An Order-Sorted Resolution with Implicitly Negative Sorts

Ken Kaneiwa[1] and Satoshi Tojo[2]

[1] National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430 JAPAN
kaneiwa@nii.ac.jp
[2] Japan Advanced Institute of Science and Technology
1-1 Asahidai, Tatsunokuchi, Ishikawa 923-1292 JAPAN
tojo@jaist.ac.jp

Abstract. We usually use natural language vocabulary for sort names in order-sorted logics, and some sort names may contradict other sort names in the sort-hierarchy. These implicit negations, called lexical negations in linguistics, are not explicitly prefixed by the negation connective. In this paper, we propose the notions of structured sorts, sort relations, and the contradiction in the sort-hierarchy. These notions specify the properties of these implicit negations and the classical negation, and thus, we can declare the exclusivity and the totality between two sorts, one of which is affirmative while the other is negative. We regard the negative affix as a strong negation operator, and the negative lexicon as an antonymous sort that is exclusive to its counterpart in the hierarchy. In order to infer from these negations, we integrate a structured sort constraint system into a clausal inference system.

1 Introduction

Order-sorted logics, or many-sorted logics, have been well discussed as tools to represent hierarchical knowledge in the field of artificial intelligence [18, 4, 3, 7, 14, 16, 19]. Recently, description logics [15, 1, 2, 10] as outlined in [6] have been studied as a theoretical approach to terminological knowledge representation, which represent structured concepts by more primitive concepts, as are similar to those in a sort-hierarchy.

However, a sort-hierarchy may contain sorts with implicitly negative meanings. These negations are called *lexical negations* in linguistics and are distinct from the negative particle *not*. Since every sort name is a mere string or a symbol, these implicitly negative sorts are not interpreted to represent their original meanings. Nevertheless, a knowledge representation system should take account of the fact that the lexical negation '*unhappy*' is opposite in meaning to the positive expression '*happy*', or '*winner*' contradicts '*loser*', in a sort-hierarchy.

In order to realize this, we have to analyze the properties of lexical negations in natural language and then consider dealing with these negations in a sort-hierarchy. In [12], lexical negations (words that implicitly have negative meaning) are classified as follows:

(i) Negative affix (in-,un-,non-):
incoherent, inactive, unfix, nonselfish, illogical, impolite, etc.
(ii) Lexicon with negative meaning:
doubt (believe not), deny (approve not), prohibit (permit not), forget (remember not), etc.

First, we introduce a hybrid knowledge representation system of Beierle [3] that distinguishes between taxonomical information (in the sort-hierarchy) and assertional information (in the assertional knowledge base), as an extension of an order-sorted logic. This system can deal with the taxonomical information in an assertional knowledge base in which a sort symbol can be expressed as a unary predicate (called a sort predicate) in clausal forms. Since a sort and a unary predicate have the same expressive power, we can regard a subsort declaration $s_1 \sqsubseteq s_2$ as the following logical implication form:

$$s_1(x) \rightarrow s_2(x)$$

where the unary predicates $s_1(x), s_2(x)$ corresponding to the sorts s_1, s_2 are sort predicates. Let C, C_1, C_2 be clauses, s, s_1, s_2 sorts (or sort predicates), θ a sorted substitution, and t, t_1, t_2 sorted terms. In order to use the information in a sort-hierarchy in a clausal knowledge base, or an assertional knowledge base, the following inference rules:

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

where $s_2 \sqsubseteq_S s_1$ and $\theta(t_1) = \theta(t_2)$, and

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

where $Sort(t)$[1] $\sqsubseteq_S s$, are added to his resolution system. This hybrid knowledge representation system provides a useful way to deal with a sort-hierarchy in a clausal knowledge base.

Hereafter, we illustrate the deductions which we would like to realize in a sort-hierarchy with lexical negations. The first example concerns a negative affix: *unhappy*. A sort *unhappy* is not only a negative expression but also a subexpression of *emotional*. Hence, the sort *emotional* can be derived from *unhappy* (like *happy*), whereas it cannot be derived from the classical negation $\neg$*happy*. In addition, the sort *unhappy* is a stronger negative statement than the classical negation $\neg$*happy*, so that $\neg$*happy* can be derived from *unhappy*, but *unhappy* cannot be derived from $\neg$*happy*. The fact $\neg$*emotional*(*bob*), that is the person *bob* is not emotional, yields that '$\neg$*happy*(*bob*) $\wedge$ $\neg$*unhappy*(*bob*).' In contrast, no premise can derive '$\neg$*happy*(*bob*) $\wedge$ $\neg\neg$*happy*(*bob*).' This shows the sort *unhappy* has the meaning of *emotional*, but the classical negation $\neg$*happy* does not have the meaning of *emotional*.

[1] For any sorted term t, the function $Sort(t)$ assigns its sort to term t.

The next example concerns lexicon with negative meaning: *loser*. Suppose a sort-hierarchy where both of *winner* and *loser* are subsumed by *player*. Needless to say, *loser* is different from the classical negation $\neg winner$ of *winner*, because *loser* means the negative event opposite to an event denoted by *win* but the classical negation $\neg winner$ denies the event denoted by *win*. Therefore, the supersort *player* can be derived from *loser* (or *winner*), but not from $\neg winner$. Furthermore, if the person *tom* is not a player, then the negations $\neg winner$ and $\neg loser$ can be derived. In contrary, if the person *tom* is a player, then *winner* or *loser* holds in the totality (i.e. *tom* must be a winner or a loser) of *winner* and *loser*. By the totality, if the person *tom* is a player but not a loser ($\neg loser$), then *tom* is a winner. If *tom* is neither a winner nor a loser ($\neg winner \wedge \neg loser$), then *tom* is not a player ($\neg player$)

We would like to derive these facts from implicitly negative sorts. However, it is hard to describe implicitly negative sorts in the sort-hierarchy, so that many knowledge bases would lose the property that implicit negations are exclusive to their antonyms and partial to their classical negation. In fact, Beierle's inference system for sort-hierarchy and order-sorted substitutions in clauses do not generate any reasoning mechanism for negative sorts. Description logics and feature logics [16] provide complex sort expressions but not any clausal reasoning mechanism with these expressions. Therefore, these inference systems with sort-hierarchy cannot immediately derive the above results from subsorts, supersorts and classical negation. In the following sections, we will propose a method to describe the properties of lexical negations implicitly included in a sort-hierarchy and develop an inference machinery.

This paper is organized as follows. In Section 2 presents an order-sorted logic that includes the complex sort expressions of implicit negations. We give an account of structured sorts, sort relations, and contradiction in a sort-hierarchy. Section 3 and Section 4 present the formalization of order-sorted logic with structured sorts, and systems of clausal resolution. In Section 5, we give our conclusions and discuss future work.

2 Implicitly negative sorts

In order to deal with implicitly negative sorts in a sort-hierarchy, we introduce structured sorts, sort relations, and contradiction in a sort-hierarchy into an order-sorted logic. These notions can be used to declare the properties of implicitly negative sorts in a sort-hierarchy.

2.1 Structured sorts and sort relations

We consider the representation of sorts in a hierarchy whose names are declared as lexical negations (classified as negative affixes or lexicons with negative meaning). In this paper, we call a sort denoted by a word with negative affix *a negative sort* and a sort denoted by a lexicon with negative meaning *an opposite sort*. In general, we call these sorts *implicitly negative sorts*. To represent these negative

sorts, we introduce the notation of structured sorts and relations between sorts whereby a negative sort is defined by the structured sort with strong negation operator [17] and an opposite sort is defined by exclusivity. In particular, we denote an opposite sort as exclusive to its antonymous sort in a hierarchy, so that these two sorts exclude each other but neither sort is negative. In fact, we should not say that an opposite sort is negative, rather we should say that these two sorts are opposite in meaning.

Structured sorts are constructed from atomic sorts, the connectives $\sqcap, \sqcup$, and the negative operators; $\overline{happy}$ is a complement (classical negation) of *happy*, and $\sim happy$ is a negative sort (strong negation) of that.

We now give several relations between structured sorts in order to represent implicitly negative sorts embedded in a sort-hierarchy. '$\sqsubseteq_S$' denotes a subsort relation between structured sorts. With this relation, a set of sorts are partially ordered (i.e. reflexive, anti-symmetric, and transitive). '$=_S$' denotes an equivalence relation between structured sorts. Furthermore, we add an exclusivity relation '$\|$' and a totality relation '$|_{s_i}$' between structured sorts; if $s \parallel s'$ then s and s' are exclusive, and if $s \mid_{s_i} s'$ then s together with s' composes the whole of s_i.

Using these sort relations, we can define the following properties (totality, partiality, and exclusivity) to declare various negations (in particular, lexical negations), as in Table 1.

Table 1. Three negations

Negation type	Expression	Relationship		Property
(1) Complement (classical negation)	$\overline{happy}$	$\overline{happy} \mid_\top happy$ $\overline{happy} \parallel happy$	(in Axioms)	totality exclusivity
(2) Negative sort (strong negation)	$\sim happy$	$\sim happy \parallel happy$ $\sim happy \sqsubseteq_S \overline{happy}$	(in Axioms)	exclusivity partiality
(3) Opposite sort (antonym)	*sad*	$sad \parallel happy$	(in Declarations)	exclusivity

2.2 A contradiction in a sort-hierarchy

We present a contradiction in a sort-hierarchy containing the three negations (complement, negative sort, and opposite sort) that we have explained.

A deductive system with implicitly negative sorts has to determine a contradiction in a sort-hierarchy in order that it can provide a sound inference

mechanism derived from the three negations and their relations to each other. In classical logic, we can say that a set Δ of formulas is contradictory if a formula A and its classical negation $\neg A$ are simultaneously derivable from Δ. In this case, we can syntactically establish the contradiction, because $\neg A$ indicates the negation of A by the negative operator $\neg$. Given the opposite sorts s and s' (e.g. *winner* and *loser*), we should also say that Δ is contradictory if the two formulas $s(x), s'(x)$ denoted by the sort predicates s and s' are simultaneously derivable from Δ. This indicates that the sort symbols s and s' have a negative relation to each other in our language definition. [2]

Using an exclusivity relation between sorts, we give a definition of contradictions in a sort-hierarchy that supports deduction from the three negations. A set Δ of formulas is said to be contradictory if there exist sorts s, s' such that $s \parallel s'$ and $s(t)$ and $s'(t)$ are derivable from Δ. In section 3, we will redefine the notion of contradiction in a sort-hierarchy that enables our deduction system to ensure the consistency of a knowledge base.

3 An order-sorted logic with structured sorts

On the specification we propose in Section 2, we define the syntax and semantics of an order-sorted logic with structured sorts.

3.1 Structured sort signature

Given a set $\mathcal{S}$ of sort symbols, every sort $s_i \in \mathcal{S}$ is called an atomic sort. We define the set of structured sorts composed by the atomic sorts, the connectives, and the negative operators as follows.

Definition 1 (Structured sorts). *Given a set $\mathcal{S}$ of atomic sorts, the set $\mathcal{S}^+$ of structured sorts is defined by:*

(1) If $s \in \mathcal{S}$, then $s \in \mathcal{S}^+$,
(2) If $s, s' \in \mathcal{S}^+$, then $(s \sqcap s')$, $(s \sqcup s')$, $(\overline{s})$, $(\sim s) \in \mathcal{S}^+$.

The structured sort $\overline{s}$ is called the classical negation of sort s and the structured sort $\sim s$ is called the strong negation of sort s. For convenience, we can denote $s \sqcap s'$, $s \sqcup s'$, $\overline{s}$ and $\sim s$ without parentheses when there is no confusion.

Example 1. Given the atomic sorts male, student, person and happy, we can give structured sorts as follows.

$$student \sqcap \overline{male},\ person \sqcup \sim happy.$$

[2] Gabbay and Hunter introduce the notation $\neg_\alpha \beta$ that means 'α negates β,' concerning the contradictory sorts [8].

The structured sort $student \sqcap \overline{male}$ means "students that are not male," and the structured sort $person \sqcup \sim happy$ means "individuals that are persons or unhappy."

We define a sorted signature on the set $\mathcal{S}^+$ of structured sorts. $\mathcal{F}_n$ is a set of n-ary function symbols $(f, f_0, f_1, \ldots)$, and $\mathcal{P}_n$ is a set of n-ary predicate symbols $(p, p_0, p_1, \ldots)$. Let $\mathcal{S} = \{s_1, \ldots, s_n\}$ be a set of atomic sorts. We introduce *the sort predicates* $p_{s_1}, \ldots, p_{s_n}$ (discussed in [3]) indexed by the sorts $s_1, \ldots, s_n$ where p_{s_i} is a unary predicate (i.e. $p_{s_i} \in \mathcal{P}_1$) and equivalent to the sort s_i. We simply write s for p_s when this will not cause confusion. For example, instead of the formula $p_s(t)$ where t is a term, we use the notation $s(t)$. We denote by $\mathcal{P}_\mathcal{S}$ the set $\{p_s \in \mathcal{P}_1 \mid s \in \mathcal{S} - \{\top, \bot\}\}$ of the sort predicates indexed by all sorts in $\mathcal{S} - \{\top, \bot\}$. A sorted signature extended to include structured sorts and sort predicates is defined as follows.

Definition 2 (Sorted signature on $\mathcal{S}^+$). *A sorted signature on $\mathcal{S}^+$, which we call a structured sort signature, is an ordered quadruple $\Sigma^+ = (\mathcal{S}^+, \mathcal{F}, \mathcal{P}, \Omega)$ satisfying the following conditions:*

(1) $\mathcal{S}^+$ is the set of all structured sorts constructed by $\mathcal{S}$.
(2) $\mathcal{F}$ is the set $\bigcup_{n\geq 0} \mathcal{F}_n$ of all function symbols.
(3) $\mathcal{P}$ is the set $\bigcup_{n\geq 0} \mathcal{P}_n$ of all predicate symbols.
(4) Ω is a set of sort declarations of functions and predicates such that:
  (i) If $f \in \mathcal{F}_n$, then $f\colon s_1 \times \ldots \times s_n \to s \in \Omega$ where $s_1, \ldots, s_n, s \in \mathcal{S} - \{\bot\}$. In particular, if $c \in \mathcal{F}_0$, then $c\colon \to s \in \Omega$.
  (ii) If $p \in \mathcal{P}_n$, then $p\colon s_1 \times \ldots \times s_n \in \Omega$ where $s_1, \ldots, s_n \in \mathcal{S} - \{\bot\}$. In particular, if $p_s \in \mathcal{P}_\mathcal{S}$, then $p_s\colon \top \in \Omega$.

Note that the sort declarations of functions and predicates are given by atomic sorts. The structured sort signatures do not include subsort declarations.

3.2 Sort-hierarchy declaration

We will build a sort-hierarchy over $\mathcal{S}^+$, instead of subsort declarations in sorted signatures of typical order-sorted logics. In our logic, we cannot enumerate all the subsort relations on $\mathcal{S}^+$ because the set of subsort declarations representing a subsort relation may be infinite. Hence, we first give a finite set of subsort declarations, so that the subsort relation should be derived by a sort constraint system. For this purpose, we deal with subsort declarations as subsort formulas but not as static expressions in signatures. Let $\Sigma^+ = (\mathcal{S}^+, \mathcal{F}, \mathcal{P}, \Omega)$ be a structured sort signature. For $s, s' \in \mathcal{S}^+$, $s \sqsubseteq_S s'$ is said to be a subsort declaration over Σ^+ that indicates s is a subsort of s'. For instance,

$$player \sqcap winner \sqsubseteq_S person$$

is a subsort declaration over a structured sort signature. We denote by $D_{\mathcal{S}^+} = \{s \sqsubseteq_S s' | s, s' \in \mathcal{S}^+\}$ the set of all subsort declarations on $\mathcal{S}^+$. In the next definition, the sort-hierarchy is obtained by a finite set of subsort declarations.

Definition 3 (Sort-hierarchy declaration). *A sort-hierarchy declaration is an ordered pair $H = (\mathcal{S}^+, D)$, where*

(1) $\mathcal{S}^+$ is the set of structured sorts constructed by $\mathcal{S}$,
(2) D is a finite set $\{s_1 \sqsubseteq_S s'_1, s_2 \sqsubseteq_S s'_2, \ldots\}$ of subsort declarations on $\mathcal{S}^+$.

Extended declarations on $\mathcal{S}^+$ are defined by subsort declarations as follows.

Definition 4. *A sort equivalence declaration, an exclusivity declaration and a totality declaration are defined respectively by*

- $s =_S s'$ *iff* $s \sqsubseteq_S s'$ *and* $s' \sqsubseteq_S s$.
- $s \parallel s'$ *iff* $(s \sqcap s') =_S \bot$.
- $s \mid_{s_i} s'$ *iff* $(s \sqcup s') =_S s_i$.

We use the abbreviation $s \mid s'$ to denote $s \mid_\top s'$. The above notations are useful for declaring complicated sort relations in a sort-hierarchy declaration $H = (\mathcal{S}^+, D)$.

Example 2. The sort-hierarchy declaration $H = (\mathcal{S}^+, D)$ consists of the set $\mathcal{S}^+$ of structured sorts constructed by

$$\mathcal{S} = \{person, winner, loser, player, \bot, \top\}$$

and the finite set D of subsort declarations with

$$\begin{aligned} D = \{&winner \sqsubseteq_S player,\ player \sqsubseteq_S person,\\ &loser \sqsubseteq_S player,\ winner \mid_{player} loser,\ winner \parallel loser\}. \end{aligned}$$

The sorts *winner* and *loser* are subsorts of *player*, and the sort *player* is a subsort of *person*. The totality declaration $winner \mid_{player} loser$ indicates that *winner* and *loser* have the property totality in *player*. The exclusivity declaration $winner \parallel loser$ indicates that *winner* and *loser* are mutually exclusive.

3.3 Structured sort constraint system

We develop a constraint system with respect to a subsort relation on $\mathcal{S}^+$.

Definition 5. *Let s, s', s'' be structured sorts. The axioms and rules of structured sort constraint system CS consist of:*

Reflexivity $s \sqsubseteq_S s$
Idempotency $s \sqsubseteq_S s \sqcap s, \quad s \sqcup s \sqsubseteq_S s$

Commutativity $s \sqcap s' \sqsubseteq_S s' \sqcap s, \quad s \sqcup s' \sqsubseteq_S s' \sqcup s$
Associativity $(s \sqcap s') \sqcap s'' =_S s \sqcap (s' \sqcap s''), \quad (s \sqcup s') \sqcup s'' =_S s \sqcup (s' \sqcup s'')$
Distributivity $(s \sqcup s') \sqcap s'' =_S (s \sqcap s'') \sqcup (s' \sqcap s''), \quad (s \sqcap s') \sqcup s'' =_S (s \sqcup s'') \sqcap (s' \sqcup s'')$

Least and greatest sorts $\bot \sqsubseteq_S s, \quad s \sqsubseteq_S \top$

Conjunction $s \sqcap s' \sqsubseteq_S s, \quad s \sqsubseteq_S s \sqcap \top$
Disjunction $s \sqsubseteq_S s \sqcup s', \quad s \sqcup \bot \sqsubseteq_S s$
Absorption $(s \sqcap s') \sqcup s \sqsubseteq_S s, \quad s \sqsubseteq_S (s \sqcup s') \sqcap s$

Classical negation $s \parallel \overline{s}, \quad s \mid \overline{s}, \quad \overline{s \sqcap s'} =_S \overline{s} \sqcup \overline{s'}, \quad \overline{s \sqcup s'} =_S \overline{s} \sqcap \overline{s'}$
Strong negation $s \parallel\!\sim s, \quad \sim s \sqsubseteq_S \overline{s}$

Transitivity rule $$\frac{s \sqsubseteq_S s' \quad s' \sqsubseteq_S s''}{s \sqsubseteq_S s''}$$

Introduction rule $$\frac{s \sqsubseteq_S s'}{s'' \sqcap s \sqsubseteq_S s'' \sqcap s'}$$

Elimination rule $$\frac{s \sqcup s' \sqsubseteq_S s \sqcup s'' \quad s \parallel s' \quad s \parallel s''}{s' \sqsubseteq_S s''} \ .$$

A derivation of an expression (a subsort declaration, or a clause which we will define) from a set of expressions is defined as follows.

Definition 6 (Derivation). *Let Δ a set of expressions. A derivation of F_n in a system X from Δ is a finite sequence $F_1, F_2, \ldots, F_n$ such that*

(i) $F_i \in \Delta$,
(ii) F_i *is an axiom of system* X, *or*
(iii) F_i *follows from* $F_j (j < i)$ *by one of the rules of system* X.

We write $\Delta \vdash_X F$ if F has a derivation from Δ in the system X. This notion of derivations can be used for the structured sort constraint system CS, and a clausal inference system which we will present.

3.4 Sorted terms and formulas with structured sort constraints

An alphabet for an order-sorted first-order language $\mathcal{L}_{\mathcal{S}^+}$ of structured sort signature Σ^+ contains the following: the set $\mathcal{V} = \bigcup_{s \in \mathcal{S} - \{\bot\}} \mathcal{V}_s$ of variables for all atomic sorts in $\mathcal{S} - \{\bot\}$ (where $\mathcal{V}_s$ is a set of variables $x_1 \colon s, x_2 \colon s, \ldots$ for atomic sort s), the connectives $\neg, \wedge, \vee, \rightarrow$, the quantifiers $\forall, \exists$, and the auxiliary parentheses and commas.

We give the expressions *sorted term* and *formula* for our order-sorted first-order language with structured sorts.

Definition 7 (Sorted terms). *Let $\Sigma^+ = (\mathcal{S}^+, \mathcal{F}, \mathcal{P}, \Omega)$ be an structured sort signature and let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration. The set $TERM_{\Sigma^+, s}$ of terms of sort s is defined by:*

(1) A variable $x \colon s$ is a term of sort s.
(2) A constant $c \colon s$ is a term of sort s where $c \in \mathcal{F}_0$ and $c \colon \rightarrow s \in \Omega$.

(3) If $t_1, \ldots, t_n$ are terms of sorts $s_1, \ldots, s_n$, then $f(t_1, \ldots, t_n)\colon s$ is a term of sort s where $f \in \mathcal{F}_n$ and $f\colon s_1 \times \ldots \times s_n \to s \in \Omega$.
(4) If t is a term of sort s' with $D \vdash_{CS} s' \sqsubseteq_S s$, then t is a term of sort s.

We denote by $TERM_{\Sigma^+}$ the set of all sorted terms $\bigcup_{s \in \mathcal{S}-\{\bot\}} TERM_{\Sigma^+,s}$.

We define a structured sort substitution with respect to a subsort relation derivable in the constraint system CS. That is, the subsort declarations are obtained by an application of the rules from CS so that the substitution is defined via the subsort declarations.

Definition 8 (Structured sort substitution). *A structured sort substitution is a function θ mapping from a finite set of variables to $TERM_{\Sigma^+}$ where $\theta(x\colon s) \neq x\colon s$ and $\theta(x\colon s) \in TERM_{\Sigma^+,s}$*[3].

In the above definition none of the terms of sort $\bot$ can be substituted for variables. If there do not exist subsorts s' of s such that $s' \neq \bot$, then the substitutions correspond to many-sorted substitutions (i.e. not order-sorted substitutions).

Definition 9 (Sorted formulas). *Let $\Sigma^+ = (\mathcal{S}^+, \mathcal{F}, \mathcal{P}, \Omega)$ be a structured sort signature and let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration. The set $FORM_{\Sigma^+}$ of sorted formulas is defined by:*

(1) If $t_1, \ldots, t_n$ are terms of $s_1, \ldots, s_n$, then $p(t_1, \ldots, t_n)$ is an atomic formula (or simply an atom) where $p \in \mathcal{P}_n$ and $p\colon s_1 \times \ldots \times s_n \in \Omega$,
(2) If A and B are formulas, then $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(\forall x\colon sA)$, and $(\exists x\colon sA)$ are formulas.

We introduce literals in order to represent formulas in clause form. A positive literal is an atomic formula $p(t_1, \ldots, t_n)$, and a negative literal is the negation $\neg p(t_1, \ldots, t_n)$ of an atomic formula. A literal is a positive or a negative literal.

Definition 10. *Let $L_1, \ldots, L_n$ be literals. The formula $L_1 \vee \ldots \vee L_n (n \geq 0)$ is said to be a clause. We denote by CL_{Σ^+} the set of all clauses.*

3.5 Σ^+-structure

As in the semantics of standard order-sorted logics, we consider a structure that consists of the universe and an interpretation over $\mathcal{S}^+ \cup \mathcal{F} \cup \mathcal{P}$ and satisfies the sort declarations of functions and predicates on $\mathcal{S}$. The interpretation of atomic sorts is defined by subsets of the universe. Hence, the interpretation of structured sorts is constructed by the interpretation of atomic sorts and the operations of set theory.

Definition 11. *Given a structured sort signature $\Sigma^+ = (\mathcal{S}^+, \mathcal{F}, \mathcal{P}, \Omega)$, a Σ^+-structure is an ordered pair $M^+ = (U, I^+)$ such that*

[3] In order to substitute variables with terms of the subsorts, the set $TERM_{\Sigma^+,s}$ of terms of sort s contain the terms of their subsorts obtained by subsort declarations that are derivable using a sort constraint system.

(1) U *is a non-empty set.*
(2) I^+ *is a function on* $\mathcal{S}^+ \cup \mathcal{F} \cup \mathcal{P}$ *where*
- $I^+(s) \subseteq U$ *(in particular,* $I^+(\top) = U$ *and* $I^+(\bot) = \emptyset$*),*
  $I^+(s \sqcap s') = I^+(s) \cap I^+(s')$,
  $I^+(s \sqcup s') = I^+(s) \cup I^+(s')$,
  $I^+(\overline{s}) = I^+(\top) - I^+(s)$,
  $I^+(\sim s) \subseteq I^+(\top) - I^+(s)$,
- $I^+(f)\colon I^+(s_1) \times \ldots \times I^+(s_n) \to I^+(s)$ *where* $f \in \mathcal{F}_n$ *and* $f\colon s_1 \times \ldots \times s_n \to s \in \Omega$,
- $I^+(p) \subseteq I^+(s_1) \times \ldots \times I^+(s_n)$ *where* $p \in \mathcal{P}_n$ *and* $p\colon s_1 \times \ldots \times s_n \in \Omega$ *(in particular,* $I^+(p_s) = I^+(s)$ *where* $p_s \in \mathcal{P}_{\mathcal{S}}$ *and* $p_s\colon \top \in \Omega$*).*

A variable assignment (or simply an assignment) in a Σ^+-structure $M^+ = (I^+, U)$ is a function $\alpha\colon \mathcal{V} \to U$ where $\alpha(x\colon s) \in I^+(s)$ for all variables $x\colon s \in \mathcal{V}$. Let α be an assignment in a Σ^+-structure $M^+ = (I^+, U)$, let $x\colon s$ be a variable in $\mathcal{V}$, and $d \in I^+(s)$. The assignment $\alpha[d/x\colon s]$ is defined by $\alpha[d/x\colon s] = (\alpha - \{(x\colon s, \alpha(x\colon s))\}) \cup \{(x\colon s, d)\}$.

We now define an interpretation over structured sort signatures Σ^+. If an interpretation $\mathcal{I}^+$ consists of a Σ^+-structure M^+ and an assignment α in M^+, then $\mathcal{I}^+$ is said to be a Σ^+-interpretation.

Definition 12. *Let* $\mathcal{I}^+ = (M^+, \alpha)$ *be a* Σ^+*-interpretation. The denotation* $[\![\ ]\!]_\alpha$ *is defined by*

(1) $[\![x\colon s]\!]_\alpha = \alpha(x\colon s)$,
(2) $[\![c\colon s]\!]_\alpha = I^+(c)$ *with* $I^+(c) \in I^+(s)$,
(3) $[\![f(t_1, \ldots, t_n)\colon s]\!]_\alpha = I^+(f)([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha)$.

We formalize a satisfiability relation indicating that a Σ^+-interpretation satisfies sorted formulas and subsort declarations.

Definition 13. *Let* $\mathcal{I}^+ = (M^+, \alpha)$ *be a* Σ^+*-interpretation and let* F *be a sorted formula or a subsort declaration. We define the satisfiability relation* $\mathcal{I} \models F$ *by the following rules:*

(1) $\mathcal{I}^+ \models p(t_1, \ldots, t_n)$ *iff* $([\![t_1]\!]_\alpha, \ldots, [\![t_n]\!]_\alpha) \in I^+(p)$,
(2) $\mathcal{I}^+ \models (\neg A)$ *iff* $\mathcal{I}^+ \not\models A$,
(3) $\mathcal{I}^+ \models (A \wedge B)$ *iff* $\mathcal{I}^+ \models A$ *and* $\mathcal{I}^+ \models B$,
(4) $\mathcal{I}^+ \models (A \vee B)$ *iff* $\mathcal{I}^+ \models A$ *or* $\mathcal{I}^+ \models B$,
(5) $\mathcal{I}^+ \models (A \to B)$ *iff* $\mathcal{I}^+ \not\models A$ *or* $\mathcal{I}^+ \models B$,
(6) $\mathcal{I}^+ \models (\forall x\colon s)A$ *iff for all* $d \in I^+(s)$, $\mathcal{I}^+[d/x\colon s] \models A$ *holds,*
(7) $\mathcal{I}^+ \models (\exists x\colon s)A$ *iff for some* $d \in I^+(s)$, $\mathcal{I}^+[d/x\colon s] \models A$ *holds,*
(8) $\mathcal{I}^+ \models s \sqsubseteq_S s'$ *iff* $I^+(s) \subseteq I^+(s')$.

Let F be a sorted formula or a subsort declaration and let $\Gamma \subseteq \mathit{FORM}_{\Sigma^+} \cup D_{\mathcal{S}^+}$. If $\mathcal{I}^+ \models F$, then $\mathcal{I}^+$ is said to be a Σ^+-model of F. We denote $\mathcal{I}^+ \models \Gamma$ if $\mathcal{I}^+ \models F$ for every $F \in \Gamma$. If $\mathcal{I}^+ \models \Gamma$, then $\mathcal{I}^+$ is said to be a Σ^+-model of Γ. If Γ has a Σ^+-model , then Γ is Σ^+-satisfiable. If Γ has no Σ^+-model , then Γ is

Σ^+-unsatisfiable. If every Σ^+-interpretation $\mathcal{I}^+$ is a Σ^+-model of F, then F is said to be Σ^+-valid. We write $\Gamma \models_{\Sigma^+} F$ (F is a consequence of Γ in the class of Σ^+-structures) if every Σ^+-model of Γ is a Σ^+-model of F ($\in FORM_{\Sigma^+} \cup D_{\mathcal{S}^+}$).

Let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration and Δ a set of clauses. In the clausal inference system we will present in the next section, their rules are applied to clauses in Δ (which expresses an assertional knowledge base), related to a subsort relation derivable from D. If $\mathcal{I}^+$ is a Σ^+-model of both D and Δ, then $\mathcal{I}^+$ is said to be a Σ^+-model of (D, Δ), denoted by $\mathcal{I}^+ \models (D, \Delta)$. We write $(D, \Delta) \models_{\Sigma^+} F$ (F is a consequence of (D, Δ) in the class of Σ^+-structures) if every Σ^+-model of (D, Δ) is a Σ^+-model of F ($\in FORM_{\Sigma^+} \cup D_{\mathcal{S}^+}$).

4 Resolution with structured sorts

In addition to structured sort constraint system CS, we design a (clausal) resolution system with structured sorts. We adopt the method (proposed in [3]) of coupling a clausal knowledge base [13, 11] and a sort-hierarchy in which every sort can be used to express the sort predicate which is included in clauses. Then, we define a hybrid inference system in order to combine the two systems.

4.1 Clausal inference system with sort predicates

We present a clausal inference system, in which clauses may include sort predicates, e.g., $p(t_1, t_2) \vee s(t)$ where s is a sort predicate.

Definition 14 (Cut rule). *Let L, L' be positive literals and C, C' clauses.*

$$\frac{\neg L \vee C \quad L' \vee C'}{(C \vee C')\theta}$$

where there exists a mgu θ for L and L'.

The cut rule is one of the usual rules included in clausal inference systems. In addition to the cut rule, our clausal inference system have to include inference rules of sort predicates related to subsort declarations. We introduce the inference rules for resolution as follows.

Definition 15 (Resolution rules with sort predicates). *Let s, s', s_i be structured sorts or sort predicates, L, L' positive literals, t, t' sorted terms, and C, C' clauses. Resolution rules with sort predicates are given as follows.*

Subsort rule

$$\frac{\neg s(t) \vee C \quad s'(t') \vee C' \quad s' \sqsubseteq_S s}{(C \vee C')\theta}$$

where there exists a mgu θ for t and t'.

Sort predicate rule [4]

$$\frac{\neg s(t) \vee C \quad s' \sqsubseteq_S s}{C}$$

where $t \in TERM_{\Sigma^+,s'}$.

Exclusivity rule

$$\frac{s(t) \vee C \quad s'(t') \vee C' \quad s \parallel s'}{(C \vee C')\theta}$$

where there exists a mgu θ *for* t *and* t'.

Totality rule

$$\frac{s_i(t) \vee C \quad \neg s(t') \vee C' \quad s \mid_{s_i} s'}{(s'(t) \vee C \vee C')\theta}$$

where there exists a mgu θ *for* t *and* t'.

In particular, the exclusivity rule and the totality rule are useful for resolutions with respect to implicit negations embedded in a sort-hierarchy. The exclusivity rule will be applied when an opposite sort is declared as $s \parallel s'$. We write *resolution system MS* for the system defined by the cut rule in Definition 14 and the resolution rules in Definition 15.

4.2 Hybrid inference system with clauses and structured sort constraints

We define a hybrid inference system obtained by combining a clausal inference system with a sort constraint system. The inference rules in the hybrid system are applied to subsort declarations and clauses including sort predicates, so that they can deal with sort-hierarchy information in an assertional knowledge base.

Definition 16 (Hybrid inference system). *A hybrid inference system is a system obtained by adding the axioms and rules in a constraint system into a clausal inference system. We write* $X+Y$ *for the hybrid inference system obtained from a clausal inference system* X *and a constraint system* Y.

The hybrid inference system $X+Y$ can be regarded as an extension of the clausal inference system X. We write $(D, \Delta) \vdash_{X+Y} F$ to denote $D \cup \Delta \vdash_{X+Y} F$.

Lemma 1. *The axioms of the structured sort constraint system* CS *are* Σ^+*-valid.*

Lemma 2. *Let* $F, F_1, \ldots, F_n$ *be subsort declarations. The conclusion* F *of each rule in the structured sort constraint system* CS *is a consequence of its premise* $\{F_1, \ldots, F_n\}$ *in the class of* Σ^+*-structures. That is,* $\{F_1, \ldots, F_n\} \models_{\Sigma^+} F$.

[4] Instead of the sort predicate rule, the subsort rule can derive the same results by adding valid atoms with sort predicates.

Proof. Elimination rule: Suppose that $I^+(s) \cup I^+(s') = I^+(s) \cup I^+(s'')$, $I^+(s) \cap I^+(s') = \emptyset$, and $I^+(s) \cap I^+(s'') = \emptyset$. Let $d \in I^+(s')$. Since $I^+(s') \subseteq I^+(s) \cup I^+(s') \subseteq I^+(s) \cup I^+(s'')$, we have $d \in I^+(s) \cup I^+(s'')$. $d \in I(s')$ and $I^+(s) \cap I^+(s') = \emptyset$ imply $d \notin I^+(s)$. Therefore $d \in I^+(s'')$. Similarly, the other rules can be proved. ∎

Lemma 3. *Let $F, F_1, \ldots, F_n$ be clauses or subsort declarations. The conclusion F of each rule in the resolution system RS is a consequence of its premise $\{F_1, \ldots, F_n\}$ in the class of Σ^+-structures. That is, $\{F_1, \ldots, F_n\} \models_{\Sigma^+} F$.*

Proof. For each rule we show $\{F_1, \ldots, F_n\} \models_{\Sigma^+} F$.

1. Exclusivity rule: Suppose that $\mathcal{I}^+ \models s(t) \vee C$, $\mathcal{I}^+ \models s'(t') \vee C'$, and $I^+(s) \cap I^+(s') = \emptyset$. Let θ be a structured sort substitution such that $\theta(t) = \theta(t')$. So $\mathcal{I}^+ \models (s(t) \vee C)\theta$ and $\mathcal{I}^+ \models (s'(t') \vee C')\theta$. By $I^+(s) \cap I^+(s') = \emptyset$, either $\mathcal{I}^+ \models s(t)\theta$ or $\mathcal{I}^+ \models s'(t')\theta$ does not hold. By the hypothesis, $\mathcal{I}^+ \models C\theta$ or $\mathcal{I}^+ \models C'\theta$. Therefore $\mathcal{I}^+ \models C\theta \vee C'\theta$
2. Totality rule: Assume that $\mathcal{I}^+ \models s_i(t) \vee C$, $\mathcal{I}^+ \models \overline{s'}(t') \vee C'$, and $\mathcal{I}^+ \models s \mid_{s_i} s'$, i.e. $I^+(s) \cup I^+(s') = I^+(s_i)$. Let θ be a structured sort substitution such that $\theta(t) = \theta(t')$. If $I^+(s) \cup I^+(s') = I^+(s_i)$, then $\mathcal{I}^+ \models s(t) \vee s'(t') \vee C$. Then, we can obtain $\mathcal{I}^+ \models s'(t)\theta \vee C\theta \vee C'\theta$. Therefore the conclusion is a consequence of its premise. ∎

The next theorem shows the soundness of the structured sort constraint system CS and the resolution system RS.

Theorem 1. *Let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration, Δ a set of clauses, and X a system. If $(D, \Delta) \vdash_X F$, then $(D, \Delta) \models_{\Sigma^+} F$.*

Proof. By Lemma 1, 2, and 3, this is proved. ∎

We give the notion of contradiction in an exclusivity relation from the sort-hierarchy. This notion is defined by deciding whether there is a contradiction between an opposite sort and its antonymous sort.

Definition 17. *Let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration, Δ a set of clauses, and X a system. (D, Δ) is said to be contradictory on an exclusivity relation if there exists sorts s, s' such that $(D, \Delta) \vdash_X s \parallel s'$ and $(D, \Delta) \vdash_X s(t)$ and $(D, \Delta) \vdash_X s'(t)$. (D, Δ) is said to be logically contradictory if $(D, \Delta) \vdash_X A$ and $(D, \Delta) \vdash_X \neg A$.*

The contradiction between A and $\neg A$ (corresponding to "logically contradictory" in the above definition) is defined in the usual manner of logics. We say that (D, Δ) is consistent if (D, Δ) is neither contradictory on an exclusivity relation nor logically contradictory.

Theorem 2. *Let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration and Δ a set of clauses. If (D, Δ) has a Σ^+-model, then (D, Δ) is consistent.*

Proof. Suppose that $\mathcal{I}^+$ is a Σ^+-model of (D, Δ). If (D, Δ) is contradictory on an exclusivity relation, then there exists s, s' such that $(D, \Delta) \vdash_X s \parallel s'$ and $(D, \Delta) \vdash_X s(t)$ and $(D, \Delta) \vdash_X s'(t)$. By Theorem 1, $\mathcal{I}^+ \models s \parallel s'$ and then $\mathcal{I}^+ \models s(t)$ and $\mathcal{I}^+ \models s'(t)$. Then $I^+(s) \cap I^+(s') = \emptyset$ but $[\![t]\!]_\alpha \in I^+(s)$ and $[\![t]\!]_\alpha \in I^+(s')$. If (D, Δ) is logically contradictory, then $\mathcal{I}^+ \models \neg A$ and $\mathcal{I}^+ \models A$. Hence, the both cases are contradiction to the hypothesis. Therefore (D, Δ) is consistent. ∎

A refutation is a derivation of the empty clause (denoted $\square$) from (D, Δ), written as $(D, \Delta) \vdash_X \square$. The next corollary guarantees that the hybrid inference system $CS + RS$ is sound.

Corollary 1. *Let $H = (\mathcal{S}^+, D)$ be a sort-hierarchy declaration and Δ a set of clauses. If $(D, \Delta) \vdash_{CS+RS} \square$, then $(D, \Delta) \models_{\Sigma^+} \square$.*

Proof. When the empty clause $\square$ is derived, the final rule applied in the refutation must be one of the rules in the resolution system RS. We consider each case as follows:

1. Cut rule: There exists a structured sort substitution θ such that $L\theta = L'\theta$, and $(D, \Delta) \vdash_{CS+RS} \neg L$ and $(D, \Delta) \vdash_{CS+RS} L'$. So, by Theorem 1, we have $(D, \Delta) \models_{\Sigma^+} \neg L$ and $(D, \Delta) \models_{\Sigma^+} L'$. Now assume that $\mathcal{I}^+$ is a Σ^+-model of (D, Δ). Then $\mathcal{I}^+ \models L\theta$ and $\mathcal{I}^+ \not\models L'\theta (= L\theta)$ contradicts our assumption. Since (D, Δ) has no Σ^+-model, $(D, \Delta) \models_{\Sigma^+} \square$ is proved.
2. Resolution rules: Similar to 1. ∎

5 Conclusions

This paper has presented an order-sorted logic that can deal with implicit negations in a sort hierarchy. We have presented a hybrid inference system that consists of a clausal inference system and a structured sort constraint system. This system includes structured sort expressions composed of atomic sorts, connectives, and negative operators, in order to deal with implicitly negative sorts embedded in a sort-hierarchy. To represent these negative sorts, we have proposed the notions of sort relations (subsort relation, equivalence relation, exclusivity relation, and totality relation) on the structured sorts, and we have axiomatized the properties of implicitly negative sorts. Thus, the structured sort constraint system can derive relationships between classical negation, strong negation, and antonyms in a sort-hierarchy. Furthermore, the contradiction in the sort-hierarchy as defined by the exclusivity relation enables us to prove the soundness of our logic with structured sorts.

We need to improve our hybrid inference system in order to tackle implementation issues caused by the complicated sort expressions. As a work which remains theoretical, the complete system must be given by revising the axioms and rules.

References

1. Baader, F., & Hanschke, P. (1991). A scheme for integrating concrete domains into concept languages. *Pages 452–457 of: Twelfth international conference on artificial intelligence*.
2. Baader, F., & Sattker, U. (1999). Expressive number restrictions in description logics. *Journal of logic and computation*, **9**(3), 319–350.
3. Beierle, C., Hedtsuck, U., Pletat, U., Schmitt, P.H., & Siekmann, J. (1992). An order-sorted logic for knowledge representation systems. *Artificial intelligence*, **55**, 149–191.
4. Cohn, A. G. (1987). A more expressive formulation of many sorted logic. *Journal of automated reasoning*, **3**, 113–200.
5. Cohn, A. G. (1989). Taxonomic reasoning with many sorted logics. *Artificial intelligence review*, **3**, 89–128.
6. Donini, F. D., Lenzerini, M., Nardi, D., & Schaerf, A. (1996). Reasoning in description logic. Brewka, G. (ed), *Principles of knowledge representation*. CSLI Publications, FoLLI.
7. Frisch, Alan M. (1991). The substitutional framework for sorted deduction: fundamental results on hybrid reasoning. *Artificial intelligence*, **49**, 161–198.
8. Gabbay, D., & Hunter, A. (1999). Negation and contradiction. Gabbay, D. M., & Wansing, H. (eds), *What is negation?* Kluwer Academic Publishers.
9. Gallier, Jean H. (1986). *Logic for computer science. foundations of automatic theorem proving*. Harper & Row.
10. Horrocks, I. (1999). A description logic with transitive and inverse roles and role hierarchies. *Journal of logic and computation*, **9**(3), 385–410.
11. Lobo, J., Minker, J., & Rajasekar, A. (1992). *Foundations of disjunctive logic programming*. The MIT Press.
12. Ota, A. (1980). *Hitei no imi (in Japanese)*. Taishukan.
13. Richards, T. (1989). *Clausal form logic. an introduction to the logic of computer reasoning*. Addison-Wesley Publishing Company.
14. Schmidt-Schauss, M. (1989). *Computational aspects of an order-sorted logic with term declarations*. Springer-Verlag.
15. Schmidt-Schauss, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial intelligence*, **48**, 1–26.
16. Smolka, G. (1992). Feature-constraint logics for unification grammars. *Journal of logic programming*, **12**, 51–87.
17. Wagner, G. (1991). Logic programming with strong negation and inexact predicates. *Journal of logic computation*, **1**(6), 835–859.
18. Walther, C. (1987). *A many-sorted calculus based on resolution and paramodulation*. Pitman and Kaufman Publishers.
19. Weibel, T. (1997). An order-sorted resolution in theory and practice. *Theoretical computer science*, **185**(2), 393–410.