an expanded knowledge base $\mathcal{K}_i^m$ of $\mathcal{K}_i$ such that $\mathcal{K}_i^m \vdash L$. So we have $\mathcal{K}_i^m \models L$ (by Theorem 1). Let $\mathcal{I}$ be a rigid Σ-interpretation for $\mathcal{S}$. Assume $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1 (= \mathcal{K}_1^0), \ldots, \mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n (= \mathcal{K}_n^0)$. By Lemma 3, $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1^m, \ldots, \mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n^m$. Hence, $\mathcal{I}, w_{\mathcal{K}_i} \models L$. Therefore, we obtain the conclusion. ∎

Let $L \leftarrow G$ be a clause with $Var(L \leftarrow G) = \{x_1 \colon s_1, \ldots, x_n \colon s_n\}$. We now define $\mathit{rigid\text{-}ground}^{+}_{\mathcal{K}_i}(L \leftarrow G)$ as the set of sorted and ill-sorted ground clauses $(L \leftarrow G)\theta^u$ with $\theta^u = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$ where $\mathcal{K}_i \vdash_{\mathcal{S}} p_{s_k}(t_k)$ for $1 \leq k \leq n$.

Lemma 4 *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases. Let $\mathcal{I}_{\mathcal{S}}$ be a rigid Herbrand interpretation for $\mathcal{S}$ and $L \leftarrow G$ be a clause. Then, the following statements hold:*

1. *$\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models L \leftarrow G$ if and only if $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models \mathit{rigid\text{-}ground}^{+}_{\mathcal{K}_i}(L \leftarrow G)$.*
2. *$\mathcal{I}_{\mathcal{S}}$ is a rigid Σ-interpretation of $\mathcal{K}_i$ such that $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models \mathcal{K}_i$.*

Proof. 1. By Definition 16 and the definition of $\mathit{rigid\text{-}ground}^{+}_{\mathcal{K}_i}(L \leftarrow G)$, this can be shown in the similar way to the proof of Lemma 2.

2. First of all, we have to show that the derivation $\vdash$ in the Horn-clause calculus can be applied to the rigid-property derivation $\vdash_{\mathcal{S}}$. Namely, if $\mathcal{K}_i \vdash A_1$, …, $\mathcal{K}_i \vdash A_n$ derives $\mathcal{K}_i \vdash B$ in an inference rule, then $\mathcal{K}_i \vdash_{\mathcal{S}} A_1$, …, $\mathcal{K}_i \vdash_{\mathcal{S}} A_n$ derives $\mathcal{K}_i \vdash_{\mathcal{S}} B$. Suppose $\mathcal{K}_i \vdash_{\mathcal{S}} A_1$, …, $\mathcal{K}_i \vdash_{\mathcal{S}} A_n$. For each A_l, there exists an expanded knowledge base $\mathcal{K}_i^{m_l}$ of $\mathcal{K}_i$ such that $\mathcal{K}_i^{m_l} \vdash A_l$ (by Definition 18). Thus, if $m_k \geq m_j$ for $1 \leq j \leq n$, then $\mathcal{K}_i^{m_k} \vdash A_1$, …, $\mathcal{K}_i^{m_k} \vdash A_n$. Hence, $\mathcal{K}_i^{m_k} \vdash B$. Therefore, $\mathcal{K}_i \vdash_{\mathcal{S}} B$.

By the above claim and the same way of the proof of Lemma 2, we can show that $\mathcal{I}_{\mathcal{S}}$ is a rigid Σ-interpretation such that $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models \mathcal{K}_i$. Furthermore, we have to prove that $\mathcal{I}_{\mathcal{S}}$ is a *rigid* rigid Σ-interpretation, i.e., for each two worlds $w_{\mathcal{K}_i}, w_{\mathcal{K}_j}$ (1) $I_{w_{\mathcal{K}_i}}(\tau) = I_{w_{\mathcal{K}_j}}(\tau)$, (2) $I_{w_{\mathcal{K}_i}}(c) = I_{w_{\mathcal{K}_j}}(c)$ and (3) $I_{w_{\mathcal{K}_i}}(f) = I_{w_{\mathcal{K}_j}}(f)$. (1) let $t \in I_{w_{\mathcal{K}_i}}(\tau)$. By definition, $\mathcal{K}_i \vdash_{\mathcal{S}} p_\tau(t)$. So, by Definition 18, $\mathcal{K}_i^m \vdash p_\tau(t)$ where $\mathcal{K}_i^m$ is an expanded knowledge base of $\mathcal{K}_i$. This derives $p_\tau(t) \in Th(\mathcal{K}_i^m)$. By Definition 17, $p_\tau(t) \in \mathcal{K}_j^{m+1}$. Then, $\mathcal{K}_j^{m+1} \vdash p_\tau(t)$. Therefore, $t \in I_{w_{\mathcal{K}_j}}(\tau) (= \{t \mid \mathcal{K}_i \vdash_{\mathcal{S}} p_\tau(t)\})$. (2) let $c \colon \rightarrow \tau \in \Omega$. Then, $I_{w_i}(c) = c \colon \tau = I_{w_{\mathcal{K}_j}}(c)$ (by Definition 16). (3) let $f \colon \tau_1 \times \cdots \times \tau_n \rightarrow \tau \in \Omega$. Because $I_{w_i}(\tau) = I_{w_{\mathcal{K}_j}}(\tau)$ for each $\tau \in T$, we have $I_{w_{\mathcal{K}_i}}(f)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \colon \tau = I_{w_{\mathcal{K}_j}}(f)(t_1, \ldots, t_n)$ where $t_k \in I_{w_{\mathcal{K}_i}}(\tau_k)$ for $1 \leq k \leq n$ (by Definition 16). ∎

Finally, we prove the completeness of the rigid-property derivation system as follows.

Theorem 4 (Completeness of Rigid-Property Derivation) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and L be a sorted or ill-sorted ground atom. If $\mathcal{K}_i \models_{\mathcal{S}} L$, then $\mathcal{K}_i \vdash_{\mathcal{S}} L$.*

Proof. By Lemma 4, $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_1} \models \mathcal{K}_1, \ldots, \mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_n} \models \mathcal{K}_n$. Hence, by assumption, $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models L$. Therefore, $\mathcal{K}_i \vdash_{\mathcal{S}} L$ (by Definition 16). ∎

5 Query-Answering System with Rigid Properties

In this section, we provide a query-answering system equipped with rigid property derivation. If a user inputs a query (denoted by goal G) into a knowledge base, the system returns the instances $G\theta$ of the goal. In other words, the instances $G\theta$, which are regarded as the answers of the query, are true in the knowledge base. In order to embody a top-down reasoning mechanism, the query answering system is implemented using a linear resolution system.

5.1 Linear Resolution

Here, we develop a linear resolution system for our proposed order-sorted logic that includes sort predicates and rigidity. We extend the standard linear resolution system by incorporating additional resolution rules corresponding to the inference rules of the proposed Horn-clause calculus.

The extended linear resolution system consists of the following inference rules.

Definition 19 (Linear Resolution System)

- **Sorted resolution rule:** *Let $L' \leftarrow G' \in \mathcal{K}$ and $L \in G$. If θ is a unifier of L and L', then $(G - \{L\})\theta \cup G'\theta$ is a goal derived from L and $L' \leftarrow G'$, written by*

$$G \xrightarrow{\theta}_{R1} (G - \{L\})\theta \cup G'\theta$$

- **Subsort resolution rule:** *Let $p_s(t) \in G$ and $p_{s'}(t') \leftarrow G' \in \mathcal{K}$. If $s' \leq s$ and θ is a unifier of t and t', then $(G - \{p_s(t)\})\theta \cup G'\theta$ is a goal derived from $p_s(t)$ and $p_{s'}(t') \leftarrow G'$, written by*

$$G \xrightarrow{\theta}_{R2} (G - \{p_s(t)\})\theta \cup G'\theta$$

- **Type-predicate resolution rule:** *Let $p_\tau(t) \in G$ and let θ be a sorted substitution. If $sort(t\theta) \leq \tau$, then $(G - \{p_\tau(t)\})\theta$ is a goal derived from $p_\tau(t)$, written by*

$$G \xrightarrow{\theta}_{R3} (G - \{p_\tau(t)\})\theta$$

- **Unsorted type-predicate resolution rule:** *Let $p_\tau(t) \in G$ and let θ^u be an unsorted substitution. If $\iota(\theta^u) = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$, $sort(t) \leq \tau$, and $\mathcal{K} \colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K} \colon \{p_{s_n}(t_n)\} \longrightarrow G_n$, then $(G - \{p_\tau(t)\})\theta^u \cup (G_1 \cup \cdots \cup G_n)\theta^u$ is a goal derived from $p_\tau(t)$, written by*

$$G \xrightarrow{\theta^u}_{R4} (G - \{p_\tau(t)\})\theta^u \cup (G_1 \cup \cdots \cup G_n)\theta^u$$

- **Unsorted resolution rule:** *Let $L' \leftarrow G' \in \mathcal{K}$ and $L \in G$, let θ_0^u be an unsorted substitution, and let θ be a sorted substitution. If $\mathcal{K} \colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K} \colon \{p_{s_n}(t_n)\} \longrightarrow G_n$, $\theta_0^u = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$, and there is a unifier θ of $L\theta_0^u$ and $L'\theta_0^u$, then $(G - \{L\})\theta^u \cup G'\theta^u \cup (G_1 \cup \cdots \cup G_n)\theta^u$ with $\theta^u = \theta_0^u \circ \theta$ is a goal derived from L and $L' \leftarrow G'$, written by*

$$G \xrightarrow{\theta^u}_{R5} (G - \{L\})\theta^u \cup G'\theta^u \cup (G_1 \cup \cdots \cup G_n)\theta^u$$

For ensuring soundness of rules, they are applied up to variable renaming (like in classical resolution), so a clause and the goal in which it is resolved do not share free variables.

We generally write Θ for a sorted substitution θ or unsorted substitution θ^u. We use the abbreviation $G \xrightarrow{\Theta} G'$ for each of $G \xrightarrow{\Theta}_{R1} G'$, …, $G \xrightarrow{\Theta}_{R5} G'$. An unrestricted resolvent is a resolvent if the unifier Θ is most general. Let $\mathcal{K}$ be a knowledge base. A finite sequence

$$\mathcal{K} \colon G_0 \xrightarrow{\Theta_1} G_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} G_n$$

is an unrestricted resolution of G_0 with respect to $\mathcal{K}$ ($n \geq 0$). We denote it by $\mathcal{K}\colon G_0 \xrightarrow{\Theta}\!\!\!\!\!\rightarrow G_n$ with $\Theta = \Theta_1 \cdots \Theta_n$. $\mathcal{K}\colon G_0 \xrightarrow{\Theta}\!\!\!\!\!\rightarrow G_n$ is called successful if $G_n = \emptyset$. We write $G_0 \longrightarrow\!\!\!\!\!\rightarrow \mathit{fail}$ if there exists no successful resolution of G_0. We write $G_0 \longrightarrow\!\!\!\!\!\rightarrow G_n$ when we do not need to emphasize the substitution Θ in $G_0 \xrightarrow{\Theta}\!\!\!\!\!\rightarrow G_n$. An unrestricted resolution is called a resolution if the unrestricted resolvents are resolvents. The composition $(\Theta_1 \cdots \Theta_n)\!\uparrow Var(G_0)$ of the substitutions to the variables in the initial goal G_0 is called a computed answer substitution. For $1 \leq i \leq n$, the restriction of Θ_i to the goal G_{i-1} (i.e. $\Theta_i\!\uparrow Var(G_{i-1})$) is denoted by $\Theta_i^{\uparrow}$. Moreover, we write $\Theta^{\uparrow}$ for $\Theta_1^{\uparrow} \cdots \Theta_n^{\uparrow}$. The following theorem guarantees the soundness of the linear resolution system.

Theorem 5 (Soundness of Linear Resolution) *Let $\mathcal{K}$ be a knowledge base and G be a goal. If there exists a successful resolution $\mathcal{K}\colon G \xrightarrow{\Theta}\!\!\!\!\!\rightarrow \emptyset$, then $\mathcal{K} \models G\Theta$.*

Proof. This theorem is proven by induction on the length n of a successful resolution of G. Let $\mathcal{K}$ be a knowledge base and let

$$\mathcal{K}\colon G \xrightarrow{\Theta_1} G_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

with $\Theta = \Theta_1 \cdots \Theta_n$ be a successful resolution of G. Suppose that $\mathcal{I}, w \models \mathcal{K}$.
Base case: $n = 1$.

1. If $G \xrightarrow{\theta_1}_{R1} \emptyset$, then θ_1 is a unifier of $L' \in \mathcal{K}$ and $L(= G)$, i.e., $L'\theta_1 = L\theta_1$. Then $\mathcal{I}, w \models L'\theta_1$. Therefore $\mathcal{K} \models G\theta_1$ since $L'\theta_1 = L\theta_1(= G\theta_1)$.

2. If $G \xrightarrow{\theta_1}_{R2} \emptyset$, then θ_1 is a unifier of t and t', i.e., $t\theta_1 = t'\theta_1$ where $p_s(t)(= G)$ and $p_{s'}(t') \leftarrow\!\!\in \mathcal{K}$. So, we can get $\mathcal{I}, w \models p_{s'}(t'\theta_1)$ since $I_w(s') \subseteq I_w(s)$ (by Definition 9).

3. If $G \xrightarrow{\theta_1}_{R3} \emptyset$, then $G = p_\tau(t)$. The substitution θ_1 is well-sorted. So, $[\![t\theta_1]\!]_{w,\alpha} \subseteq I_w(\tau)$. This follows $\mathcal{K} \models p_\tau(t\theta_1)$.

4. If $G \xrightarrow{\theta_1^u}_{R4} \emptyset$, then $G = p_\tau(t)$ and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow \emptyset$, $\ldots$, $\mathcal{K}\colon \{p_{s_r}(t_r)\} \longrightarrow \emptyset$. By induction hypothesis, we have $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_r}(t_r)$. These imply $\mathcal{K} \models p_\tau(t\theta_1^u)$ by the proof of Theorem 1.

5. If $G \xrightarrow{\theta_1^u}_{R5} \emptyset$, then θ_1^u is a unifier of $L' \in \mathcal{K}$ and $L(= G)$, i.e., $L'\theta_1^u = L\theta_1^u$, and $G = p_\tau(t)$ and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow \emptyset$, $\ldots$, $\mathcal{K}\colon \{p_{s_r}(t_r)\} \longrightarrow \emptyset$. By induction hypothesis, $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_r}(t_r)$. Hence, $\mathcal{I}, w \models L'\theta_1^u$ by the proof of Theorem 1. Therefore, $\mathcal{K} \models G\theta_1^u$ since $L'\theta_1^u = L\theta_1^u(= G\theta_1^u)$.

Induction step: $n > 1$.

1. If $G \xrightarrow{\theta_1}_{R1} G_1$, then

   $$G'\theta_1 \cup (G - \{L\})\theta_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

   is a resolution of $G_1(= G'\theta_1 \cup (G - \{L\})\theta_1)$ where $L \in G$, $L' \leftarrow G' \in \mathcal{K}$, and $L'\theta_1 = L\theta_1$. By induction hypothesis, $\mathcal{K} \models (G'\theta_1 \cup (G - \{L\})\theta_1)\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$. Then, $\mathcal{K} \models L'\theta_1\Theta(= L\theta_1\Theta)$ since $\mathcal{K} \models (L' \leftarrow G')\theta_1\Theta$. Therefore $\mathcal{K} \models G\theta_1\Theta$.

2. If $G \xrightarrow{\theta_1}_{R2} G_1$, then

$$G'\theta_1 \cup (G - \{p_s(t)\})\theta_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

is a resolution of $G_1 (= G'\theta_1 \cup (G - \{p_s(t)\})\theta_1)$ where θ_1 is a unifier of t and t', i.e., $t\theta_1 = t'\theta_1$, $p_s(t) \in G$, and $p_{s'}(t') \leftarrow G' \in \mathcal{K}$. By induction hypothesis, $\mathcal{I}, w \models G'\theta_1\Theta \cup (G - \{p_s(t)\})\theta_1\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$. By Definition 9, $\mathcal{I}, w \models p_s(t\theta_1\Theta)$ since $\mathcal{I}, w \models p_{s'}(t') \leftarrow G'$ and $I_w(s') \subseteq I_w(s)$. Hence, $\mathcal{K} \models G\theta_1\Theta$.

3. If $G \xrightarrow{\theta_1}_{R3} G_1$, then

$$(G - \{p_\tau(t)\})\theta_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

is a resolution of $G_1 (= (G - \{p_\tau(t)\})\theta_1)$ where $p_\tau(t) \in G$ and θ_1 is an unsorted substitution. By induction hypothesis, $\mathcal{I}, w \models (G - \{p_\tau(t)\})\theta_1\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$. By the proof of Theorem 1, $[\![t\theta_1\Theta]\!]_{w,\alpha} \subseteq I_w(\tau)$. It follows $\mathcal{K} \models p_\tau(t\theta_1\Theta)$, and so $\mathcal{K} \models G\theta_1\Theta$.

4. If $G \xrightarrow{\theta_1}_{R4} G_1$, then

$$(G - \{p_\tau(t)\})\theta_1^u \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

is a resolution of $G_1 (= (G - \{p_\tau(t)\})\theta_1^u \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u)$ where $p_\tau(t) \in G$ and θ^u is an unsorted substitution. By induction hypothesis, $\mathcal{I}, w \models (G - \{p_\tau(t)\})\theta_1^u\Theta \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$. By definition, $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G'_1$, $\ldots$, $\mathcal{K}\colon \{p_{s_r}(t_r)\} \longrightarrow G'_r$. By induction hypothesis, $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_r}(t_r)$. Therefore, by the proof of Theorem 1, $\mathcal{K} \models p_\tau(t\theta\Theta)$. So, $\mathcal{K} \models G\theta_1\Theta$.

5. If $G \xrightarrow{\theta_1}_{R5} G_1$, then

$$(G - \{L\})\theta^u \cup G'\theta_1^u \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

is a resolution of $G_1 (= (G - \{L\})\theta_1^u \cup G'\theta_1^u \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u)$ where $L' \leftarrow G' \in \mathcal{K}$, $L \in G$, and θ_1^u is an unsorted substitution. By induction hypothesis, $\mathcal{I}, w \models (G - \{L\})\theta_1^u\Theta \cup G'\theta_1^u\Theta \cup (G'_1 \cup \cdots \cup G'_r)\theta_1^u\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$. By definition, θ_1^u is a unifier of L' and L, i.e., $L'\theta_1^u = L\theta_1^u$, and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G'_1$, $\ldots$, $\mathcal{K}\colon \{p_{s_r}(t_r)\} \longrightarrow G'_r$. By induction hypothesis, $\mathcal{I}, w \models p_{s_1}(t_1), \ldots, \mathcal{I}, w \models p_{s_r}(t_r)$. Then, by the proof of Theorem 1, $\mathcal{I}, w \models L'\theta_1^u\Theta$. Therefore, $\mathcal{K} \models G\theta_1^u\Theta$ since $L'\theta_1^u = L\theta_1^u$. ∎

Lemma 5 *Let $\mathcal{K}$ be a knowledge base and $L \leftarrow G$ be a ground clause. If $\mathcal{K} \vdash L \leftarrow G$, then $\mathcal{K}\colon G \longrightarrow\!\!\!\!\rightarrow \emptyset$ implies $\mathcal{K}\colon L \longrightarrow\!\!\!\!\rightarrow \emptyset$.*

Proof. We show the lemma by induction on the length n of a derivation tree of $L \leftarrow G$ from $\mathcal{K}$.

$n = 1$: If $L \leftarrow G$ is derived by the sorted substitution rule in the Horn-clause calculus, then $L' \leftarrow G' \in \mathcal{K}$ where $L \leftarrow G$ is a ground instance of $L' \leftarrow G'$, i.e., $(L' \leftarrow G')\theta = L \leftarrow G$. Suppose $\mathcal{K}\colon G \longrightarrow\!\!\!\!\rightarrow \emptyset$. By the sorted resolution rule, it is easy to derive $\mathcal{K}\colon L \xrightarrow{\theta}_{R1} G \longrightarrow\!\!\!\!\rightarrow \emptyset$.

If $p_\tau(t) \leftarrow$ is derived by the type predicate rule, then $sort(t) \leq \tau$, and so $\mathcal{K}\colon p_\tau(t) \xrightarrow{\epsilon}_{R3} \emptyset$ (by the type-predicate resolution rule).

If $p_\tau(t) \leftarrow G$ is derived by the unsorted type predicate rule, then there exists an unsorted substitution θ^u such that for some term t', $t'\theta^u = t$, and $sort(t') \leq \tau$. Let $\iota(\theta^u) = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$. Then, we have $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$ and $G = G_1 \cup \cdots \cup G_n$. Hence, $\mathcal{K} \colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K} \colon \{p_{s_n}(t_n)\} \longrightarrow G_n$. Therefore, if $\mathcal{K} \colon G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$, then $\mathcal{K} \colon p_\tau(t) \xrightarrow{\theta^u}_{R4} G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$.

If $L \leftarrow G$ is derived by the unsorted substitution rule, then $L' \leftarrow G' \in \mathcal{K}$ where $L \leftarrow G$ is a ground instance of $L' \leftarrow G' \cup G_1 \cup \cdots \cup G_n$. That is, there exists an unsorted substitution θ^u such that $(L' \leftarrow G' \cup G_1 \cup \cdots \cup G_n)\theta^u = L \leftarrow G$. Let $\iota(\theta^u) = \{x_1 \colon s_1/t_1, \ldots, x_n \colon s_n/t_n\}$. So, we have $\mathcal{K} \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K} \vdash p_{s_n}(t_n) \leftarrow G_n$. By induction hypothesis, $\mathcal{K} \colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K} \colon \{p_{s_n}(t_n)\} \longrightarrow G_n$. Therefore, if $\mathcal{K} \colon G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$, then $\mathcal{K} \colon L \xrightarrow{\theta^u}_{R5} G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ (by the unsorted type-predicate resolution).

$n > 1$: If $L \leftarrow G$ is derived by the cut rule, by induction hypothesis, there are subsets G' and G'' of G such that $G = G' \cup G''$. So, $\mathcal{K} \colon G' \cup \{L'\} \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ implies $\mathcal{K} \colon L \longrightarrow\!\!\!\!\!\rightarrow \emptyset$, and $\mathcal{K} \colon G'' \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ implies $\mathcal{K} \colon L' \longrightarrow\!\!\!\!\!\rightarrow \emptyset$. Hence if $\mathcal{K} \colon G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$, then $\mathcal{K} \colon L \longrightarrow\!\!\!\!\!\rightarrow \emptyset$.

If $p_{s'}(t) \leftarrow G$ is derived by the subsort rule, then $\mathcal{K} \vdash p_s(t) \leftarrow G$ and $s' \leq s$. By induction hypothesis, if $\mathcal{K} \colon G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$, then $\mathcal{K} \colon p_s(t) \longrightarrow\!\!\!\!\!\rightarrow \emptyset$. Therefore, $\mathcal{K} \colon p_s(t) \xrightarrow{\epsilon}_{R2} p_{s'}(t) \longrightarrow\!\!\!\!\!\rightarrow \emptyset$.

In the case $n > 1$, the other rules cannot derive $L \leftarrow G$ because they have no antecedent. So, the proof has been done. ∎

Theorem 6 (Ground Completeness of Linear Resolution) *Let $\mathcal{K}$ be a knowledge base and G be a ground goal. If $\mathcal{K} \models G$, then there exists a resolution $\mathcal{K} \colon G \xrightarrow{\Theta} \!\!\!\!\!\rightarrow \emptyset$.*

Proof. Let $\mathcal{I}, w \models \mathcal{K}$. By assumption, $\mathcal{I}, w \models G$. Let $\mathcal{S}$ be a finite set of knowledge bases including $\mathcal{K}$. Then $\mathcal{I}_\mathcal{S}, w_\mathcal{K} \models G$ by Lemma 2. By Definition 16, $\mathcal{K} \vdash L$ in $\mathcal{K}$ for all $L \in G$. Hence, $\mathcal{K} \colon L \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ by Lemma 5. Therefore, we can obtain $\mathcal{K} \colon G \xrightarrow{\Theta} \!\!\!\!\!\rightarrow \emptyset$. ∎

Lemma 6 (Lifting) *Let $\mathcal{K}$ be a knowledge base. If $\mathcal{K}$ has an unrestricted resolution*

$$\mathcal{K} \colon G_0\Theta_0 \xrightarrow{\Theta_1} G_1 \xrightarrow{\Theta_2} \cdots \xrightarrow{\Theta_n} G_n,$$

then $\mathcal{K}$ has a resolution

$$\mathcal{K} \colon G_0 \xrightarrow{\Theta'_1} G'_1 \xrightarrow{\Theta'_2} \cdots \xrightarrow{\Theta'_n} G'_n$$

and the following holds.

(i) $\gamma_0 = \Theta_0$, and for $1 \leq i \leq n$, $(\gamma_{i-1}\!\upharpoonright\! Var(G_{i-1}))\Theta_i = \Theta'_i\gamma_i$ and $G_i = G'_i\gamma_i$, and

(ii) there exists a substitution γ'_n with $G_0\Theta_0\Theta_1\!\upharpoonright\!\cdots\Theta_n\!\upharpoonright = G_0\Theta'_1\!\upharpoonright\!\cdots\Theta'_n\!\upharpoonright\!\gamma'_n$.

Proof. We show the lemma by induction on the length n of a resolution. The proof is based on Lemmas 5.33 and Theorem 5.37 in [14]. ∎

We will show the completeness of the linear resolution system corresponding to the extended Horn-clause calculus.

Theorem 7 (Completeness of Linear Resolution) *Let $\mathcal{K}$ be a knowledge base, G be a goal, and Θ be a sorted or unsorted substitution. If $\mathcal{K} \models G\Theta$, then there exists a successful resolution $\mathcal{K} \colon G \xrightarrow{\Theta'} \!\!\!\!\!\rightarrow \emptyset$ such that $G\Theta = G\Theta'\!\upharpoonright\!\gamma$.*

Proof. We first introduce a new constant $c_i\colon s_i$ for every $x_i\colon s_i \in Var(G\Theta)$. Let β be a sorted substitution defined by $Dom(\beta) = Var(G\Theta)$ and $\beta(x_i\colon s_i) = c_i\colon s_i$. By definition, $\mathcal{K} \models G\Theta$ leads to $\mathcal{K} \models G\Theta\beta$. By Theorem 6, there exists a resolution $\mathcal{K}\colon G\Theta\beta \xrightarrow{\delta} \!\!\!\!\rightarrow \emptyset$. By Lemma 6, we have a resolution $\mathcal{K}\colon G \xrightarrow{\Theta_0} \!\!\!\!\rightarrow \emptyset$ with $G\Theta\beta\delta^{\uparrow} = G\Theta'^{\uparrow}\gamma$. Due to $Var(G\Theta\beta) = \emptyset$, $G\Theta\beta = G\Theta'^{\uparrow}\gamma$. So, $G\Theta'^{\uparrow}$ does not include the new constants $c_i\colon s_i$. Thus, $\gamma(y_i\colon s_i) = c_i\colon s_i$ holds for a variable $y_i\colon s_i$ (the sort of which is the same as of $x_i\colon s_i$). If $(y_i\colon s_i)\gamma_0 = x_i\colon s_i$ and $\gamma' = \{(a,b) \in \gamma | a \neq y_i\colon s_i\} \cup \gamma_0$, then $G\Theta = G\Theta'^{\uparrow}\gamma'$. ∎

5.2 Rigid-Property Resolution for Multiple Knowledge Bases

We develop an extended linear resolution system (called a rigid-property resolution system) that is more effective than the rigid-property derivation system. To be precise, the three linear resolution rules (sorted resolution, unsorted type-predicate resolution, and unsorted resolution rules) are extended to deal with the extraction of rigid property information in multiple knowledge bases.

The rigid-property resolution system contains the following resolution rules.

Definition 20 (Rigid-Property Resolution System) *Let $\mathcal{S}$ be a finite set of knowledge bases and let $\mathcal{K} \in \mathcal{S}$.*

- **Sorted resolution rule with rigidity:** *Let $p_\tau(t') \leftarrow G' \in \mathcal{K}'$ with $\mathcal{K}' \in \mathcal{S}/\{\mathcal{K}\}$ and $p_\tau(t) \in G$. If θ is a unifier of t and t', then $(G - \{p_\tau(t)\})\theta$ and $G'\theta$ are goals derived from $p_\tau(t)$ and $p_\tau(t') \leftarrow G'$, written by*

$$\mathcal{K}\colon G \xrightarrow[R1^+]{\theta} (G - \{p_\tau(t)\})\theta \quad \textit{and} \quad \mathcal{K}'\colon G'\theta$$

- **Unsorted type-predicate resolution rule with rigidity:** *Let $p_\tau(t) \in G$ and let θ^u be an unsorted substitution. If $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$, $sort(t) \leq \tau$, and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G_1$, …, $\mathcal{K}\colon \{p_{s_m}(t_m)\} \longrightarrow G_m$, then $(G - \{p_\tau(t)\})\theta^u \cup (G_1 \cup \cdots \cup G_m)\theta^u$, and $G_{m+1}\theta^u, \ldots, G_n\theta^u$ are goals derived from $p_\tau(t)$, written by*

$$\begin{gathered}\mathcal{K}\colon G \xrightarrow[R4^+]{\theta^u} (G - \{p_\tau(t)\})\theta^u \cup (G_1 \cup \cdots \cup G_m)\theta^u \\ \mathcal{K}'_{m+1}\colon \{p_{s_{m+1}}(t_{m+1})\} \longrightarrow G_{m+1}\theta^u, \ldots, \mathcal{K}'_n\colon \{p_{s_n}(t_n)\} \longrightarrow G_n\theta^u\end{gathered}$$

  where $\mathcal{K}'_{m+1}, \ldots, \mathcal{K}'_n \in \mathcal{S}/\{\mathcal{K}\}$.

- **Unsorted resolution rule with rigidity:** *Let $L' \leftarrow G' \in \mathcal{K}$ and $L \in G$ and let θ^u be an unsorted substitution. If $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$, θ^u is a unifier of L and L' and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G_1$, …, $\mathcal{K}\colon \{p_{s_m}(t_m)\} \longrightarrow G_m$, then $(G - \{L\})\theta^u \cup G'\theta^u \cup (G_1 \cup \cdots \cup G_m)\theta^u$, and $G_{m+1}\theta^u, \ldots, G_n\theta^u$ are goals derived from L and $L' \leftarrow G'$, written by*

$$\begin{gathered}\mathcal{K}\colon G \xrightarrow[R5^+]{\theta^u} (G - \{L\})\theta^u \cup G'\theta^u \cup (G_1 \cup \cdots \cup G_m)\theta^u \\ \mathcal{K}'_{m+1}\colon \{p_{s_{m+1}}(t_{m+1})\} \longrightarrow G_{m+1}\theta^u, \ldots, \mathcal{K}'_n\colon \{p_{s_n}(t_n)\} \longrightarrow G_n\theta^u\end{gathered}$$

  where $\mathcal{K}'_{m+1}, \ldots, \mathcal{K}'_n \in \mathcal{S}/\{\mathcal{K}\}$.

With respect to the resolution rules with rigidity, we use the abbreviation $G \xrightarrow{\Theta}_{+} G'$ for each of $G \xrightarrow{\Theta} G'$, $G \xrightarrow{\Theta}_{R1^+} G'$, $G \xrightarrow{\Theta}_{R4^+} G'$, and $G \xrightarrow{\Theta}_{R5^+} G'$. We denote a rigid-property resolution by $\mathcal{K}\colon G_0 \xrightarrow{\Theta}\!\!\!\!\!\rightarrow_{+} G_n$. We can say that there exists a successful rigid-property resolution of G in $\mathcal{K}$ if one of the following conditions holds:

1. There exists a successful resolution $\mathcal{K}\colon G \xrightarrow{\Theta'}\!\!\!\!\!\rightarrow \emptyset$ of G.
2. There exists a rigid-property resolution $\mathcal{K}\colon G \xrightarrow{\Theta'}\!\!\!\!\!\rightarrow_{+} G'$ of G such that $G' = \emptyset$ and, whenever a resolution rule with rigidity ($R1^+$, $R4^+$, or $R5^+$) is applied, there exist successful rigid-property resolutions of its newly created goals in the other knowledge bases ($\mathcal{K}'$ or $\mathcal{K}'_{m+1}, \ldots, \mathcal{K}'_n$).

Example 5 *Suppose we have the sorted signature $\Sigma = (T, N, \Omega, \leq^*)$ of Example 2 and the knowledge bases $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$ of Example 4. Consider a rigid-property resolution $\mathcal{K}_3\colon \{canfly(z\colon animal)\} \xrightarrow{\Theta}\!\!\!\!\!\rightarrow_{+} \emptyset$ of the goal $\{canfly(z\colon animal)\}$.*

First, the sorted resolution rule is applied to the goal in the knowledge base $\mathcal{K}_3$.

$$\mathcal{K}_3\colon \{canfly(z\colon animal)\} \xrightarrow{\theta_1}_{R1} \{bird(z_1\colon animal)\}$$

where $\theta_1 = \{z\colon animal/z_1\colon animal, x\colon animal/z_1\colon animal\}$.

Second, the subgoal $\{bird(z_1\colon animal)\}$ derives the empty clause if the sorted resolution rule with rigidity is successfully applied to another knowledge base $\mathcal{K}_4$.

$$\mathcal{K}_3\colon \{bird(z_1\colon animal)\} \xrightarrow{\theta_2}_{R1^+} \emptyset$$
$$\mathcal{K}_4\colon \{bird(tony\colon animal)\} \xrightarrow{\theta_3}_{R1} \{bird(peter\colon animal), father(peter\colon animal, tony\colon animal)\}$$

where $\theta_2 = \{z_1\colon animal/tony\colon animal\}$ and $\theta_3 = \emptyset$. To obtain the rigid property information "tony is a bird" from the knowledge base $\mathcal{K}_4$, the new goal $\{bird(tony\colon animal)\}$ is resolved by deriving the subgoal $\{bird(peter\colon animal), father(peter\colon animal, tony\colon animal)\}$ by applying the sorted resolution rule.

Moreover, the subgoal is resolved by applying the sorted resolution rule with rigidity if there is a successful resolution of the goal $\{bird(peter\colon animal)\}$ in another knowledge base $\mathcal{K}_3$.

$$\mathcal{K}_4\colon \{bird(peter\colon animal)\} \xrightarrow{\theta_4}_{R1^+} \emptyset$$
$$\mathcal{K}_3\colon \{bird(peter\colon animal)\} \xrightarrow{\theta_5}_{R2} \emptyset$$

where $\theta_4 = \emptyset$ and $\theta_5 = \emptyset$.

Therefore, there is a successful rigid-property resolution of the goal $\{canfly(z\colon animal)\}$ in $\mathcal{K}_3$.

We show the soundness of the rigid-property resolution system as follows.

Theorem 8 (Soundness of Rigid-Property Resolution) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and G be a goal. If there exists a successful rigid-property resolution $\mathcal{K}_i\colon G \xrightarrow{\Theta}\!\!\!\!\!\rightarrow_{+} \emptyset$ of G, then $\mathcal{K}_i \models_{\mathcal{S}} G\Theta$.*

Proof. This theorem is proven by induction on the length n of a successful rigid-property resolution of G. Let $\mathcal{K}$ be a knowledge base and let

$$\mathcal{K}_i\colon G \xrightarrow{\Theta_1} G_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$

with $\Theta = \Theta_1 \cdots \Theta_n$ be a successful rigid-property resolution of G.
Base case: $n = 1$. The resolution rules with rigidity ($R1^+$, $R4^+$, and $R5^+$) are the same as the corresponding linear resolution rules ($R1$, $R4$, and $R5$).

Induction step: $n > 1$. We show only the cases where resolution rules with rigidity ($R1^+$, $R4^+$, and $R5^+$) are applied as follows.

1. If $G \xrightarrow{\theta_1}_{R1^+} G_1$, then

$$\mathcal{K}\colon G'\theta_1 \cup (G - \{p_\tau(t)\})\theta_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$
$$\mathcal{K}'\colon G'\theta_1 \xrightarrow{\Theta'_2} G'_2 \xrightarrow{\Theta'_3} \cdots \xrightarrow{\Theta'_r} \emptyset$$

   is a resolution of $G_1 (= G'\theta_1 \cup (G - \{p_\tau(t)\})\theta_1)$ where $p_\tau(t) \in G$, $p_\tau(t') \leftarrow G' \in \mathcal{K}$, and $t'\theta_1 = t\theta_1$. By induction hypothesis, $\mathcal{K} \models_{\mathcal{S}} (G - \{p_\tau(t)\})\theta_1\Theta'$ with $\Theta = \Theta_2 \cdots \Theta_n$ and $\mathcal{K}' \models_{\mathcal{S}} G'\theta_1\Theta'$ with $\Theta' = \Theta'_2 \cdots \Theta'_r$. On the other hand, $\mathcal{K}' \models_{\mathcal{S}} p_\tau(t'\theta_1\Theta')(= p_\tau(t\theta_1\Theta'))$ since $\mathcal{K}' \models_{\mathcal{S}} (p_\tau(t') \leftarrow G')\theta_1\Theta'$. Therefore $\mathcal{K}' \models_{\mathcal{S}} p_\tau(t\theta_1\Theta')$. By Definition 9, $\mathcal{K} \models_{\mathcal{S}} p_\tau(t\theta_1\Theta')$, and so $\mathcal{K} \models_{\mathcal{S}} G\theta_1\Theta'$.

2. If $G \xrightarrow{\theta^u}_{R4^+} G_1$, then

$$\mathcal{K}\colon (G - \{p_\tau(t)\})\theta^u \cup (G'_1 \cup \cdots \cup G'_m)\theta^u \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$
$$\mathcal{K}'_{m+1}\colon \{p_{s_{m+1}}(t_{m+1})\} \longrightarrow G'_{m+1}\theta^u \xrightarrow{\Theta_{(m+1,2)}} G_{(m+1,2)} \xrightarrow{\Theta_{(m+1,3)}} \cdots \xrightarrow{\Theta_{(m+1,k_{m+1})}} \emptyset,$$
$$\cdots,$$
$$\mathcal{K}'_r\colon \{p_{s_r}(t_r)\} \longrightarrow G'_r\theta^u \xrightarrow{\Theta_{(r,2)}} G_{(r,2)} \xrightarrow{\Theta_{(r,3)}} \cdots \xrightarrow{\Theta_{(r,k_r)}} \emptyset$$

   is a resolution of $G_1 (= (G - \{p_\tau(t)\})\theta^u \cup (G'_1 \cup \cdots \cup G'_m)\theta^u)$ where $p_\tau(t) \in G$ and θ^u is an unsorted substitution. By induction hypothesis, $\mathcal{K} \models_{\mathcal{S}} (G - \{p_\tau(t)\})\theta^u\Theta \cup (G'_1 \cup \cdots \cup G'_m)\theta^u\Theta$ with $\Theta = \Theta_2 \cdots \Theta_n$, and $\mathcal{K}'_{m+1} \models_{\mathcal{S}} G'_{m+1}\theta^u\Theta'_{m+1}$ with $\Theta'_{m+1} = \Theta_{(m+1,2)} \cdots \Theta_{(m+1,k_{m+1})}$, $\ldots$, $\mathcal{K}'_r \models_{\mathcal{S}} G'_r\theta^u\Theta'_r$ with $\Theta'_r = \Theta_{(r,2)} \cdots \Theta_{(r,k_r)}$. By definition, $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G_1$, $\ldots$, $\mathcal{K}\colon \{p_{s_m}(t_m)\} \longrightarrow G'_m$, Then, $\mathcal{K} \models_{\mathcal{S}} p_{s_1}(t_1), \ldots,$ $\mathcal{K} \models_{\mathcal{S}} p_{s_m}(t_m)$. For $m + 1 \leq j \leq r$, $\mathcal{K}'_j \models_{\mathcal{S}} p_{s_j}(t_j\theta\Theta'_j)$, and by Definition 9, $\mathcal{K} \models_{\mathcal{S}} p_{s_j}(t_j\theta\Theta'_j)$. According to the proof of Theorem 1, $\mathcal{K} \models_{\mathcal{S}} p_\tau(t\theta\Theta)$. Hence, $\mathcal{K} \models_{\mathcal{S}} G\theta^u\Theta$.

3. If $G \xrightarrow{\theta^u}_{R5^+} G_1$, then

$$\mathcal{K}\colon (G - \{L\})\theta^u \cup G'\theta^u \cup (G'_1 \cup \cdots \cup G'_m)\theta^u \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} \emptyset$$
$$\mathcal{K}'_{m+1}\colon \{p_{s_{m+1}}(t_{m+1})\} \longrightarrow G'_{m+1}\theta^u \xrightarrow{\Theta_{(m+1,2)}} G_{(m+1,2)} \xrightarrow{\Theta_{(m+1,3)}} \cdots \xrightarrow{\Theta_{(m+1,k_{m+1})}} \emptyset,$$
$$\cdots,$$
$$\mathcal{K}'_r\colon \{p_{s_r}(t_r)\} \longrightarrow G'_r\theta^u \xrightarrow{\Theta_{(r,2)}} G_{(r,2)} \xrightarrow{\Theta_{(r,3)}} \cdots \xrightarrow{\Theta_{(r,k_r)}} \emptyset$$

is a resolution of $G_1(=(G-\{L\})\theta^u \cup G'\theta^u \cup (G'_1 \cup \cdots \cup G'_m)\theta^u)$ where $L' \leftarrow G' \in \mathcal{K}$, $L \in G$, and θ^u is an unsorted substitution. By induction hypothesis, $\mathcal{K} \models_{\mathcal{S}} (G - \{L\})\theta^u\Theta \cup G'\theta^u\Theta \cup (G'_1 \cup \cdots \cup G'_m)\theta^u\Theta$ with $\Theta = \Theta_2 \cdots \Theta_r$, and $\mathcal{K}'_{m+1} \models_{\mathcal{S}} G'_{m+1}\theta^u\Theta'_{m+1}$ with $\Theta'_{m+1} = \Theta_{(m+1,2)} \cdots \Theta_{(m+1,k_{m+1})}, \ldots, \mathcal{K}'_r \models_{\mathcal{S}} G'_r\theta^u\Theta'_r$ with $\Theta'_r = \Theta_{(r,2)} \cdots \Theta_{(r,k_r)}$. By definition, θ^u is a unifier of L' and L, i.e., $L'\theta^u = L\theta^u$, and $\mathcal{K}\colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K}\colon \{p_{s_m}(t_m)\} \longrightarrow G'_m$. For $m+1 \le j \le r$, $\mathcal{K}'_j \models_{\mathcal{S}} p_{s_j}(t_j\theta\Theta'_j)$, and by Definition 9, $\mathcal{K} \models_{\mathcal{S}} p_{s_j}(t_j\theta\Theta'_j)$. Then, by the proof of Theorem 1, $\mathcal{K} \models_{\mathcal{S}} L'\theta^u\Theta$. Hence, $\mathcal{K} \models_{\mathcal{S}} G\theta^u\Theta$. Therefore, $\mathcal{K} \models_{\mathcal{S}} G\theta^u\Theta$ since $L'\theta^u = L\theta^u$. ∎

Lemma 7 *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and $L \leftarrow G$ be a ground clause. If $\mathcal{K}_i \vdash_{\mathcal{S}} L \leftarrow G$, then $\mathcal{K}_i\colon G \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$ implies $\mathcal{K}_i\colon L \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$.*

Proof. We show the lemma by induction on the length n of a derivation tree of $L \leftarrow G$ from $\mathcal{K}_i$.

- Let $\mathcal{K}_i^0 \vdash L \leftarrow G$. By Lemma 5, $\mathcal{K}_i^0\colon G \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ implies $\mathcal{K}_i^0\colon L \longrightarrow\!\!\!\!\!\rightarrow \emptyset$.
- Let $\mathcal{K}_i^m \vdash L \leftarrow G$ $(m > 0)$.

  $n = 1$:

  1. If $L \leftarrow G$ is derived by the sorted substitution rule, then $L \leftarrow G$ has the form $p_s(t) \leftarrow \emptyset$ and by definition $\mathcal{K}_j^{m'} \vdash p_s(t)$ with $0 \le m' < m$ and $i \ne j$. By Theorem 4, $\mathcal{K}_j^{m'}\colon p_s(t) \longrightarrow\!\!\!\!\!\rightarrow \emptyset$. By the sorted resolution rule with rigidity, we can to derive $\mathcal{K}_i\colon p_s(t) \xrightarrow[R1^+]{\epsilon} \emptyset$.
  2. If $L \leftarrow G$ is derived by the unsorted substitution rule, then we have $L' \leftarrow G' \in \mathcal{K}_i$ and there exists an unsorted substitution θ^u such that $(L' \leftarrow G' \cup G_1 \cup \cdots \cup G_k)\theta^u = L \leftarrow G$. Let $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$ with $k \le n$. So, we have $\mathcal{K}_i^0 \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K}_i^0 \vdash p_{s_k}(t_k) \leftarrow G_k$, and $\mathcal{K}_{i_{k+1}}^{m_{k+1}} \vdash p_{s_{k+1}}(t_{k+1}), \ldots, \mathcal{K}_{i_n}^{m_n} \vdash p_{s_n}(t_n)$ where for $k+1 \le j \le n$, $m_j > 0$ and $i_j \ne i$. By induction hypothesis, $\mathcal{K}_i\colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K}_i\colon \{p_{s_k}(t_k)\} \longrightarrow G_n$, and by Theorem 4, $\mathcal{K}_{i_{k+1}}^{m^{k+1}}\colon p_{s_{k+1}}(t_{k+1}) \overset{\Theta_{k+1}}{\longrightarrow\!\!\!\!\!\rightarrow} \emptyset, \ldots, \mathcal{K}_{i_n}^{m^n}\colon p_{s_n}(t_n) \overset{\Theta_n}{\longrightarrow\!\!\!\!\!\rightarrow} \emptyset$. If $\mathcal{K}_i\colon G \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$, then $\mathcal{K}_i\colon L \xrightarrow[R5^+]{\theta^u} (G' \cup G_1 \cup \cdots \cup G_k)\theta^u \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ (by the unsorted substitution resolution rule with rigidity).
  3. If $p_\tau(t) \leftarrow G$ is derived by the unsorted type-predicate rule, where $G = G_1 \cup \cdots \cup G_k$, then there exists an unsorted substitution θ^u such that for some term t', $t'\theta^u = t$, and $sort(t') \le \tau$. Let $\iota(\theta^u) = \{x_1\colon s_1/t_1, \ldots, x_n\colon s_n/t_n\}$ with $k \le n$. So, we have $\mathcal{K}_i^0 \vdash p_{s_1}(t_1) \leftarrow G_1, \ldots, \mathcal{K}_i^0 \vdash p_{s_k}(t_k) \leftarrow G_k$, and $\mathcal{K}_{i_{k+1}}^{m_{k+1}} \vdash p_{s_{k+1}}(t_{k+1}), \ldots, \mathcal{K}_{i_n}^{m_n} \vdash p_{s_n}(t_n)$ where for $k+1 \le j \le n$, $m_j > 0$ and $i_j \ne i$. By induction hypothesis, $\mathcal{K}_i\colon \{p_{s_1}(t_1)\} \longrightarrow G_1, \ldots, \mathcal{K}_i\colon \{p_{s_k}(t_k)\} \longrightarrow G_n$, and by Theorem 4, $\mathcal{K}_{i_{k+1}}^{m^{k+1}}\colon p_{s_{k+1}}(t_{k+1}) \overset{\Theta_{k+1}}{\longrightarrow\!\!\!\!\!\rightarrow} \emptyset, \ldots, \mathcal{K}_{i_n}^{m^n}\colon p_{s_n}(t_n) \overset{\Theta_n}{\longrightarrow\!\!\!\!\!\rightarrow} \emptyset$. If $\mathcal{K}_i\colon G \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$, then $\mathcal{K}_i\colon p_\tau(t) \xrightarrow[R4^+]{\theta^u} (G_1 \cup \cdots \cup G_k)\theta^u \longrightarrow\!\!\!\!\!\rightarrow \emptyset$ (by the unsorted type-predicate resolution rule with rigidity).

  $n > 1$:

1. If $L \leftarrow G$ is derived by the cut rule, then $\mathcal{K}_i^m \vdash L' \leftarrow G''$ and $\mathcal{K}_i^m \vdash L \leftarrow G' \cup \{L'\}$. By induction hypothesis, there are subsets G' and G'' of G such that $G = G' \cup G''$, $\mathcal{K}_i^m \colon G' \cup \{L'\} \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$ implies $\mathcal{K}_i^m \colon L \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$, and $\mathcal{K}_i^m \colon G'' \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$ implies $\mathcal{K}_i^m \colon L' \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$. Hence $\mathcal{K}_i^m \colon G \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$ implies $\mathcal{K}_i^m \colon L \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$.
2. If $p_{s'}(t) \leftarrow G$ is derived by the subsort rule, then, $\mathcal{K}_i^m \vdash p_s(t) \leftarrow G$ and $s' \leq s$. By the induction hypothesis, if $\mathcal{K}_i^m \colon G \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$, then $\mathcal{K}_i^m \colon p_s(t) \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$. Therefore, $\mathcal{K}_i^m \colon p_s(t) \xrightarrow{\epsilon}_{R2} p_{s'}(t) \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$. ∎

Theorem 9 (Ground Completeness of Rigid-Property Resolution) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and G be a ground goal. If $\mathcal{K}_i \models_{\mathcal{S}} G$, then there exists a rigid-property resolution $\mathcal{K}_i \colon G \overset{\Theta}{\longrightarrow\!\!\!\!\!\rightarrow}_+ \emptyset$.*

Proof. Let $\mathcal{I}, w_{\mathcal{K}_1} \models \mathcal{K}_1$, …, $\mathcal{I}, w_{\mathcal{K}_n} \models \mathcal{K}_n$. By assumption, $\mathcal{I}, w_{\mathcal{K}_i} \models_{\mathcal{S}} G$. Then $\mathcal{I}_{\mathcal{S}}, w_{\mathcal{K}_i} \models G$ by Lemma 2. By Definition 16, $\mathcal{K}_i \vdash_{\mathcal{S}} L$ for all $L \in G$. Hence, $\mathcal{K}_i \colon L \longrightarrow\!\!\!\!\!\rightarrow_+ \emptyset$ by Lemma 7. Therefore, we can obtain $\mathcal{K}_i \colon G \overset{\Theta}{\longrightarrow\!\!\!\!\!\rightarrow}_+ \emptyset$. ∎

Lemma 8 (Lifting) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases. If $\mathcal{K}_i$ has an unrestricted rigid-property resolution*

$$\mathcal{K}_i \colon G_0\Theta_0 \xrightarrow{\Theta_1}_+ G_1 \xrightarrow{\Theta_2}_+ \cdots \xrightarrow{\Theta_n}_+ G_n,$$

then $\mathcal{K}_i$ has a rigid-property resolution

$$\mathcal{K}_i \colon G_0 \xrightarrow{\Theta'_1}_+ G'_1 \xrightarrow{\Theta'_2}_+ \cdots \xrightarrow{\Theta'_n}_+ G'_n$$

and the following holds.

(i) $\gamma_0 = \Theta_0$, and for $1 \leq i \leq n$, $(\gamma_{i-1}\!\uparrow Var(G_{i-1}))\Theta_i = \Theta'_i\gamma_i$ and $G_i = G'_i\gamma_i$, and

(ii) there exists a substitution γ'_n with $G_0\Theta_0\Theta_1\!\uparrow \cdots \Theta_n\!\uparrow = G_0\Theta'_1\!\uparrow \cdots \Theta'_n\!\uparrow\gamma'_n$.

Proof. Similar to Lemma 6. ∎

Using the above theorems and lemmas, the completeness of the rigid-property resolution system can be proven as follows.

Theorem 10 (Completeness of Rigid-Property Resolution) *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases, G be a goal, and Θ be a sorted or unsorted substitution. If $\mathcal{K}_i \models_{\mathcal{S}} G\Theta$, then there exists a successful rigid-property resolution $\mathcal{K}_i \colon G \overset{\Theta'}{\longrightarrow\!\!\!\!\!\rightarrow}_+ \emptyset$ such that $G\Theta = G\Theta'\!\uparrow\gamma$.*

Proof. Theorem 9 and Lemma 8 derive the completeness similar to Theorem 7. ∎

5.3 Query-Answering Algorithm

The purpose of this subsection is to develop a query-answering algorithm of the rigid-property resolution. We focus on function-free, recursive knowledge bases because the elimination of n-ary functions ($n \geq 1$) results in a decidable algorithm by means of the techniques for deductive databases [50, 11]. The linear resolution and rigid-property resolution systems do not contain any inference strategy since a detailed procedure on how to select inference rules and clauses has not been described.

```
1  Algorithm solve
2  input set of clauses K, set of atoms G, family of sets of clauses S, substitution Θ0
3  output set of substitutions (successful) or ⊥ (fails)
4  global variables Δ_K = ∅, Σ_K = ∅
5  begin
6     if G = ∅ then return {Θ0};
7     Sub = ∅;
8     L = select(G);
9     if L ∈ instance(Δ_K) then
10       for L' ∈ Σ_K do
11          θ = unify(L, L');
12          if θ ≠ ⊥ then
13             δ = solve(K, (G − {L})θ, S, Θ0 ∘ θ);
14             if δ ≠ ⊥ then Sub = Sub ∪ δ;
15          fi;
16       rof;
17    else Δ_K = Δ_K ∪ {L};
18       for C ∈ K and r ∈ {R1, ..., R5, R1+, R4+, R5+} do
19          (G_a, G_b, Θ) = apply_rule(L, C, G, r);
20          if  Θ ≠ ⊥ then
21             δ_a = solve(K, G_a, S, Θ0 ∘ Θ);
22             if δ_a ≠ ⊥ then
23                Sub = Sub ∪ δ_a;
24                for Θ' ∈ δ_a do
25                   if LΘ' ∉ instance(Σ_K) then rigid Σ_K = Σ_K ∪ {LΘ'};
26                   δ_b = solve(K, G_bΘ', S, Θ0 ∘ Θ ∘ Θ');
27                   if δ_b ≠ ⊥ then Sub = Sub ∪ δ_b;
28                rof;
29             fi;
30          fi;
31       rof;
32    else;
33    if Sub ≠ ∅ then return Sub else return ⊥;
34 end;
```

In the query-answering algorithm, the subroutine *select*(G) as a selection function selects the leftmost atom in the goal G. The function *instance*$(\Delta_{\mathcal{K}})$ denotes the set of sorted

instances of clauses in $\Delta_{\mathcal{K}}$. Moreover, the subroutine $unify(L, L')$ returns the most general unifier for L and L' if they are unifiable, otherwise, it returns $\perp$. The unification can be implemented by the order-sorted unification algorithm in [27]. When we apply a resolution rule to a selected atom in a goal, the following subroutine $apply\_rule(L, C, G, r)$ is called, where L is an atom, C is a clause, G is a goal, and r is a resolution rule.

Algorithm *apply_rule*
input atom L, clause C, goal G, resolution rule r
output set of atoms G_a, set of atoms G_b, and
        set of substitutions (successful) or $\perp$ (fails)
begin
  if r is applicable to the pair (L, C) or the atom L **then**
    switch(r)
      case $R1$ or $R2$:
        $G_a = (G - \{L\})\theta$; $G_b = G'\theta$; $\Theta = \theta$;
        (by applying the sorted or subsort resolution rule)
      case $R3$:
        $G_a = (G - \{L\})\theta$; $G_b = \emptyset$; $\Theta = \theta$;
        (by applying the type-predicate resolution rule)
      case $R4$:
        $G_a = (G - \{L\})\theta^u$; $G_b = (G_1 \cup \cdots \cup G_n)\theta^u$; $\Theta = \theta^u$;
        (by applying the unsorted type-predicate resolution rule)
      case $R5$:
        $G_a = (G - \{L\})\theta^u$; $G_b = G'\theta^u \cup (G_1 \cup \cdots \cup G_n)\theta^u$; $\Theta = \theta^u$;
        (by applying of the unsorted resolution rule)
      case $R1^+$:
        $G_a = (G - \{L\})\theta^u$; $G_b = \emptyset$; $\Theta = \theta^u$;
        (by applying the sorted resolution rule with rigidity)
        $\delta_{\mathcal{K}'} = solve(\mathcal{K}', G'\theta, \mathcal{S}, \emptyset)$;
        if $\delta_{\mathcal{K}'} = \perp$ **return** $(G_a, G_b, \perp)$;
      case $R4^+$:
        $G_a = (G - \{L\})\theta^u$; $G_b = (G_1 \cup \cdots \cup G_m)\theta^u$; $\Theta = \theta^u$;
        (by applying the unsorted resolution rule with rigidity)
        for $\mathcal{K}' \in \{\mathcal{K}'_{m+1}, \ldots, \mathcal{K}'_n\}$ **do**
          $\delta_{\mathcal{K}'} = solve(\mathcal{K}', G'\theta, \mathcal{S}, \emptyset)$;
          if $\delta_{\mathcal{K}'} = \perp$ **return** $(G_a, G_b, \perp)$;
        rof;
      case $R5^+$:
        $G_a = (G - \{L\})\theta^u$; $G_b = G'\theta^u \cup (G_1 \cup \cdots \cup G_m)\theta^u$; $\Theta = \theta^u$;
        (by applying the unsorted resolution rule with rigidity)
        for $\mathcal{K}' \in \{\mathcal{K}'_{m+1}, \ldots, \mathcal{K}'_n\}$ **do**
          $\delta_{\mathcal{K}'} = solve(\mathcal{K}', G'\theta, \mathcal{S}, \emptyset)$;
          if $\delta_{\mathcal{K}'} = \perp$ **return** $(G_a, G_b, \perp)$;
        rof;
    return (G_a, G_b, Θ);
  fi;
  return $(\emptyset, \emptyset, \perp)$;

42 **end**;

We consider a search strategy of the rigid-property resolution, i.e., a selection function is fixed in a resolution of a goal G_0:

$$\mathcal{K}\colon G_0 \xrightarrow{\Theta_1} G_1 \xrightarrow{\Theta_2} G_2 \xrightarrow{\Theta_3} \cdots \xrightarrow{\Theta_n} G_n$$

where one of the atoms in each goal G_i is selected when a resolution rule is applied to the goal. We denote a sequence $\mathcal{K}_1\colon G_1, \ldots, \mathcal{K}_k\colon G_k$ $(k \geq 0)$ of pairs of knowledge bases and goals by *Seq* and denote a sequence $\mathcal{K}_1\colon \emptyset, \ldots, \mathcal{K}_l\colon \emptyset$ $(l \geq 0)$ of pairs of knowledge bases and the empty set by *ESeq*.

We define an SLD-tree that is a derivation tree of SLD-resolution (selection-rule driven linear resolution for definite clauses). If a selection function is added to a linear resolution system, then we obtain an SLD-resolution system. As one of the techniques of automated reasoning, the truth of a goal in a set of clauses can be checked to terminate under SLD-resolution. We make use of this technique in order to develop a decidable goal-oriented reasoning algorithm for our proposed order-sorted logic (logic programming with tabling [49] is an alternative decidable technique).

Let $\mathcal{S}$ be a finite set of knowledge bases, $\mathcal{K}$ be a knowledge base in $\mathcal{S}$, and G_0 be a goal. A leftmost rigid-property SLD-tree for G_0 is a tree such that

(i) the root is labeled with $(\mathcal{K}\colon G_0)$;

(ii) if a node d is labeled with $(\mathit{ESeq}, \mathcal{K}\colon G, \mathit{Seq})$ and a resolution rule (either of $R1$ to $R5$) is applicable to G in $\mathcal{K}$ (i.e., $\mathcal{K}\colon G \xrightarrow{\theta}_{R*} G'$ for some $* \in \{1, \ldots, 5\}$), the node d has a child node d' labeled with $(\mathit{ESeq}, \mathcal{K}\colon G', \mathit{Seq})$ and the edge (d, d') is labeled with θ;

(iii) if a node d is labeled with $(\mathit{ESeq}, \mathcal{K}\colon G, \mathit{Seq})$ and the sorted resolution rule with rigidity $(R1^+)$ is applicable to G in $\mathcal{K}$ (i.e., $\mathcal{K}\colon G \xrightarrow{\theta}_{R1^+} G'$ and $\mathcal{K}'\colon G''$), the node d has a child node d' labeled with $(\mathit{ESeq}, \mathcal{K}'\colon G'', \mathcal{K}\colon G', \mathit{Seq})$ and the edge (d, d') is labeled with θ;

(iv) if a node d is labeled with $(\mathit{ESeq}, \mathcal{K}\colon G, \mathit{Seq})$ and the unsorted type-predicate resolution rule with rigidity or unsorted resolution rule with rigidity ($R4^+$ or $R5^+$) is applicable to G in $\mathcal{K}$ (i.e., $\mathcal{K}\colon G \xrightarrow{\theta^u}_{R*^+} G'$ for some $* \in \{4, 5\}$ and $\mathcal{K}'_1\colon G'_1, \ldots, \mathcal{K}'_m\colon G'_m$), the node d has a child node d' labeled with $(\mathit{ESeq}, \mathcal{K}'_1\colon G'_1, \ldots, \mathcal{K}'_m\colon G'_m, \mathcal{K}\colon G', \mathit{Seq})$ and the edge (d, d') is labeled with θ.

A SLD-tree contains a success if there exists a leaf node labeled with $\mathit{ESeq} = \mathcal{K}_1\colon \emptyset, \ldots, \mathcal{K}_l\colon \emptyset$ in it.

Lemma 9 *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases and G_0 be a goal. Every leftmost rigid-property SLD-tree contains a success with a computed answer substitution Θ if and only if there exists a substitution Θ in $\{\Theta_1, \ldots, \Theta_m\}$ such that $\mathit{solve}(\mathcal{K}_i, G_0, \mathcal{S}, \emptyset) = \{\Theta_1, \ldots, \Theta_m\}$.*

Proof. ($\Leftarrow$) Let us assume $\mathit{solve}(\mathcal{K}_i, G_0, \mathcal{S}, \emptyset) = \{\Theta_1, \ldots, \Theta_m\}$. In the process of $\mathit{solve}(\mathcal{K}_i, G_0, \mathcal{S}, \emptyset)$, we can construct a tree for G in the following operations:

- Create the root d_0 labeled with $(\mathcal{K}_i\colon G_0)$.
- Create a new node d' and the edge (d, d') under conditions (ii) - (iv) of leftmost rigid-property SLD trees if a node d labeled with $(ESeq, \mathcal{K}\colon G, Seq)$ is created and $apply\_rule(L, C, G, r)$ is called where $L = select(G)$, $C \in \mathcal{K}$, and $r \in \{R1, \ldots, R5, R1^+, R4^+, R5^+\}$.
- Add the proof segment of L' to the node if a node d labeled with $(ESeq, \mathcal{K}\colon G, Seq)$ is created, $L = select(G)$, $L \in instance(\Delta_\mathcal{K})$, $L' \in \Sigma_\mathcal{K}$, $\theta = unify(L, L')$, and $\theta \neq \bot$.

The trees constructed by these operations satisfy the conditions of leftmost rigid-property SLD trees, where for each substitution $\Theta \in \{\Theta_1, \ldots, \Theta_m\}$, there exists a leaf node of the form $ESeq$ with the computed answer substitution Θ.

($\Rightarrow$) Let $\mathfrak{T}$ be a leftmost rigid-property SLD-tree for G_0 that contains a success with a computed answer substitution Θ. By definition, $\mathfrak{T}$ has a leaf node labeled with $ESeq = \mathcal{K}'_1\colon \emptyset, \ldots, \mathcal{K}'_m\colon \emptyset$ where $\{\mathcal{K}'_1, \ldots, \mathcal{K}'_m\} \subseteq \mathcal{S}$. This implies that we have a path from $(\mathcal{K}_i\colon G_0)$ to $ESeq$ as follows:

$$(\mathcal{K}_i\colon G_0) \rightarrow (ESeq_1, Seq_1) \rightarrow \cdots \rightarrow (ESeq_k, Seq_k) \rightarrow (ESeq)$$

This sequence generates a successful rigid-property resolution of G_0. By the definition of leftmost rigid-property SLD-trees, the leftmost atom of each goal is selected when a resolution rule is applied to a node, and the goals $G''_1, \ldots, G''_r$ in other knowledge bases $\mathcal{K}''_1, \ldots, \mathcal{K}''_r$ are added in front of the goal $\mathcal{K}\colon G'$ containing the selected atom (i.e., $(ESeq, \mathcal{K}''_1\colon G''_1, \ldots, \mathcal{K}''_r\colon G''_r, \mathcal{K}\colon G', Seq)$) after one of the rigid-property resolution rules is applied to a node.

The applications of resolution rules are performed in the query answering algorithm $solve(\mathcal{K}_i, G, \mathcal{S}, \emptyset)$. Note that if $L \in instance(\Delta_\mathcal{K})$ where L is selected by $L = select(G)$, then the resolution process of the atom L is skipped. It indicates that the atom L has been already resolved successfully in the former processes and there is no need to repeat the same resolution steps because resolution rules are identically applied. Therefore, the algorithm derives the empty goal $\emptyset$ for the knowledge bases $\mathcal{K}'_1, \ldots, \mathcal{K}'_m$ where $\{\mathcal{K}'_1, \ldots, \mathcal{K}'_m\} \subseteq \mathcal{S}$. It returns a set of substitutions $\{\Theta_1, \ldots, \Theta_m\}$ that contains the computed answer substitution Θ in the successful rigid-property resolution of G_0. ∎

The completeness of the algorithm *solve* is given as follows.

Theorem 11 (Completeness of *solve***)** *Let $\mathcal{S} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$ be a finite set of knowledge bases, G be a goal, and Θ be a sorted or unsorted substitution. If $\mathcal{K}_i \models_\mathcal{S} G\Theta$, then there exists a substitution Θ in $\{\Theta_1, \ldots, \Theta_m\}$ such that $G\Theta = G\Theta'^{\uparrow}\gamma$ and $solve(\mathcal{K}_i, G, \mathcal{S}, \emptyset) = \{\Theta_1, \ldots, \Theta_m\}$.*

Proof. By Theorem 10, we have a successful rigid-property resolution $\mathcal{K}_i\colon G \xrightarrow{\Theta'}{}_+ \emptyset$ such that $G\Theta = G\Theta'^{\uparrow}\gamma$. Hence, we can construct a leftmost rigid-property SLD-tree containing a success. Therefore, by Lemma 9, this theorem can be proven. ∎

The decidability of the algorithm *solve* is guaranteed if the inputted knowledge bases are function-free as follows.

Theorem 12 (Termination of *solve***)** *The query-answering algorithm solve terminates.*

Proof. Each selected atom L in line 8 of the algorithm is added to $\Delta_{\mathcal{K}}$ if L is not an instance of each atom in $\Delta_{\mathcal{K}}$. So, the number of recursive calls depends on the number of elements of $\Delta_{\mathcal{K}}$ because lines 10 - 16 are the only routine to decrease the elements of next goal each atom of which recursively calls the algorithm. If every knowledge base contains no functions, the set $\Delta_{\mathcal{K}}$ is limited to finite. Therefore, it terminates. ∎

6 Discussion

Prominent languages such as RDF [33], OWL [41], and RuleML [1] have been designed for modeling ontologies and rules for the Semantic Web; sound and complete decidable reasoning services for these languages are provided by Description Logics [5] and DATALOG. Moreover, SWRL [26], a combination of OWL and RuleML, corresponds to a combination of Description Logics and DATALOG. However, SWRL becomes undecidable even when the rules are function-free [25]. In order to make it decidable, the rules must be non-recursive [35, 15], DL-safe [24, 42], or Description Logic Programs (DLP) [19]; however, these restrictions result in the loss of expressive power of rules. Recently, as discussed in [32], the new OWL 2 [2] has provided us the tractable language profile OWL 2 EL with DL-rules, which are based on the Description Logic $\mathcal{EL}^{++}$. The rule-based language ELP [32], which extends $\mathcal{EL}^{++}$ using local reflexivity, concept products, and the conjunctions of simple roles, preserves tractability. These sophisticated proposals enable low complexities but restrict the expressive power of rules.

A solution to the problem of undecidability or limited rule expressions is the use of order-sorted logic programming, which is a combination of sort hierarchy and logic programming [36, 4]. Sort hierarchy provides a simple ontology of partially ordered sorts that does not include negation, disjunction, or quantification. Our approach is an extension of order-sorted logic programming that preserves decidability in function-free, recursive rules. In addition to decidability, we present effective reasoning services in an order-sorted linear resolution system. In previous studies, researchers had proposed many refinements to inference systems for automated reasoning. For example, hyperresolution reduces a sequence of resolution steps into a single inference step [34]. Hyperresolution is an efficient method for reducing general clauses since it helps resolve more than two clauses in a single step. We use linear resolution as a refinement because it is useful for combining Horn clauses and goal-oriented reasoning (i.e., only inference steps that are related to a goal are generated).

In the Semantic Web context, multiple software agents access distributed ontologies and rules on the Web. In multi-agent reasoning [16], multiple knowledge bases can be assigned to agents. In other words, each agent should have its own knowledge base because it updates knowledge through its experience by communicating with other agents. Researchers have previously modified the OWL global semantics and proposed C-OWL (Context OWL [8]) for distributed OWL ontologies. In C-OWL, the semantics of a set of distributed ontologies $\langle i, O_i \rangle$ is defined in local domains and valuations; context mappings among different ontologies are represented by a set of bridge rules.

In comparison, our order-sorted logic programming handles distributed facts and rules with a common ontology. In particular, logic programming with a sort hierarchy can be used to propagate rigid properties from multiple knowledge bases. Each knowledge base indirectly and safely accesses facts and rules available in other knowledge bases. Rigid-property derivation makes use of the fact that rigid-property assertions are unconditionally

true for any other agent since they are true even if the situation or time changes. This derivation is computationally improved by developing a rigid-property resolution system as a top-down algorithm. The top-down algorithm is important for rigid-property derivation in multiple knowledge bases. In the bottom-up algorithm, mutualizing all the rigid-property assertions would lead to a result identical to that obtained when each rigid property is transferred from other knowledge bases; however, mutualization is ineffective because the dynamically extended set of rigid assertions becomes very large (unlike ontologies that are more easily shared than the assertions). The top-down linear resolution system is effective for multiple knowledge bases because of the goal-oriented reasoning method. In other words, global accesses are limited to some rigid properties in a limited number of other knowledge bases, as shown in Example 5.

In addition to the proposed reasoning system, the axiomatic approach, which involves axioms of ontological considerations, can be used to realize the abovementioned rigid property reasoning. Both approaches are closely related; however, the axiomatic approach may lead to ineffective bottom-up reasoning by instantiating the general axioms because of its inability to control the reasoning strategy unlike the linear resolution system.

7 Conclusion and Future Work

In this paper, we described an effective framework for order-sorted logic programming for multiple knowledge bases. In our study, the order-sorted language was extended to contain three types of property expressions (types, non-rigid sorts, and unary predicates) so that sorted terms and formulas adhere to the rigidity of properties. We also developed a query-answering algorithm equipped with rigid-property resolution that is decidable and effective for function-free, recursive knowledge bases. The algorithm provides new reasoning services for multiple knowledge bases so that each knowledge base can extract rigid properties from other knowledge bases. The feasibility of these reasoning services is guaranteed by the fact that the truth of rigid-property assertions is independent of the situation in which each knowledge base is built. From a technical perspective, after developing the bottom-up algorithms (the order-sorted Horn-clause calculus and the rigid-property derivation system), the decidable and effective query-answering algorithm is obtained from the top-down algorithms (the linear resolution and rigid-property resolution systems).

In future studies, we intend to further refine the rigid-property derivation to establish structural relationships among multiple knowledge bases. We plan to extend distributed reasoning services to facts and rules by exploiting inclusion, equation, and disjoint relations among contexts, similar to the context mapping in C-OWL. Furthermore, we can effectively obtain reliable results if knowledge bases are authorized to be trustworthy.

Acknowledgment

This research has been partially supported by the Japanese Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (20700147).

References

[1] http://www.ruleml.org/.

[2] http://www.w3.org/tr/owl2-profiles/.

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

[4] K. R. Apt. *From Logic Programming to Prolog.* Prentice-Hall, 1997.

[5] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge (2nd Edition).* 2007.

[6] C. Beierle, U. Hedtsück, U. Pletat, P.H. Schmitt, and J. Siekmann. An order-sorted logic for knowledge representation systems. *Artificial Intelligence*, 55:149–191, 1992.

[7] B. Bennett and C. Fellbaum, editors. *International Conference on Formal Ontology in Information Systems (FOIS), Baltimore, Maryland, USA.* IOS Press, 2006.

[8] P. Bouquet, F. Giunchiglia, F. Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing Ontologies. In *Proc. of the 2nd International Semantic Web Conference (ISWC 2003)*, pages 164–179, 2003.

[9] H-J. Bürckert. A resolution principle for constraint logics. *Artificial Intelligence*, 66:235–271, 1994.

[10] M. Carrara and P. Giaretta. Identity criteria and sortal concepts. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 234–243. ACM Press, 2001.

[11] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases.* Springer-Verlag, Berlin, 1990.

[12] A. G. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning*, 3:113–200, 1987.

[13] A. G. Cohn. Taxonomic reasoning with many sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

[14] K. Doets. *From Logic to Logic Programming.* The MIT Press, 1994.

[15] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-LOG: Integrating datalog and description logic. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[16] R. Fagin, J. Y. Halpern, V. Moses, and M. Y. Vardi. *Reasoning About Knowledge.* MIT Press, 1995.

[17] A. M. Frisch. A general framework for sorted deduction: Fundamental results on hybrid reasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 1989.

[18] A. Gangemi, N. Guarino, and A. Oltramari. Conceptual analysis of lexical taxonomies: the case of wordnet top-level. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 285–296. ACM Press, 2001.

[19] B. Grosof, I. Horrocks, R. Volz, and S. Decker. Description Logic Programs: Combining Logic Programs with Description Logics. In *Proc. of WWW-2003*, Budapest, Hungary, 05 2003.

[20] N. Guarino, M. Carrara, and P. Giaretta. An ontology of meta-level categories. In *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning*, pages 270–280, 1994.

[21] N. Guarino and C. Welty. A formal ontology of properties. In *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management*, pages 97–112, 2000.

[22] N. Guarino and C. Welty. Ontological analysis of taxonomic relationships. In *Proceedings of ER-2000: The Conference on Conceptual Modeling*, 2000.

[23] M. Hanus. Logic programming with type specifications. In F. Pfenning, editor, *Types in Logic Programming*. The MIT Press, 1992.

[24] P. Hitzler and B. Parsia. Ontologies and rules. In S. Staab and R. Studer, editors, *Handbook on Ontologies (2nd Edition)*. (To appear).

[25] I. Horrocks and P. F. Patel-Schneider. A proposal for an owl rules language. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)*, pages 723–731. ACM, 2004.

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

[27] K. Kaneiwa. *An Order-Sorted Logic with Predicate Hierarchy, Eventuality, and Implicit Negation*. PhD thesis, Japan Advanced Institute of Science and Technology, 2001.

[28] K. Kaneiwa. The completeness of logic programming with sort predicates. *Systems and Computers in Japan*, 35(1):37–46, 2004.

[29] K. Kaneiwa and R. Mizoguchi. Ontological knowledge base reasoning with sort-hierarchy and rigidity. In *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR2004)*, pages 278–288, 2004.

[30] A. N. Kaplan. Towards a consistent logical framework for ontological analysis. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 244–255. ACM Press, 2001.

[31] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.

[32] M. Krötzsch, S. Rudolph, and P. Hitzler. ELP: Tractable rules for OWL 2. In *Proceedings of the 7th International Semantic Web Conference (IWSC 2008)*, LNCS 5318, pages 649–664, 2008.

[33] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax specification, 1998.

[34] A. Leitsch. *The Resolution Calculus.* EATCS Texts in Theoretical Computer Science. Springer, 1997.

[35] A. A. Levy and M-C. Rousset. CARIN: A representation language combining Horn rules and description logics. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[36] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[37] E. J. Lowe. *Kinds of Being. A Study of Individuation, Identity and the Logic of Sortal Terms.* Basil Blackwell, Oxford., 1989.

[38] M. Manzano. Introduction to many-sorted logic. In *Many-sorted Logic and its Applications*, pages 3–86. John Wiley and Sons, 1993.

[39] A. Oberschelp. Untersuchungen zur mehrsortigen quantorelogik. *Mathematische Annalen 145*, pages 297–333, 1962.

[40] A. Oberschelp. Order sorted predicate logic. In *Workshop on Sorts and Types in Artificial Intelligence*, 1989.

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

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

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

[44] B. Smith. Basic concepts of formal ontology. In *Formal Ontology in Information Systems.* 1998.

[45] B. Smith, W. Ceusters, B. Klagges, J. Kohler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome Biol*, 6(5):R46, 2005.

[46] G. Smolka. *Logic Programming over Polymorphically Order-Sorted Types.* PhD thesis, Universitat Kaiserslautern., 1989.

[47] R. Socher-Ambrosius and P. Johann. *Deduction Systems.* Springer-Verlag, 1996.

[48] P. F. Strawson. *Individuals: An Essay in Descriptive Metaphysics.* Methuen, London, 1959.

[49] S. Verbaeten, D. De Schreye, and K. Sagonas. Termination proofs for logic programs with tabling.

[50] L. Vieille. Recursive query processing: The power of logic. *Theoretical Computer Science*, 69(1):1–53, 1989.

[51] C. Walther. A mechanical solution of Schubert's steamroller by many-sorted resolution. *Artificial Intelligence*, 26(2):217–224, 1985.

[52] C. Walther. *A Many-Sorted Calculus Based on Resolution and Paramodulation*. Pitman and Kaufman Publishers, 1987.

[53] C. Walther. Many-sorted unification. *Journal of the Association for Computing Machinery*, 35:1, 1988.

[54] T. Weibel. An order-sorted resolution in theory and practice. *Theoretical Computer Science*, 185(2):393–410, 1997.

[55] C. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39(1):51–74, 2001.

Appendix

A. An Example of Reasoning

Let us consider rigid-property derivation in the knowledge bases $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$ of Example 4.

Example 6 *Suppose that we have the sorted signature $\Sigma = (T, N, \Omega, \leq^*)$ of Example 2 and knowledge bases $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$ of Example 4. By using the expanded knowledge bases $\mathcal{K}_i^0$, $\mathcal{K}_i^1, \ldots$ of each $\mathcal{K}_i$ in $\mathcal{S} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4\}$, the following derivations can be obtained. At the beginning, for $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$, we have the following theories:*

$$
\begin{aligned}
Th(\mathcal{K}_1) &= \Gamma \cup \{\ excellent(john\colon person),\\
&\qquad obtaining\_a\_discount(john\colon person),\\
&\qquad male\_customer(john\colon person),\\
&\qquad customer(john\colon person),\\
&\qquad male(john\colon person)\ \}\\
Th(\mathcal{K}_2) &= \Gamma \cup \{\ pet\_seller(mary\colon person)\ \}\\
Th(\mathcal{K}_3) &= \Gamma \cup \{\ canfly(peter\colon animal),\\
&\qquad bird(peter\colon animal),\\
&\qquad canary(peter\colon animal)\ \}\\
Th(\mathcal{K}_4) &= \Gamma \cup\\
&\qquad \{\ father(tony\colon animal, peter\colon animal)\ \}
\end{aligned}
$$

where $\Gamma = \{$ *person*$(john\colon person)$, *animal*$(john\colon person)$, *person*$(mary\colon person)$, *animal* $(mary\colon person)$, *animal*$(peter\colon animal)$, *animal*$(tony\colon animal)\}$.

It should be noted that theories $Th(\mathcal{K}_1)$ and $Th(\mathcal{K}_3)$ contain the rigid atomic formulas $male(john\colon person)$ and $bird(peter\colon animal)$, $canary(peter\colon animal)$ respectively. In the following steps, $\mathcal{K}_1^0$, $\mathcal{K}_2^0$, $\mathcal{K}_3^0$, and $\mathcal{K}_4^0$ are expanded by adding these rigid atomic formulas:

Step 1: *We set the initial values of $\mathcal{K}_1^0$, $\mathcal{K}_2^0$, $\mathcal{K}_3^0$, and $\mathcal{K}_4^0$.*

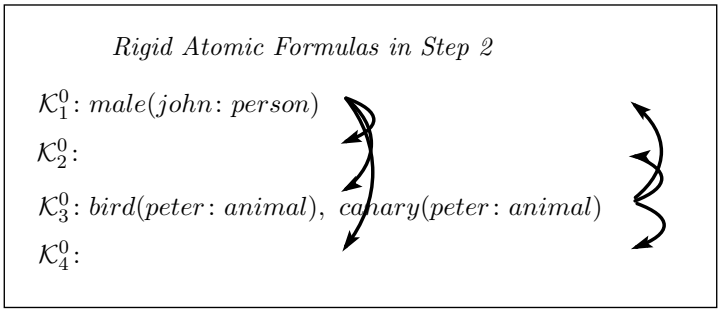
$\mathcal{K}_i^0 = \mathcal{K}_i$ $(1 \leq i \leq 4)$

Step 2: *Each $\mathcal{K}_i^0$ is expanded to $\mathcal{K}_i^1$.*

$\mathcal{K}_i^1 = \mathcal{K}_i^0 \cup \Delta(T^0)$ $(1 \leq i \leq 4)$ *where*

$T^0 = Th(\mathcal{K}_1^0) \cup Th(\mathcal{K}_2^0) \cup Th(\mathcal{K}_3^0) \cup Th(\mathcal{K}_4^0)$ *and*

$\Delta(T^0) = \Gamma \cup \{\ male(john\colon person),\ bird(peter\colon animal),\ canary(peter\colon animal)\ \}$



Step 3: *As a result of the expansion, the new conclusions* $cares\_about(mary\colon person, peter\colon animal)$ *in* $Th(\mathcal{K}_2^1)$ *and* $bird(tony\colon animal)$ *in* $Th(\mathcal{K}_4^1)$ *can be derived in the Horn-clause calculus.*

$Th(\mathcal{K}_1^1) = Th(\mathcal{K}_1^0) \cup \Delta(T^0)$

$Th(\mathcal{K}_2^1) = Th(\mathcal{K}_2^0) \cup \Delta(T^0) \cup \{\ cares\_about(mary\colon person, peter\colon animal)\ \}$

$Th(\mathcal{K}_3^1) = Th(\mathcal{K}_3^0) \cup \Delta(T^0)$

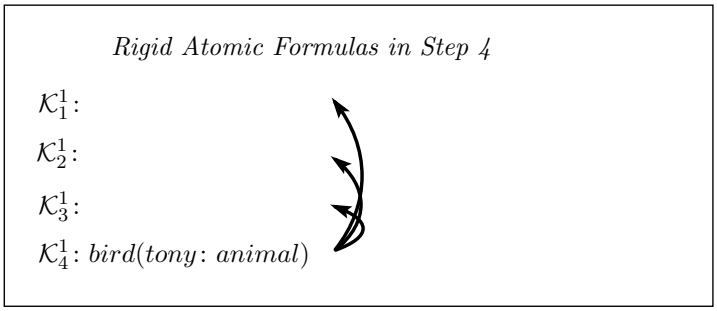
$Th(\mathcal{K}_4^1) = Th(\mathcal{K}_4^0) \cup \Delta(T^0) \cup \{\ bird(tony\colon animal)\ \}$

Step 4: *The rigid atomic formula* $bird(tony\colon animal)$ *expands* $\mathcal{K}_1^1$, $\mathcal{K}_2^1$, $\mathcal{K}_3^1$, *and* $\mathcal{K}_4^1$.

$\mathcal{K}_i^2 = \mathcal{K}_i^1 \cup \Delta(T^1)$ $(1 \leq i \leq 4)$ *where*

$T^1 = Th(\mathcal{K}_1^1) \cup Th(\mathcal{K}_2^1) \cup Th(\mathcal{K}_3^1) \cup Th(\mathcal{K}_4^1)$ *and*

$\Delta(T^1) = \Delta(T^0) \cup \{\ bird(tony\colon animal)\ \}$



Step 5: *We can obtain additional results* $cares\_about(mary\colon person,\ tony\colon animal)$ *in* $Th(\mathcal{K}_2^2)$ *and* $canfly(tony\colon animal)$ *in* $Th(\mathcal{K}_3^2)$ *that are generated from the expanded knowledge bases* $\mathcal{K}_1^2$, $\mathcal{K}_2^2$, $\mathcal{K}_3^2$, *and* $\mathcal{K}_4^2$.

$Th(\mathcal{K}_1^2) = Th(\mathcal{K}_1^1) \cup \Delta(T^1)$

$Th(\mathcal{K}_2^2) = Th(\mathcal{K}_2^1) \cup \Delta(T^1) \cup \{\ cares\_about(mary\colon person, tony\colon animal)\ \}$

$Th(\mathcal{K}_3^2) = Th(\mathcal{K}_3^1) \cup \Delta(T^1) \cup \{\ canfly(tony\colon animal)\ \}$

$Th(\mathcal{K}_4^2) = Th(\mathcal{K}_4^1) \cup \Delta(T^1)$

Step 6: *The derivation terminates because* $\mathcal{K}_1^2$, $\mathcal{K}_2^2$, $\mathcal{K}_3^2$, *and* $\mathcal{K}_4^2$ *can no longer be expanded.*

$\mathcal{K}_i^3 = \mathcal{K}_i^2 \cup \Delta(T^2)$ $(1 \leq i \leq 4)$ *where*

$T^2 = Th(\mathcal{K}_1^2) \cup Th(\mathcal{K}_2^2) \cup Th(\mathcal{K}_3^2) \cup Th(\mathcal{K}_4^2)$ *and*

$\Delta(T^2) = \Delta(T^1)$.

In this example, the following conclusion holds.

$$\begin{array}{lll} \mathcal{K}_1 & \vdash_{\mathcal{S}} & \mathit{customer}(\mathit{john}\colon \mathit{person}) \\ \mathcal{K}_2 & \nvdash_{\mathcal{S}} & \mathit{cares\_about}(\mathit{mary}\colon \mathit{person}, \mathit{john}\colon \mathit{person}) \end{array}$$

However, $\mathcal{K}_2$ cannot extract the fact $\mathit{customer}(\mathit{john}\colon \mathit{person})$ from $\mathcal{K}_1$ because the sort *customer* is not rigid. This means that we cannot determine whether John is a customer with respect to $\mathcal{K}_2$.

$$\begin{array}{rlll} & \mathcal{K}_2 & \vdash_{\mathcal{S}} & \mathit{cares\_about}(\mathit{mary}\colon \mathit{person}, \mathit{peter}\colon \mathit{animal}) \\ (\text{but } & \mathcal{K}_2 & \nvdash & \mathit{cares\_about}(\mathit{mary}\colon \mathit{person}, \mathit{peter}\colon \mathit{animal})) \\ & \mathcal{K}_2 & \vdash_{\mathcal{S}} & \mathit{cares\_about}(\mathit{mary}\colon \mathit{person}, \mathit{tony}\colon \mathit{animal}) \\ (\text{but } & \mathcal{K}_2 & \nvdash & \mathit{cares\_about}(\mathit{mary}\colon \mathit{person}, \mathit{tony}\colon \mathit{animal})) \\ & \mathcal{K}_3 & \vdash_{\mathcal{S}} & \mathit{canfly}(\mathit{tony}\colon \mathit{animal}) \\ (\text{but } & \mathcal{K}_3 & \nvdash & \mathit{canfly}(\mathit{tony}\colon \mathit{animal})) \end{array}$$

These were not derivable in the Horn-clause calculus without rigid property derivation (as denoted by $\nvdash$). However, by using our method, they can be derived from the knowledge bases $\mathcal{K}_2^2$ and $\mathcal{K}_3^2$ expanded by means of extracting the rigid-property information that Tony and Peter are birds:

$$\begin{array}{lll} \mathcal{K}_4 & \vdash_{\mathcal{S}} & \mathit{bird}(\mathit{tony}\colon \mathit{animal}) \\ \mathcal{K}_3 & \vdash_{\mathcal{S}} & \mathit{bird}(\mathit{peter}\colon \mathit{animal}) \end{array}$$