Resolution for Label-based Formulas in Hierarchical Representation

Ken KANEIWA

National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 JAPAN

kaneiwa@nii.ac.jp

Abstract Order-sorted logic includes many and partially ordered sorts as a sort-hierarchy. In the field of knowledge representation and reasoning, it is useful to develop reasoning systems for terminological knowledge, together with assertional knowledge. However, the expression of sort-hierarchies cannot sufficiently capture the lexical diversity of terminological knowledge. In addition to sorts, various kinds of symbols: constants, functions and predicates are semantically and hierarchically associated with each other. This is because natural language words identifying these symbols can be employed in the description of terminological knowledge. In this paper, we present a label-based language for consistently handling the variety of hierarchical relationships among symbol names. For this language we develop a sorted resolution system whose reasoning power is enhanced by adding hierarchical inference rules with labeled substitutions.

Keywords Terminological Knowledge, Order-sorted Logic, Resolution System, Knowledge Representation, Label-based Expressions.

§1 Introduction

Logical languages with class-hierarchy formally concentrate upon representation and reasoning for terminological knowledge[3] (and ontologies[7]), together with assertional knowledge. The significant feature is that one can specify classes as sets of individuals and their hierarchical relationships (e.g. *is-a* relations) in the languages. In the past, there have been several approaches to

the formalisms: order-sorted logic,[15, 16, 5, 17] typed logic programming,[1, 2, 9] description logics[8, 13] and object-oriented deduction languages.[12, 18] In particular, order-sorted logic gives us the advantage of interacting terminological knowledge and assertional knowledge. This logic theoretically corresponds to a first-order predicate logic with sort-hierarchy, where sort symbols are many and partially ordered. The sorted language can represent terminological knowledge as a sort-hierarchy and assertional knowledge as sorted formulas.

However, the expression of sort-hierarchies cannot sufficiently capture the lexical diversity of terminological knowledge since hierarchical information exclusive of sorts is ignored in logical languages. The elements of terminological knowledge can be regarded as various kinds of symbols in logic: sorts, constants, functions and predicates, whereas the symbols are semantically separated from each other and merely used as components in formulas. From the viewpoint of symbolic knowledge representation, we require that the symbol names be semantically and hierarchically associated with each other, without losing their roles. To enrich hierarchical reasoning, Kaneiwa and Tojo[10] proposed an order-sorted logic with sort and predicate hierarchies, but the hierarchies did not enable complicated expressions combining different kinds of symbols (e.g. $\mathit{father} \prec \mathit{parent}$ as a subordinate relation of a function and a predicate).

The aim of this paper is to provide a logical framework for consistently handling various hierarchies of symbol names (of sorts, constants, functions and predicates) and reasoning over them. We generalize an order-sorted logic by means of consisting of label-based expressions. A label-based language contains label-based terms and formulas and a label hierarchy, for representing assertional knowledge and terminological knowledge. Labels denote all the symbol names of sorts, constants, functions and predicates occurring in terms and formulas, and build a common hierarchy of the symbols. The semantics of the label-based language is specified in keeping the constraints on hierarchical relationships among symbol names. More precisely, labels and their hierarchy are interpreted as the combinations of different roles of symbols. We develop a sorted resolution system for label-based clausal formulas by introducing labeled substitutions and hierarchical resolution rules. This system performs reasoning on a label hierarchy, gained by various usages of labels and their hierarchical relationships (e.g. the same label might be used for function and predicate symbols).

This paper is organized as follows. Section 2 explains the notions of logic with sort-hierarchy[14] and an order-sorted logic with sort predicates. In

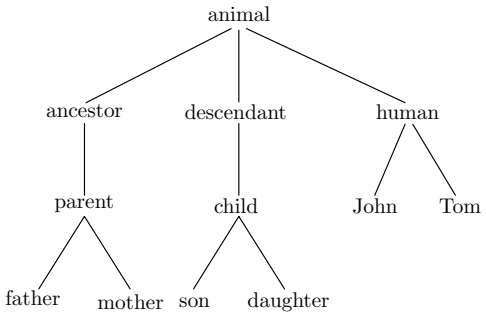

Fig. 1 A terminological hierarchy

Section 3, we show a requirement for hierarchical expressions, under the discussion about various usages of labels in a terminological hierarchy. Section 4 provides the syntax and semantics of a label-based language. In Section 5, we define labeled substitutions and hierarchical resolution rules in an inference system for label-based expressions. In Section 6, we discuss conclusions and future work.

§2 Logic with sort-hierarchy

Order-sorted logic is a first-order predicate logic with sort-hierarchy. A *sort-hierarchy* is a pair $(S, <_s)$ of a set S of sort symbols $s, s_1, s_2, \ldots$ and a subsort relation $<_s$ over S. A subsort relation is defined by a set of subsort declarations $s_i <_s s_j$. For instance, $father <_s parent$ and $mother <_s parent$ declare that *father* and *mother* are subsorts of the sort *parent*. As shown in Fig. 1, the hierarchy can be built by subsort declarations. We use sort symbols to express the restricted domains and ranges of variables, functions and predicates. A *sorted variable* is denoted as $x\colon s$. For example, $x\colon father$ is a sorted variable of the sort *father*. A function declaration is denoted by $f\colon s_1 \times \cdots \times s_n \to s$, and a predicate declaration is denoted by $p\colon s_1 \times \cdots \times s_n$.

For knowledge base reasoning, Beierle et al.[4] extended an order-sorted resolution system that can strongly connect separated terminological knowledge and assertional knowledge. In the sorted logic, each sort symbol can be used not only as the sort of a term but also as a unary predicate (called a *sort predicate*). Let C, C_1, C_2 be clauses, s, s_1, s_2 be sorts (or sort predicates), θ be a sorted substitution, and t, t_1, t_2 be sorted terms. In addition to sorted substitutions, the authors introduced inference rules of sort predicates as follows:

$$\frac{\neg s_1(t_1) \vee C_1 \quad s_2(t_2) \vee C_2}{(C_1 \vee C_2)\theta} \ (subsort\ resolution\ rule)$$

if $s_2 \leq_s s_1$ (i.e. $s_2 = s_1$ or $s_2 <_s s_1$) and $t_1\theta = t_2\theta$, and

$$\frac{\neg s(t) \vee C}{C\theta} \ (elimination\ rule)$$

if $t\theta$ is a term of the sort s or a subsort of s. Alternatively, in order to explicitly embed negative information in a sort-hierarchy, Kaneiwa and Tojo[11] developed an order-sorted resolution system with structured sorts. In both sorted logics, sort predicates are usefully introduced for interacting terminological knowledge and assertional knowledge.

§3 Requirements for hierarchical expressions

In this section, we first show possible usages of labels as symbol names using order-sorted logic, which will entail a requirement for hierarchical expressions of symbol names. For terminological knowledge, we then discuss the necessity of a sublabel relation to generally represent different types of subordinate relations.

Five uses for labels. Given the terminological hierarchy as shown in Fig. 1, each element of the hierarchy (e.g. *John*, *human*, *father* and *child*) is available for the names of various symbols: sorts, constants, functions, unary predicates and binary predicates. If the label 'John' is used as a constant, then the formula $parent(John)$ can describe the sentence "John is a parent." If the label 'human' plays the role of a unary predicate, the formula $human(John)$ is the sentence "John is human," in which 'human' can be relabeled more precisely as 'is_human.' For the label 'father' as a function that maps from a person into its father, the term $father(Tom)$ expresses Tom's father. If we have the relationship between a father and his child, then 'child' as a binary predicate coincides with 'is_a_child_of.' Hence $child(Tom, John)$ expresses "Tom is a child of John." Finally, the formula $walking(John : father)$ indicates "John who is a father is walking" where 'father' is used as a sort.

These usages reflect the meaning and pragmatics of natural language words. Fig. 2 classifies the roles of symbols as conceivable for the labels. Every label except for *John* and *Tom* can be used as a unary predicate or sort, and the labels exclusive of *animal* and *human* can be regarded as binary predicates.

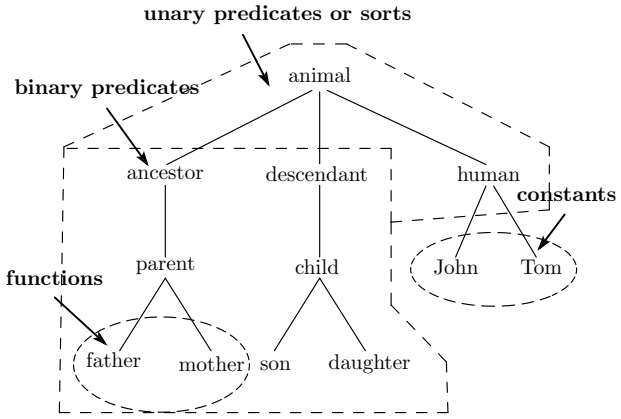

Fig. 2 The roles of symbols in a terminological hierarchy

Moreover, *mother* and *father* can be functions as sublabels of *parent*, and $John$ and Tom can be constants as sublabels of *human*.

When logically supporting the usages, the formality of logic gives rise to the syntax limited by the rigorous definition of symbol roles. Therefore, the labels variously used as above cannot be incorporated in the hierarchical information. To overcome the limitation, we will have to devise a method for covering various types of subordinate relations, which are composed from sorts, constants, functions, unary predicates and binary predicates. In other words, there are different combinations of symbols for each sublabel relation as follows:

types of subordinate relation R_i	**sublabel relation**
R_1 : function $\prec$ sort	
R_2 : sort $\prec$ sort	
R_3 : sort $\prec$ unary predicate	father $\prec$ parent
R_4 : binary predicate $\prec$ binary predicate	
...	

The sort-hierarchy in order-sorted logic does not express such hierarchical relationships including other kinds of symbols. Even if they are described in first-order formulas, we lose the conciseness of terminological knowledge, namely, different types of subordinate relations $R_1, R_2, \ldots$ cannot be generally treated as one sublabel relation. In the next section, we will formalize a label-based language for concisely representing the various hierarchical relationships.

§4 Label-based language $\mathcal{L}_{LB}$

We present *a label-based language* where labels are commonly used as the names of different kinds of symbols. Thus, a hierarchy over a set of labels (which we will call *a label hierarchy*) generally imply various hierarchies of first-order symbols, and the terms and formulas in the language are composed of labels.

4.1 Syntax

Let LB be a set of labels. In the following, we write its elements as L, M, N, or $L_0, L_1, L_2, \ldots$. A subordinate relation of labels (called *a sublabel relation*) is declared as follows:

$$L_2 \prec L_1,\ L_3 \prec L_1,\ \ldots$$

For example, for $bird, animal \in LB$, the declaration $bird \prec animal$ indicates that $bird$ is a sublabel of $animal$.

A label-based language $\mathcal{L}_{LB}$ contains a label hierarchy $LH = (LB, \prec)$ where LB is a set of labels and $\prec$ is a sublabel relation over LB. Note that the label hierarchy is a partially ordered set of labels. By indexing c, fn, pn, or s to each label L, the labels

$$L^c, L^{fn}, L^{pn}, L^s$$

are distinguished as constants, n-ary functions, n-ary predicates and sorts, respectively. Using this notation, symbols in a first-order language are introduced by the labels.

- $C \subseteq \{L^c \mid L \in LB\}$ is a set of constant symbols of labels,
- $F_n \subseteq \{L^{fn} \mid L \in LB\}$ is a set of n-ary function symbols of labels,
- $P_n \subseteq \{L^{pn} \mid L \in LB\}$ is a set of n-ary predicate symbols of labels.

We denote the set of all function symbols by F and the set of all predicate symbols by P.

Let $S \subseteq \{L^s \mid L \in LB\}$ be a set of sort symbols of labels. *A label-based sort-hierarchy* is obtained by a label hierarchy in such a way that a subsort relation $<_s$ over S is defined on a sublabel relation as below:

$$L_i^s <_s L_j^s \text{ if there are } L_i^s, L_j^s \in S \text{ such that } L_i \prec L_j$$

Namely, given a label hierarchy $LH = (LB, \prec)$, a label-based sort-hierarchy is a pair $(S, <_s)$ where S is a set of sort symbols denoted by labels in LB and $<_s = \{(L_i^s, L_j^s) \in S \times S \mid L_i \prec L_j\}$. The domain of each sorted variable

(denoted $x{:}\, L_i^s$) in $\mathcal{L}_{LB}$ is restricted by a sort L_i^s in S. $V_{L_i^s}$ is a set of variables $x{:}\, L_i^s, y{:}\, L_i^s, \ldots$ of sort L_i^s.

Remark. Each label-based language can be flexibly defined by containing any sets C, F, P, S of symbols which labels in LB denote. For example, consider a language where a label $L \in LB$ is employed as a constant $L^c \in C$ and a unary function $L^{f1} \in F$, but another label $M \in LB$ is employed only as a unary predicate $M^{p1} \in P$ (i.e. $M^c \notin C$ and $M^{f1} \notin F$).

Definition 4.1 (Labeled signature Σ_{LH})
Let LH be a label hierarchy. A signature of a label-based language $\mathcal{L}_{LB}$ with LH (called a labeled signature) is a tuple $\Sigma_{LH} = (S, C, F, P, D)$ where

- $(S, <_s)$ is a label-based sort-hierarchy,
- C, F, P are sets of constant symbols, function symbols and predicate symbols of labels with $C \cap F \cap P = \emptyset$,
- D is a set of sort declarations such that
  - $L^c{:} \to L_0^s \in D$ for every $L^c \in C$ (constant declaration),
  - $L^{fn}{:}\, L_1^s \times \cdots \times L_n^s \to L_0^s \in D$ for every $L^{fn} \in F$ (function declaration),
  - $L^{pn}{:}\, L_1^s \times \cdots \times L_n^s \in D$ for every $L^{pn} \in P$ (predicate declaration),
  - $L^{fn}{:}\, L_1^s \times \cdots \times L_n^s \to L_0^s \in D$ iff $L^{pn+1}{:}\, L_0^s \times L_1^s \times \cdots \times L_n^s \in D$.

A labeled signature can include more than one sort declaration for each symbol in $C \cup F \cup P$. For example, for $John^c \in C$, we have the constant declarations $John^c{:}\, human^s$ and $John^c{:}\, animal^s$ in D. For each label $L \in LB$, if $L^{fn} \in F$ and $L^{pn+1} \in P$, then some of their sort declarations might be different but the condition that $L^{fn}{:}\, L_1^s \times \cdots \times L_n^s \to L_0^s \in D$ $(n \geq 1)$ if and only if $L^{pn+1}{:}\, L_0^s \times L_1^s \times \cdots \times L_n^s \in D$ must be satisfied.

In a label-based language, we can view terms and formulas as compositions of mere labels when the role of each label is undescribed. We suppose that the expression

$$L_1(x{:}\, L_2) \wedge L_3(y{:}\, L_4)$$

is a formula of the form $A \wedge B$ where A, B are atomic formulas. Then, we recognizes that $L_1(x{:}\, L_2)$ and $L_3(y{:}\, L_4)$ are atomic formulas, and the positions of the labels determine that L_1, L_3 are unary predicates and L_2, L_4 are the sorts of variables x, y. In rigorous formalism, symbols in any formula must respectively have their unique roles and defer to a labeled signature Σ_{LH}. For example, we have

$$L_1^{p1}(x{:}\, L_2^s) \wedge L_3^{p1}(y{:}\, L_4^s)$$

where $p1$ and s indicate their unique roles as a unary predicate and a sort respectively. We define expressions of the language $\mathcal{L}_{LB}$: *label terms* and *label formulas* as follows.

Definition 4.2 (Label terms of sort L_0^s)
The set $\mathit{TERM}_{L_0^s}$ of label terms of sort L_0^s is defined inductively by the following rule:

$$t_{L_0^s} ::= x{:}\, L_0^s \;\mid\; L^c{:}\, L_0^s \;\mid\; L^{fn}(t_{L_1^s}, \ldots, t_{L_n^s}){:}\, L_0^s \;\mid\; t_{M^s}$$

where $L^c{:} \rightarrow L_0^s \in D$, $L^{fn}{:}\, L_1^s \times \cdots \times L_n^s \rightarrow L_0^s \in D$, for $1 \leq i \leq n$, $t_{L_i^s}$ is a label term of sort L_i^s, and t_{M^s} is a label term of sort M^s with $M^s <_s L_0^s$.

The set of label terms of all sorts is denoted by $\mathit{TERM} = \bigcup_{L^s \in S} \mathit{TERM}_{L^s}$.

Example 4.1
Consider the following label terms:

$$\mathit{John}^c{:}\, \mathit{human}^s,\ \ \mathit{father}^{f1}(\mathit{father}^{f1}(\mathit{Tom}^c{:}\, \mathit{human}^s))$$

The labels John and Tom are used as constants, the label *father* is as a unary function, and the label *human* is as a sort.

Definition 4.3 (Label formulas)
The set *FORM* of label formulas is defined inductively by the following rule:

$$A ::= L^{pn}(t_{L_1^s}, \ldots, t_{L_n^s}) \;\mid\; \neg A \;\mid\; A_1 \vee A_2 \;\mid\; A_1 \wedge A_2 \;\mid \exists x{:}\, L^s A \;\mid\; \forall x{:}\, L^s A$$

where $L^{pn}{:}\, L_1^s \times \cdots \times L_n^s \in D$, and $t_{L_i^s} \in \mathit{TERM}_{L_i^s}$ for $1 \leq i \leq n$.

Example 4.2
To construct an atomic formula, a label is employed as an n-ary predicate. For example, the label *father* is a unary predicate in the following formula:

$$\mathit{father}^{p1}(\mathit{John}^c{:}\, \mathit{human}^s)$$

whereas the same label is a unary function father^{f1} in Example 4.1.

4.2 Semantics

The semantics of a label-based language $\mathcal{L}_{LB}$ is defined in the usual manner of order-sorted logic. We interpret a label hierarchy LH in restricted

structures. Normally, a sort-hierarchy is interpreted by a subset relation of sorts. In contrast, the interpretation of a label hierarchy implies many relationships among constants, functions, predicates and sorts (i.e. subordinate relations over $S \cup C \cup F \cup P$). If a sublabel relation $L_i \prec L_j$ holds in a label hierarchy LH, then the labels L_i, L_j can be regarded semantically as various symbols in $S \cup C \cup F \cup P$. This observation leads to simple sublabel relations and complicated sublabel relations.

Simple sublabel relations are obtained by the same role symbols as follows:

(1) $L_i^s \prec L_j^s$: a subsort relation as $I(L_i^s) \subseteq I(L_j^s)$
(2) $L_i^{pn} \prec L_j^{pn}$: a subpredicate relation as $I(L_i^{pn}) \subseteq I(L_j^{pn})$

where $L_i^X \prec L_j^Y \Leftrightarrow_{def}$ there are $L_i^X, L_j^Y \in S \cup C \cup F \cup P$ such that $L_i \prec L_j$, and $I(L_i^X)$ denotes an interpretation of symbol L_i^X. A subsort relation $L_i^s \prec L_j^s$ of sorts (resp. a subpredicate relation $L_i^{pn} \prec L_j^{pn}$ of n-ary predicates) can be interpreted by the subset relation of $I(L_i^s)$ and $I(L_j^s)$ (resp. $I(L_i^{pn})$ and $I(L_j^{pn})$). For example, let $human \prec animal$ and $parent \prec ancestor$ for $human, animal, ancestor, parent \in LB$. Then, the simple sublabel relations $human^s \prec animal^s$ and $parent^{p2} \prec ancestor^{p2}$ are interpreted by $I(human^s) \subseteq I(animal^s)$ and $I(parent^{p2}) \subseteq I(ancestor^{p2})$. Furthermore, constants and functions may have a semantic association with the same role symbols, but in this paper we do not assume their sublabel relations.

Complicated sublabel relations combined by different role symbols are conceivable as in the following:

(3) $L_i^c \prec L_j^s$: a membership relation of constants and sorts as $I(L_i^c) \in I(L_j^s)$
(4) $L_i^f \prec L_j^s$: a subset relation of the ranges of functions and sorts as
$$\{u \mid (u_1, \ldots, u_n, u) \in I(L_i^{fn})\} \subseteq I(L_j^s)$$
(5) $L_i^s \prec L_j^{p1}$: a subset relation of sorts and unary predicates as $I(L_i^s) \subseteq I(L_j^{p1})$

In semantics, constant and function symbols are interpreted as mappings into individuals, and sort symbols are interpreted as sets of individuals. Hence, each constant (and function) must belong to the sorts denoted by the same label and its superlabels. For all $L_i^c \prec L_j^s$, the membership relation $I(L_i^c) \in I(L_j^s)$ holds, and for all $L_i^f \prec L_j^s$, the subset relation $\{u \mid (u_1, \ldots, u_n, u) \in I(L_i^{fn})\} \subseteq I(L_j^s)$ holds. On the other hand, both unary predicates and sorts correspond to sets of individuals, so that $L_i^s \prec L_j^{p1}$ is interpreted by the subset rela-

tion $I(L_i^s) \subseteq I(L_j^{p1})$. Analogously, the inverse $L_i^{p1} \prec L_j^s$ is interpreted as $I(L_i^{p1}) \subseteq I(L_j^s)$. For example, let $John \prec human$ and $father \prec animal$ for $John, father, human, animal \in LB$. The complicated sublabel relations $John^c \prec animal^s$, $father^{f1} \prec animal^s$ and $human^s \prec animal^{p1}$ are respectively interpreted by $I(John^c) \in I(animal^s)$, $\{u \mid (u_1, u) \in I(father^{f1})\} \subseteq I(animal^s)$ and $I(human^s) \subseteq I(animal^{p1})$.

Under these semantic specifications, we define restricted structures on a labeled signature Σ_{LH} as follows.

Definition 4.4 (Σ_{LH}-structure)
Let Σ_{LH} be a labeled signature. A sorted structure M is a pair (U, I) of a nonempty set U and an interpretation function I for $S \cup C \cup F \cup P$ such that:

1. $I(L^s) \subseteq U$,
2. $I(L^c) \in I(L_0^s)$ with $L^c{:}\rightarrow L_0^s \in D$,
3. $I(L^{fn}){:}\, I(L_1^s) \times \cdots \times I(L_n^s) \rightarrow I(L_0^s)$ with $L^{fn}{:}\, L_1^s \times \cdots \times L_n^s \rightarrow L_0^s \in D$,
4. $I(L^{pn}) \subseteq I(L_1^s) \times \cdots \times I(L_n^s)$ with $L^{pn}{:}\, L_1^s \times \cdots \times L_n^s \in D$.

A Σ_{LH}-structure M is a sorted structure such that for all $L_i \prec L_j \in D$, the following conditions hold:

1. $I(L_i^s) \subseteq I(L_j^s)$ with $L_i^s, L_j^s \in S$,
2. $I(L_i^{pn}) \subseteq I(L_j^{pn})$ with $L_i^{pn}, L_j^{pn} \in P$,
3. $I(L_i^c) \in I(L_j^s)$ with $L_i^c \in C$ and $L_j^s \in S$,
4. $\{u \mid (u_1, \ldots, u_n, u) \in I(L_i^{fn})\} \subseteq I(L_j^s)$ with $L_i^{fn} \in F$ and $L_j^s \in S$,
5. $I(L_i^s) \subseteq I(L_j^{p1})$ with $L_i^s \in S$ and $L_j^{p1} \in P$,
6. $I(L_i^{p1}) \subseteq I(L_j^s)$ with $L_i^{p1} \in P$ and $L_j^s \in S$.

Remark. Although each Σ_{LH}-structure satisfies the additional conditions for the labeled signature Σ_{LH}, the conditions are imposed only on the symbols introduced in a label-based language $\mathcal{L}_{LB}$. For each label, any kinds of symbols are not always introduced. For example, for the labels $ancestor$, $animal \in LB$, we can define $ancestor^{p1}, animal^{p1}, ancestor^{p2} \in P$ but $animal^{p2} \notin P$ in $\mathcal{L}_{LB}$. In this case, the sublabel relation $ancestor \prec animal$ imposes the condition $I(ancestor^{p1}) \subseteq I(animal^{p1})$ but does not the condition $I(ancestor^{p2}) \subseteq I(animal^{p2})$.

A variable assignment on a Σ_{LH}-structure $M = (U, I)$ is a mapping $\alpha{:}\, V \rightarrow U$ such that $\alpha(x{:}\, L_i^s) \in I(L_i^s)$ for all $x{:}\, L_i^s \in V$. The variable assignment

$\alpha[x{:}\, L^s/d]$ is defined as $(\alpha - \{(x{:}\, L^s, \alpha(x{:}\, L^s))\}) \cup \{(x{:}\, L^s, d)\}$.

Definition 4.5 (Σ_{LH}-interpretation)
A Σ_{LH}-interpretation $\mathcal{I}$ is a pair (M, α) where M is a Σ_{LH}-structure and α is a variable assignment on M. The denotation $[\![\ ]\!]_\alpha$ is defined by the following rules:

- $[\![x{:}\, L_0^s]\!]_\alpha = \alpha(x{:}\, L_0^s)$,
- $[\![L^c{:}\, L_0^s]\!]_\alpha = I(L^c)$,
- $[\![L^{fn}(t_1, \ldots, t_n){:}\, L_0^s]\!]_\alpha = I(L^{fn})([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha)$.

Let $\mathcal{I} = (M, \alpha)$ be a Σ_{LH}-interpretation. The Σ_{LH}-interpretation $\mathcal{I}[x{:}\, L^s/d]$ denotes $(M, \alpha[x{:}\, L^s/d])$.

Definition 4.6 (Satisfaction relation)
Let $\mathcal{I} = (M, \alpha)$ be a Σ_{LH}-interpretation and F be a label formula. The satisfaction relation $\mathcal{I} \models F$ is defined by the following rules:

- $\mathcal{I} \models L^{pn}(t_1, \ldots, t_n)$ *iff* $([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha) \in I(L^{pn})$,
- $\mathcal{I} \models \neg A$ *iff* $\mathcal{I} \not\models A$,
- $\mathcal{I} \models A \wedge B$ *iff* $\mathcal{I} \models A$ and $\mathcal{I} \models B$,
- $\mathcal{I} \models A \vee B$ *iff* $\mathcal{I} \models A$ or $\mathcal{I} \models B$,
- $\mathcal{I} \models A \Rightarrow B$ *iff* $\mathcal{I} \not\models A$ or $\mathcal{I} \models B$,
- $\mathcal{I} \models \forall x{:}\, L^s A$ *iff* for all $d \in I(L^s)$, $\mathcal{I}[x{:}\, L^s/d] \models A$,
- $\mathcal{I} \models \exists x{:}\, L^s A$ *iff* for some $d \in I(L^s)$, $\mathcal{I}[x{:}\, L^s/d] \models A$.

Let F be a label formula and Γ be a set of label formulas. F is Σ_{LH}-satisfiable if there exists a Σ_{LH}-interpretation $\mathcal{I}$ such that $\mathcal{I} \models F$ ($\mathcal{I}$ is called a Σ_{LH}-model of F). Γ is Σ_{LH}-satisfiable if there exists a Σ_{LH}-interpretation $\mathcal{I}$ such that for all $F \in \Gamma$, $\mathcal{I} \models F$ ($\mathcal{I}$ is called a Σ_{LH}-model of Γ).

§5 Inference system

5.1 Deduction rules

We consider reasoning for clausal label formulas with a label hierarchy LH. Let l_i be an atomic formula or the negation of an atomic formula. The clausal form $l_1 \vee \cdots \vee l_n$ ($n \geq 0$) (called a *label clause*) expresses the label formula of the universal closure $\forall x_1 \cdots \forall x_m (l_1 \vee \cdots \vee l_n)$ where $x_1, \ldots, x_m$ are all the variables occurring in $l_1 \vee \cdots \vee l_n$. The empty clause ($n = 0$) is denoted by $\Box$. We here show enriched deduction for label clauses that conforms to the interpretation of the simple and complicated sublabel relations we explained in

Section 4.2. Let C be a label clause and $\bar{\mu}$ be a sequence $t_1, \ldots, t_n$ of label terms. For a label hierarchy LH, we have the following deduction rules:

(1) $\dfrac{L_i^{pn}(\bar{\mu})}{L_j^{pn}(\bar{\mu})}$ if $L_i^{pn} \prec L_j^{pn}$ (2) $\dfrac{C[x\colon L_j^s]}{C[t_{L_i^s}]}$ if $L_i^s \prec L_j^s$

(3) $\dfrac{C[x\colon L_j^s]}{C[L_i^c\colon L_k^s]}$ if $L_i^c \prec L_j^s$, $L_i^c \prec L_k^s$ *and* $L_k^s \not\prec L_i^c$

(4) $\dfrac{}{L_j^{p1}(x\colon L_i^s)}$ if $L_i^s \prec L_j^{p1}$

(5) $\dfrac{C[x\colon L_j^s]}{C[L_i^{fn}(\bar{\mu})\colon L_k^s]}$ if $L_i^{fn} \prec L_j^s$, $L_i^{fn} \prec L_k^s$ *and* $L_k^s \not\prec L_i^{fn}$

where $L_i^X \prec L_j^Y \Leftrightarrow_{def}$ there are $L_i^X, L_j^Y \in S \cup C \cup F \cup P$ such that $L_i \prec L_j$, and $C[t]$ denotes a label clause C where a term t occurs.

5.2 Labeled substitution and resolution rules

To conform to a practical viewpoint, we will adopt a refutation proof method obtained by resolution for clausal forms. As a variant of the deduction system, we develop a resolution system for label clauses, containing sorted and labeled substitutions and hierarchical resolution rules.

The subsort relation $L_i^s <_s L_j^s$ results in deduction rule (2), which can be implemented by sorted substitutions in the resolution system.

Definition 5.1 (Sorted substitution)
A sorted substitution is a mapping θ from a finite set of sorted variables to *TERM* such that $\theta(x\colon L_i^s) \neq x\colon L_i^s$ and $\theta(x\colon L_i^s) \in \mathit{TERM}_{L_i^s}$.

Deduction rules (3) and (5) according to the complicated sublabel relations $L_i^c \prec L_j^s$ and $L_i^{fn} \prec L_j^s$ (explained in Section 4.2) carry out the replacements of $x\colon L_j^s$ into $L_i^c\colon L_k^s$ and $L_i^{fn}(\bar{\mu})\colon L_k^s$. To incorporate these replacements into resolution, we define a labeled substitution by extending the sorted substitution.

Definition 5.2 (Labeled substitution)
A labeled substitution is a mapping θ^+ from a finite set of sorted variables to *TERM* such that for each variable $x\colon L_j^s \in \mathit{Dom}(\theta)$, the following condition holds:

- $\theta^+(x\colon L_j^s) = L_i^c\colon L^s$ where $L_i \prec L_j$,
- $\theta^+(x\colon L_j^s) = L_i^{fn}(\bar{\mu})\colon L^s$ where $L_i \prec L_j$, or

- $\theta^+(x{:}\, L_j^s) = \theta(x{:}\, L_j^s)$ where θ is a sorted substitution.

A labeled substitution θ^+ is expanded to label terms and formulas E, denoted by $E\theta^+$. Let E, E' be expressions. A labeled substitution θ^+ is a unifier for E and E' if $E\theta^+ = E'\theta^+$. A unifier θ^+ for E and E' is most general (denoted by mgu) if for every unifier θ_i^+ for E and E' there exists a labeled substitution γ such that $\theta^+ = \theta_i^+\gamma$. In order to find a unique mgu, we assume that every sort-hierarchy is a lower semi-lattice, i.e., there exists the greatest lower bound for any two sorts. Even if a sort-hierarchy is not a lower semi-lattice, we can reconstruct it to be the structure, as discussed by Cohn.[6)]

Deduction rule (1) captures the interpretation of a subpredicate relation $L_i^{pn} \prec L_j^{pn}$ of n-ary predicates, and deduction rule (4) actualizes that a subset relation $L_i^s \prec L_j^{p1}$ of sorts and unary predicates makes $L_j^{p1}(x{:}\, L_i^s)$ valid. These deduction rules are embedded in the hierarchical resolution rules we will present as below.

Definition 5.3 (Hierarchical resolution rules)
Let t, t_i, t_i' be label terms, A, A' be atomic label formulas and C, C' be label clauses. The resolution system for $\mathcal{L}_{LB}$ includes the following hierarchical resolution rules:

Resolution principle

$$\frac{\neg A \vee C \quad A' \vee C'}{(C \vee C')\theta^+}$$

where θ^+ is a mgu for A and A'.

Sublabel rule

$$\frac{\neg L_j^{pn}(t_1, \ldots, t_m) \vee C \quad L_i^{pn}(t_1', \ldots, t_m') \vee C' \quad L_i \prec L_j}{(C \vee C')\theta^+}$$

where θ^+ is a mgu for $(t_1, \ldots, t_m)$ and $(t_1', \ldots, t_m')$.

Elimination rule

$$\frac{\neg L_j^{p1}(t) \vee C \quad L_i \prec L_j}{C\theta^+}$$

where $t\theta^+ \in \mathit{TERM}_{L_i^s}$, or $t\theta^+ = L_i^c{:}\, L^s$ or $L_i^{fn}(\bar{\mu}){:}\, L^s$.

Reflexivity rule

$$\frac{}{L \prec L}$$

Transitivity rule

$$\frac{L_i \prec L \quad L \prec L_j}{L_i \prec L_j}$$

The hierarchical resolution rules are similar to the subsort resolution rule and the elimination rule (in Section 2), since our resolution system is an extension of the sorted resolution system with sort predicates.[4] In other words, ordinary sorted resolution systems correspond to the hierarchical resolution system without the sublabel rule, the elimination rule and labeled substitutions. Moreover, the sorted resolution system with sort predicates corresponds to the hierarchical resolution system without the sublabel rule for n-ary predicates ($n > 1$) and labeled substitutions.

The hierarchical resolution system is applied to a knowledge base consisting of a labeled signature and a set of label clauses.

Definition 5.4 (Knowledge base)
Let LH be a label hierarchy. A knowledge base KB is a pair (Σ_{LH}, Γ) where Σ_{LH} is a labeled signature with LH and Γ is a nonempty finite set $\{C_1, C_2, \ldots\}$ of label clauses except for the empty clause.

We write $KB \models C$ if, for every Σ_{LH}-model $\mathcal{I}$ of Γ in KB, $\mathcal{I} \models C$.

Definition 5.5 (Consistency)
Let $KB = (\Sigma_{LH}, \Gamma)$ be a knowledge base. We write $KB \vdash_{LH} C$ if C is derivable from Γ in KB by applying hierarchical resolution rules. A knowledge base KB is inconsistent if $KB \vdash_{LH} \square$, and it is consistent otherwise.

The hierarchical relationships among labels generate new logical conclusions and inconsistent formulas, which are not treated in ordinary sorted inference systems.

Proposition 5.1
Let Σ_{LH} be a labeled signature and $KB = (\Sigma_{LH}, \Gamma)$ be a knowledge base. KB is inconsistent if and only if there exists at least one of the following inconsistent sets $\{A_1\theta^+, \ldots, A_n\theta^+\}$ with $KB \vdash_{LH} A_1, \ldots, KB \vdash_{LH} A_n$:

$\{A,\ \neg A\}$,
$\{\neg L^{p1}(t)\}$ with $t \in TERM_{L^s}$,
$\{L_i^{pn}(\bar{\mu}),\ \neg L_j^{pn}(\bar{\mu})\}$,
$\{\neg L_j^{p1}(L_i^c{:}\, L^s)\}$,
$\{\neg L_j^{p1}(L_i^{fn}(t_1, \ldots, t_n){:}\, L^s)\}$

where $L_i \prec L_j \in LH$.

Proof. ($\Leftarrow$) For each inconsistent set, the empty clause can be derived by the resolution system. ($\Rightarrow$) Any knowledge base KB does not contain the empty clause. So, in order to derive the empty clause, at least one resolution rule must be applied to KB. The resolution principle is applied when the inconsistent set $\{A, \neg A\}$ exists. The sublabel rule can derive the empty clause if there exists the set $\{L_i^{pn}(\bar{\mu}), \neg L_j^{pn}(\bar{\mu})\}$. For an application of the elimination rule, we can conclude that there is either $\{\neg L^{p1}(t)\}$, $\{\neg L_j^{p1}(L_i^c{:}\, L^s)\}$ or $\{\neg L_j^{p1}(L_i^{fn}(t_1, \ldots, t_n){:}\, L^s)\}$. ∎

Proposition 5.2
Let Σ_{LH} be a labeled signature, θ^+ be a labeled substitution, C be a label clause and $\mathcal{I}$ be a Σ_{LH}-interpretation. If $\mathcal{I} \models C$, then $\mathcal{I} \models C\theta^+$.

Proof. Let $x_1{:}\, L_1^s, \ldots, x_n{:}\, L_n^s$ be all the free variables occurring in C, and define $\theta^+{\uparrow}VS = \{(v{:}\, L^s, t) \in \theta^+ \mid v{:}\, L^s \in Dom(\theta^+) \cap VS\}$. For $x_j{:}\, L_j^s \notin Dom(\theta^+)$, $C\theta^+{\uparrow}\{x_j{:}\, L_j^s\} = C$ holds. For $x_j{:}\, L_j^s \in Dom(\theta^+)$, we have to check the three cases: (i) $\theta^+(x_j{:}\, L_j^s) = \theta(x_j{:}\, L_j^s)$, (ii) $\theta^+(x_j{:}\, L_j^s) = L_i^c{:}\, L^s$ with $L_i \prec L_j$ and (iii) $\theta^+(x_j{:}\, L_j^s) = L_i^{fn}(\bar{\mu}){:}\, L^s$ with $L_i \prec L_j$. For (i), $\mathcal{I} \models C\theta{\uparrow}\{x_j{:}\, L_j^s\}$ follows since θ is a sorted substitution. (ii) if $\theta^+(x_j{:}\, L_j^s) = L_i^c{:}\, L^s$, then the condition $I(L_i^c) \in I(L_j^s)$ in the Σ_{LH}-structure entails $\mathcal{I} \models C\theta^+{\uparrow}\{x_j{:}\, L_j^s\}$. (iii) if $\theta^+(x_j{:}\, L_j^s) = L_i^f(\bar{\mu}){:}\, L^s$, then $\mathcal{I} \models C\theta^+{\uparrow}\{x_j{:}\, L_j^s\}$ by the condition $\{u \mid (u_1, \ldots, u_n, u) \in I(L_i^{fn})\} \subseteq I(L_j^s)$ in the Σ_{LH}-structure. Therefore, we can derive $\mathcal{I} \models C\theta^+$. ∎

Theorem 5.1
Let Σ_{LH} be a labeled signature, C be a label clause and KB be a knowledge base. If $KB \vdash_{LH} C$, then $KB \models C$.

Proof. We verify it for applying (i) sublabel rule and (ii) elimination rule. For (i), suppose $\mathcal{I} \models \neg L_j^{pn}(\bar{\mu}) \vee C$ and $\mathcal{I} \models L_i^{pn}(\bar{\mu}') \vee C'$. Let θ^+ be a mgu for $\bar{\mu}$ and $\bar{\mu}'$. By Proposition 5.2, we have $\mathcal{I} \models \neg L_j^{pn}(\bar{\mu}\theta^+) \vee C\theta^+$ and $\mathcal{I} \models L_i^{pn}(\bar{\mu}'\theta^+) \vee C'\theta^+$

$$\dfrac{\dfrac{\neg hum(y\colon hum) \vee \neg anc(y\colon hum) \quad fat(J\colon hum)}{\neg hum(J\colon hum)} \quad \dfrac{fat \prec par \quad par \prec anc}{fat \prec anc}}{\dfrac{\neg hum(J\colon hum) \quad J \prec hum}{\square}}$$

$$\dfrac{\dfrac{\dfrac{\neg luc(y\colon hum) \vee \neg fat(y\colon hum) \quad \neg won(x) \vee luc(x)}{\neg won(y\colon hum) \vee \neg fat(y\colon hum)} \quad won(fat(T\colon hum))}{\neg fat(fat(T\colon hum)) \quad fat \prec fat}}{\square}$$

$$\dfrac{\dfrac{\dfrac{\neg luc(y\colon hum) \vee \neg mot(y\colon hum) \quad \neg won(x) \vee luc(x)}{\neg won(y\colon hum) \vee \neg mot(y\colon hum)} \quad won(fat(T\colon hum))}{\neg mot(fat(T\colon hum))}}{fail}$$

Fig. 3 Resolution steps for a knowledge base

with $\bar{\mu}\theta^+ = \bar{\mu}'\theta^+$. $L_i \prec L_j$ entails $I(L_i^{pn}) \subseteq I(L_j^{pn})$, and so $\mathcal{I} \models (C \vee C')\theta^+$. (ii) let $\mathcal{I} \models \neg L_j^{p1}(t) \vee C$. The conditions of the Σ_{LH}-structure imply $\mathcal{I} \models L_j^{p1}(t\theta^+)$, and then $\mathcal{I} \not\models \neg L_j^{p1}(t\theta^+)$. Hence, $\mathcal{I} \models C\theta^+$. ∎

Theorem 5.2
Let Σ_{LH} be a labeled signature and KB be a knowledge base. If KB has a Σ_{LH}-model, then KB is consistent.

Proof. Suppose that KB has a Σ_{LH}-model. Let us assume that KB is inconsistent. According to Proposition 5.1, there is an inconsistent set $\{A_1\theta^+, \ldots, A_n\theta^+\}$ such that $KB \vdash_{LH} A_1, \ldots, KB \vdash_{LH} A_n$. Theorem 5.1 and Proposition 5.2 lead to $KB \models A_1\theta^+, \ldots, KB \models A_n\theta^+$. However, the inconsistent set has no Σ_{LH}-model. This is a contradiction. Therefore, KB is consistent. ∎

Corollary 5.1
Let Σ_{LH} be a labeled signature and KB be a knowledge base. If $KB \vdash_{LH} \square$, then KB has no Σ_{LH}-model.

Proof. This can be straightforwardly proved by Theorem 5.2. ∎

5.3 An example of resolution

We give an example of resolution with respect to the terminological hierarchy shown in Fig. 2. Let the label hierarchy $LH = (LB, \prec)$ with

$ancestor \prec animal$, $descendant \prec animal$, $parent \prec ancestor$,
$child \prec descendant$, $son \prec child$, $daughter \prec child$,
$father \prec parent$, $mother \prec parent$, $human \prec animal$,
$John \prec human$, $Tom \prec human$,

and let the labeled signature $\Sigma_{LH} = (S, C, F_1, P_1 \cup P_2, D)$ where

$$
\begin{aligned}
S = \{\ & animal, ancestor, descendant, human, parent, child, \\
& father, mother, son, daughter\ \}, \\
C = \{\ & John, Tom\ \}, \\
F_1 = \{\ & father, mother\ \}, \\
P_1 = \{\ & animal, ancestor, descendant, human, parent, child, \\
& father, mother, son, daughter\ \}, \\
P_2 = \{\ & ancestor, descendant, parent, child, father, mother, \\
& son, daughter\ \}, \\
D = \{\ & John\colon \rightarrow human,\ Tom\colon \rightarrow human, \\
& father\colon human \rightarrow human,\ father\colon human, \\
& human\colon \top,\ won\_in\_a\_lottery\colon \top,\ lucky\colon \top, \quad \cdots \quad \}.
\end{aligned}
$$

Let the knowledge base $KB = (\Sigma_{LH}, \Gamma)$ where

$$
\begin{aligned}
\Gamma = \{\ & father(John\colon human), \\
& won\_in\_a\_lottery(father(Tom\colon human)), \\
& won\_in\_a\_lottery(x) \Rightarrow lucky(x)\ \}.
\end{aligned}
$$

The first and second facts state that John is a father and Tom's father won in a lottery, and the rule means that if x won in a lottery, then x was lucky. For the knowledge base KB, consider the following queries:

?-$human(y\colon human) \wedge ancestor(y\colon human)$,
?-$lucky(y\colon human) \wedge father(y\colon human)$,
?-$lucky(y\colon human) \wedge mother(y\colon human)$.

In order to decide the answer (yes or no) of each query in KB, the resolution system attempts to derive the empty clause from the negation of the query and the knowledge base KB. By applying hierarchical resolution rules, the derivation processes are given as in Fig. 3. The derivations for the first and second queries are successful, but the derivation for the final query fails.

§6 Conclusions

This paper investigates general representation and reasoning for the hierarchies composed by different kinds of first-order symbols. We have presented a label-based language as an extended order-sorted logic which provides the label-based expressions: a label hierarchy, label terms and label formulas. In the semantic connections of various symbols in the language, a label hierarchy can be commonly employed to express not only a sort-hierarchy but also hierarchies of different types of subordinate relations. To develop its reasoning system, we have defined labeled substitutions and hierarchical resolution rules that are extensions of sorted substitutions and sorted resolution rules respectively. Beierle et al. enhanced the reasoning power of an order-sorted resolution system by adding inference rules of sort predicates. Along the way, we have achieved a further refinement of the sorted resolution system in the label-based language.

We will need to redesign the hierarchical resolution rules in order to make the system complete. The derivability of the system is still insufficient to correspond to the class of Σ_{LH}-structures.

Acknowledgment

We thank the anonymous referees for many useful suggestions on this paper. This research has been partially supported by the Kayamori Foundation of Informational Science Advancement and the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (14780311).

References

1) H. Aït-Kaci and R. Nasr. LOGIN: A logic programming language with built-in inheritance. *Journal of Logic Programming*, pages 185–215, 1986.
2) H. Aït-Kaci and A. Podelski. Towards a meaning of LIFE. *Journal of Logic Programming*, pages 195–234, 1993.
3) F. Baader, H.-J. Bürckert, J. Heinsohn, J. Müller, B. Hollunder, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. DFKI Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz, Kaiserslautern, 1990.
4) 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.
5) A. G. Cohn. A more expressive formulation of many sorted logic. *Journal of Automated Reasoning*, 3:113–200, 1987.
6) A. G. Cohn. Taxonomic reasoning with many sorted logics. *Artificial Intelligence Review*, 3:89–128, 1989.

7) I. Horrocks, D. Fensel, J. Broekstra, S. Decker, M. Erdmann, C. Goble, F. van Harmelen, M. Klein, S. Staab, R. Studer, and E. Motta. OIL: The Ontology Inference Layer. Technical Report IR-479, Vrije Universiteit Amsterdam"Sciences, 2000.

8) I. Horrocks and U. Sattler. Ontology reasoning in the *SHOQ(D)* description logic. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001.

9) K. Nitta, et al. Knowledge representation of new HELIC-II. In *Workshop on Legal Application of Logic Programming,ICLP '94*, 1994.

10) K. Kaneiwa and S. Tojo. Event, property, and hierarchy in order-sorted logic. In *Proceedings of the 1999 Int. Conf. on Logic Programming*, pages 94–108. The MIT Press, 1999.

11) K. Kaneiwa and S. Tojo. An order-sorted resolution with implicitly negative sorts. In *Proceedings of the 2001 Int. Conf. on Logic Programming*, pages 300–314. Springer-Verlag, 2001. LNCS 2237.

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

13) M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

14) R. Socher-Ambrosius and P. Johann. *Deduction Systems*. Springer-Verlag, 1996.

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

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

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

18) K. Yokota. *Quixote:A Constraint Based Approach to a Deductive Object-Oriented Database*. PhD thesis, Kyoto University, 1994.