A Sorted Generalization for Legal Reasoning

Makoto HARAGUCHI and Tokuyasu KAKUTA

Department of Systems Science, Tokyo Institute of Technology
E-mail: makoto@sys.titech.ac.jp, kaku@sys.titech.ac.jp

1 Introduction

This paper presents a reasoning system that performs analogical reasoning for legal rules under an order-sorted representation. Analogy is often used in the domain of law to derive an appropriate conclusion for a particular case not covered by any legal rule. For such a case, lawyers are considered to find a legal rule whose requirement is similar to the case with respect to some significant points. Then the rule is analogically applied to the case for which the same or a similar conclusion of the rule is derived.

To develop a system for reasoning by analogy in law, it is necessary to have knowledge for deciding the similarities at a conceptual level. This is because both legal rules and cases are described in terms of legal concepts. It is well known that a taxonomic hierarchy on legal concepts is useful for this purpose. Those concepts with the same super concept in the hierarchy can be understood as similar ones. From a logical point of view, such a hierarchy is represented as an ordered set of sort symbols denoting concepts. Then our legal rules will be represented as well-sorted formulas of an order-sorted logic with the sort hierarchy.

The purpose of this paper is first to show that the order-sorted logic [9, 11] is more appropriate in representing legal rules than a standard unsorted logic. Especially a new symbol system is introduced to consider a predicate that takes another predicate instances as its arguments. Such a predicate frequently appears in legal texts, as shown in Section 2 and 3. A similar symbol system based on unsorted Horn logic can be found in [6] in which predicates have their identifiers to be referred by other predicates. Since the notion of identifiers goes beyond the scope of semantics of first order logic, it is hard to understand what the identifiers mean. On the other hand, it is not a difficult task to have a formal semantics based on sorted Herbrand models, although the present paper does not concern such a model theory. In addition to the model theoretic aspect, it is widely known that the sorted logic or a typed system can well control the inference processes because of its type-checking mechanism. The system presented in this paper also enjoys this property.

The second purpose of this paper is to show a computational method for analogical reasoning for legal rules under the order-sorted representation of legal knowledge. Although we can find many studies (see [18, 16, 17, 20] for instance) on computational analogy, few studies which emphasize the importance of sorted representations are found. In [19], an idea of using a type theory for analogical reasoning is described. However the details of how we make use of the typed knowledge in analogical reasoning and what operations are necessary to realize the idea are not fully developed. This paper, on the other hand, tries to present a set of concrete computational operators for doing that for legal rules based on our order-sorted representations.

The analogical reasoning in this paper can be viewed as a combined process of both deduction and generalization along the sort hierarchy. We classify the generalization into the followings: extension of the applicability of legal rules, generalization of predicates, and abstraction of individual objects or acts. These three types of generalization are carried out by the corresponding three generalization rules: sorted generalization, predicate generalization and term generalization. All of these are applied to goal clauses of a standard deductive interpreter for order-sorted Horn logic.

At first, the sorted generalization replaces a variable appeared in a rule to another variable of more general sort to eliminate type errors occurred in applying the rule analogically. Although the applicability of rules is restricted to sorts of variables, the sorts are now extended by the generalization. Thus, our sorted generalization produces a rule with wider applicability than the original one. Moreover, it is easy to compute the generalization and therefore to apply the rule analogically, since the sorted generalization is determined by a simple algebraic operation for our sort hierarchy. Compared with the previous study [5] on analogy in law, which concerns the same legal case as this paper does, a great computational reduction is achieved due to the algebraic operation.

The term generalization, the second generalization rule, abstracts an individual instance of a sort to a variable of more general sort. The variable is then instantiated to another instance of a more specific sort. The latter instance is treated as a candidate for an analogue of the original one. This is because their sorts share

the generalized sort as their common super sort. Thus, the term generalization together with the instantiation is used to find an analogue.

Finally, the predicate generalization has an effect of replacing a predicate with another one of more general extension. In contrast with the first two generalization rules, it does not depend on our sort hierarchy. A logical structure of (object-level) rules completely determines the generalization. Technically speaking, the generalization of this type is necessary to eliminate some type errors invoked by term generalization.

In the research field of machine learning, we can find many studies on generalization. Especially the notion of absorption, defined in [4, 13], becomes important also in this paper to realize the generalizations along the legal taxonomy. Our sorted generalization is a natural extension of the absorption. The basic idea thus comes from [4, 13]. Similarly both the predicate generalization and the term generalization can be kinds of absorption, if we translate the sorted representations into unsorted ones. However, the process of controlling how we apply the absorption is strongly constrained due to the sort information given by our sort hierarchy, as we will see in Section 5. This point distinguishes this paper from the others.

We organize this paper as follows: In Section 2, a preliminary definition of order-sorted logic is described, and then a symbol system for representing legal rules is presented based on the order-sorted logic. In Section 3, a legal rule appeared in Japanese Civil Code and a real case for which the rule has been applied analogically are presented and analyzed. All the necessary knowledge to realize the analogy is shown and is also represented as sorted clauses. Section 4 is devoted for presenting a general explanation of what is our sorted generalization. This section is just technical, and does not concern legal aspects. In Section 5, the three generalization rules as well as some control rules are presented. Some examples to show why the rules are needed and how they behave are described. In the final section, some future works are discussed.

2 An order-sorted representation of legal rules

We first present preliminary definitions of order-sorted logic, and then introduce a new symbol system under which legal rules are represented.

2.1 Preliminaries

It is necessary to have a large taxonomic hierarchy to describe legal knowledge. The hierarchy consists of legal conceptual classes linked with *ISA* relations [1]. According to a standard logic, each *ISA* link is represented as a definite clause. For instance, a clause meaning that s_1 is a subclass of s_2 can be written as:

$$\forall x.\ \ s_1(x) \rightarrow s_2(x),$$

where $s_j(x)$ is true iff x is an instance of the conceptual class which s_j denotes. Some studies [10, 11, 9] have already pointed that it is unnatural to have *ISA* relations in the forms of logical rules as in the above. For instance, suppose we have another clause expressing that s_2 is a subclass of s_3. In order to conclude a fact that s_1 is also a subclass of s_3, we must apply inference rules like Modus Ponens. However the fact is a direct consequence of the transitive law of partial ordering, provided we have the taxonomic hierarchy in the form of partially ordered set. From this simple observation, each conceptual class and the taxonomic hierarchy are now formalized as a sort symbol and a partially ordered set $(S, \leq)$ of sort symbols, respectively ([11],[9]). In what follows, the statement that s_1 is a subclass of s_2 is denoted by $s_1 \leq s_2$.

Each non-logical symbol is supposed to have a corresponding sorted specification. A function symbol f of arity n has its sorted specification written as $f : s_1, ..., s_n \rightarrow s$, which means that the function (denoted by) f takes instances of $s_1, ..., s_n$ and returns an instance of s. Similarly, a predicate symbol p of arity n has its sorted specification $p : s_1, ..., s_n$ meaning that the predicate p is defined for n-tuples whose j-th arguments are instances of s_j. Moreover any variable x is assumed to have its sort s, and is written as $x : s$. The sort s specifies the range of possible instances for the variable x to have.

Given these sorted specification of symbols, $[t]$ of a term t, called a principal sort of t, is defined by:

(1) If t is a variable $x : s$ then $[t] = s$.

(2) If $t = f(t_1, ..., t_n)$ for some $f : s_1, ..., s_n \rightarrow s$ $(n \geq 0)$, then $[t] = s$.

Moreover, a first order term t is said to be well-sorted if, for any subterm $f(t_1, ..., t_n)$ with a specification $f : s_1, ..., s_n \rightarrow s$, $[t_j] \leq s_j$ holds for each j. An expression $t : s$ denotes that t is an well-sorted term such that $[t] \leq s$. Similarly a substitution $\theta = \{x_j/t_j\}$, which replaces the variable x_j with the term t_j, is called well-sorted, if t_j is well-sorted and $[t_j] \leq [x_j]$. Thus, by the well-sorted substitution, possible instances of a variable $x : s$ are restricted to well-sorted terms of sorts s' such that $s' \leq s$.

Given a sort hierarchy $(S, \leq)$, a unification that takes the sorted information into account is called an order-sorted unification. Intuitively speaking, the order-sorted unification matches two or more well-sorted

[1] Although there exist several ways of explaining *ISA* relation [12], this paper interprets it as "is_a_subclass_of".

terms and forms them into a single well-sorted term, where the matching is done by applying well-sorted substitutions. For instance, suppose we have an ordering in which $s \leq s_1$ and $s \leq s_2$ hold for sorts s, s_1, and s_2. Then the terms $x_1 : s_1$ and $x_2 : s_2$ can be unifiable, since they become the same well-sorted term $x : s$ by the well-sorted substitution $\{x_1/x, x_2/x\}$. On the other hand, if s_1 and s_2 has no common subsort, then the order-sorted unification fails.

Formally a unifier τ for a set E of well-sorted terms is a substitution such that $e_1\tau = e_2\tau$ for all $e_1, e_2 \in E$. An order-sorted unifier for a set of well-sorted terms is defined as a unifier of the set that is well-sorted. Furthermore, for order-sorted unifiers θ_1 and θ_2 of a set E of well-sorted terms, an ordering $\theta_1 \leq \theta_2$ is defined as:

> $\theta_1 \leq \theta_2$ iff there exists an well-sorted substitution τ such that $x\theta_1\tau = x\theta_2$ for any variable x in E

In this case, we say that the substitution θ_1 is more general than θ_2. It is well known that, for a set of well-sorted terms that have an order-sorted unifier, there always exists a maximally general order-sorted unifier (mgosu, for short), provided our sort hierarchy is finite ([11]). An order-sorted resolution is now defined as the standard resolution for well-sorted clauses, where all the unifiers should be mgosu. For more details, see [9] for instance.

2.2 A symbol system

Legal rules generally govern human's individual acts and events in our social life. Their effectiveness and the validness depend on what properties they have, what legal concepts they belong to, and what relations hold between them. Since our order-sorted logic is basically a first order logic, the properties and the relations should be formalized as first order predicates that take individual acts as their arguments. By the same reason, once we decide to adopt order-sorted representations, the individual acts and their legal conceptual classes must be expressed as well-sorted terms and their sorts, respectively. Thus, they are distinguished at symbol level. In most legal texts, however, the individual acts and their classes are sometimes confused at symbol level. For instance, consider the following sentence in the form of natural language:

> a contract that is made by a person a with a person b for an object c is a contract.

The first occurrence of the term "contract" given an indefinite article denotes an individual contract whose attributes are specified by the phrase followed by the relative pronoun. Thus, it works as a name for some individual act. On the other hand, "is_a" is a copula that connects the individual contract and a class named "contract". The second occurrence of "contract" therefore works as a general name. A symbol system defined now in this section distinguishes each individual name from a general name. For this purpose, a sort symbol and a functor are associated with a legal conceptual class on actions. Formally we need the following definition.

Definition 1 (Sorts of Event Type) *We first assume that we have a designated sort symbol event in our set S of sorts. Each sort s more specific than event is called a sort of event type. Otherwise, s is called a sort of object type* [2]. *We assume the greatest sort object among sorts of object type. A functor f whose codomain sort s is event type is called an event formation functor for s. If there exists the unique event formation functor f for s, f is rather denoted by s_f.*

For instance, the sort *contract* ($\leq$ *event*) is supposed to have the unique event formation functor

$$contract\_f : person, person, object \rightarrow contract \ ,$$

meaning that an instance of *contract* is determined by specifying two persons and an object for the contract. Then, given constants $a : person$, $b : person$ and $imm\_X : object$, the sentence we have examined in the preceding page corresponds to the following well-sorted term:

$$contract\_f(a, b, imm\_X) : contract.$$

The arguments a, b and imm_X work as "attributes" or "features" of that term of *contract*. To get the attributes from a term of event type, we consider the following "attr"-predicates. For each event formation functor $f : s_1, ..., s_n \rightarrow s$, we assume a unit clause

$$attr(f(X_1 : s_1, ..., X_n : s_n), a_j, X_j) \ ,$$

where a_j is an attribute name [3] (or a feature name) associated with the j-th domain sort s_j of f. For instance, using an attribute name $agt2$, one of the attr-predicates for $contract\_f$ can be written as:

$$attr(contract\_f(X : person,\ Y : person,\ Z : object), agt2, Y),$$

meaning that the second agent of $contract\_f(X, Y, Z)$ is Y. In what follows, the sorted specification for an event formation functor $f : s_1, ..., s_n \rightarrow s$ with its attribute names $a_1, ..., a_n$ is rather written as:

$$f(a_1 : s_1,\ \ ...,\ \ a_n : s_n) : s \ .$$

[2] Precisely speaking, s is called a sort of object type, if $\neg(s \leq event)$ and $s \neq label$.

[3] The attribute names are constant symbols of a designated sort *label*. *label* is assumed to be incomparable to any other sort in our sort hierarchy. Moreover it can be neither a domain sort nor a co-domain sort of a functor as well as a predicate except *attr*.

We understand this expression as a statement that a_j of f is s_j and that f forms an instance of s. The sorted specification for $contract\_f$ can be now written as:

$$contract\_f(agt1 : person, agt2 : person, \\ obj : object) : contract.$$

In contrast with the usual first order terms, any term of event type is a theoretical being, and is not assumed to always have a real denotation. In order to designate something specified by such a term really happens, we introduce a special predicate $oc : event$ [4]. For instance, an atomic formula $oc(contract\_f(a, b, c))$ means that the contract made by a and b for c really occurs in a world we are going to axiomatize.

A query is an well-sorted goal clause. $\leftarrow oc(X : lawful\_act)$ is an example meaning the question "Does some lawful act X occur?". Our order-sorted resolution directly solves this goal by finding an order-sorted unifier

$$\theta = \{X : lawful\_act/contract\_f(a, b, c)\},$$

provided our sort hierarchy includes

$$contract \leq juristic\_act \leq lawful\_act \\ \leq human\_act \leq event.$$

3 Analogical reasoning for legal rules

Now we are ready to show a legal rule, Clause 2, Article 94 of Japanese Civil Code, which judges have applied analogically for many cases. For Article 94 is normally considered to concern the difference between external appearances and real intentions of contracts, we describe the article in a simplified form so that we can understand how the article is applied to resolve conflicts on contracts [5].

Clause 1: A contract with a false declaration of intention made by a party to the contract is null and void.

Clause 2: From the nullity of the contract in the preceding clause, one cannot set up against a person in good faith, who does not know the falsity that the declared intention is different from the real intention.

According to Clause 1, the contract with the false declaration of intention is null for the party to the contract. However the nullity is not applied to the other person, provided he believes the false declaration without knowing its real intention. The right of the person will be protected by Clause 2.

[4] The use of *oc* predicate is introduced in [12] for an unsorted case to analyze higher-order ISA relations.

[5] The original article is stated more abstractly, and requires a contract to be made in collusion with its parties.

The Clause2, for instance, can be encoded in the form of order-sorted representation followed by its sorted specifications:

Rule 1 (Article94, Clause2)

$oc(Ctrct1 : contract),$
$\quad attr(Ctrct1, agt1, X), attr(Ctrct1, agt2, Y)$
$\quad attr(Ctrct1, obj, Object)$
$oc(Falsity : falsity), attr(Falsity, obj, Ctrct1),$
$oc(Ctrct2 : contract), attr(Ctrct2, agt1, Y),$
$\quad attr(Ctrct2, agt2, Z), attr(Ctrct2, obj, Object),$
$good\_faith(Z, Falsity)$
$\quad \rightarrow cannot\_set\_up(X, Z, Ctrct2)$

Sorted Specifications:
$contract\_f(agt1 : person, agt2 : person,$
$\qquad obj : object) : contract.$
$contract \leq juristic\_act \leq lawful\_act$
$\qquad \leq human\_act \leq event.$
$good\_faith : person, event. \qquad person \leq object.$
$cannot\_set\_up : person, person, contract.$
$falsity\_f(obj : human\_act) : falsity \leq event.$

Rule 1 is concerned with two contracts, *Ctrct1* and *Ctrct2*. The variable $Falsity$ will be bound to the falsity of the first contract $Ctrct1$, as we see in the next section. The atom *cannot_set_up(X,Z,Ctrct2)* means that X cannot claim the invalidity of $Ctrct2$ which Z made.

The meaning of falsity is clarified by the theory interpretation rule 2, which states that a contract is concluded as false if the representation and the state of affairs are different.

Rule 2 (Falsity of Contracts)

$oc(Ctrct : contract)$
$repr\_of\_ctrct(Ctrct, ReprCts),$
$soa\_of\_ctrct(Ctrct, RealCts)),$
$ReprCts \neq RealCts$
$\quad \rightarrow oc(falsity\_f(Ctrct)).$
Sorted Specifications:
$\quad (repr\_of\_ctrct : contract, top.),$
$\quad\quad (soa\_of\_ctrct : contract, top.)$
Abbreviations:
$\quad$ *"repr" : representation, "soa": state of affairs.*

The predicate *good_faith* is defined as follows:

$(oc(Evt) - > not(know(Agt, Evt)))$
$\quad \rightarrow good\_faith(Agt : person, Evt : event)$

where *not* and $A - > B$ are a negation defined by negation as failure rule and a built-in prediate meaning "if A then B", respectively. The predicate $know : person, top$ [6] is defined so that an atom $know(Agt, Evt)$ succeeds iff the Evt, which will be bound to a term of event type, is proved under the set of facts Agt knows. We use an auxiliary predicate $know\_fact : person, top$. All the possible instances of $know\_fact$ as well as rules for inferring "knowing" relations are initially presented in our fact database, as shown in Fact 1. According to such a representation of facts, the predicate *know* is easily realized as a kind of meta-interpreter of Prolog.

[6] *top* is the least upper bound of *event* and *object*.

Now we present a legal case for which Clause 2 of Article 94 is applied analogically. In the case, since p_b sold the house to p_c without its real ownership, p_a, who was the real owner, claimed his ownership right. However, the judge approved that p_c had the real ownership right by legal analogy.

Case of a petition against registration of passage of a house's title ((O)No.107-1951,judgment of the second pretty bench, Aug. 20th 1951))

Case: After p_a bought a house, which p_o owned, from p_o, he approved the registration of passage of title from p_o to p_b without his real intention of the passage. Registered the passage of title, p_b sold the house to a good faith, p_c, who did not know the real intention of p_a and only know that p_b registered. p_c registered the passage of ownership title. p_a claimed that p_c must do cancellation procedure of passage of title and others because the ownership of the house in the case should belong to p_a, and p_b and p_c did not have the ownership of the house.

Judgment: By analogical application of Clause2, Art. 94 of Japanese Civil Code, the nullity of the registration of 'B" cannot be set up against the good faith p_c, who did not know the real intention of p_a, so that p_a cannot claim that p_c must do cancellation procedure of passage of title and others of the registration.

We represent the legal case by a set of facts approved by the court.

Fact 1 (Facts representing the legal case)

Relations:

$oc(sale\_of\_immovables\_f(p\_o, p\_a, imm\_X))$.
$ownership(p\_a, imm\_X)$.
$oc(reg\_of\_ptitle\_f(p\_o, p\_b, imm\_X))$.
$oc(reg\_f(p\_b, imm\_X))$.
$oc(approval\_f(p\_a, reg\_of\_ptitle\_f(p\_o, p\_b, imm\_X)))$.
$oc(sale\_of\_immovables\_f(p\_b, p\_c, imm\_X))$.
$oc(reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X))$.
$know\_fact(p\_c, reg\_of\_ptitle\_f(p\_o, p\_b, imm\_X))$.
$know\_fact(p\_c, reg\_f(p\_b, imm\_X))$.
$know\_fact(p\_c, sale\_of\_immovables\_f(p\_b, p\_c, imm\_X))$.
$know\_fact(p\_c, reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X))$.)

Sorted Specifications:

p_o, p_a, p_b, p_c : $person$. imm_X : $house$.
$ownership$: $person, object$.
$reg\_of\_ptitle\_f(agt1 : person, agt2 : person, obj : imm\_prop) : reg\_of\_ptitle$
$house \leq imm\_prop \leq prop \leq object$
$reg\_f(agt : person, obj : imm\_proprop) : reg$
$reg\_of\_ptitle \leq reg \leq quarsi\_juritstic\_act \leq lawful\_act$
$sale\_of\_imm \leq sale \leq human\_act$
$approval\_f(agt : person, obj : human\_act) : approval \leq event$
$sale\_of\_imm\_f(agt1 : person, agt2 : person, obj : imm\_prop) : sale\_of\_imm$

Abbreviations:

agt: agent *prop: property* *reg: registration*
imm_prop: immovable property *ptitle : passage of titles*
prop: property
sale_of_imm: sale of immovable properties
reg_of_ptitle: registration of passage of titles

Rule 1 and the case represented by Fact 1 concern contracts and registrations, respectively. The judges thus seems to have applied a rule on contract to a case of registration. Moreover, the notion of falsity that Article 94 mentions is restricted to the falsity of contracts, as defined in Rule 2. Hence, for the analogy the judges used, the following conceptual operation might have been performed:

(1) The rule about contract is generalized so that it can be applied to registrations.

(2) The registration instances in Fact 1 are treated as analogues of contract instances.

(3) The notion of falsity bound to the class of contracts is generalized to more general notion of falsity that covers registrations.

It suffices to have three generalization rules, introduced in the next section, to realize both (1) and (2). In addition to these generalization rules, we need to have a knowledge used to judge the falsity of registrations (3) . Such knowledge can be supplied by the following rules:

Rule 3 (Knowledge of Registration)

$oc(Regp : reg\_of\_ptitle))$,
$attr(Regp, agent2, Agt), attr(Regp, object, Obj) \rightarrow repr\_of\_reg(Regp, Agt)$.
$oc(Regp : reg\_of\_ptitle), attr(Regp, object, Obj), ownership(Agt, Obj) \rightarrow soa\_of\_reg(Regp, Agt)$.
$repr\_of\_ctrct(X : contract, Y : top) \rightarrow repr(X, Y)$.
$soa\_of\_ctrct(X : contract, Y : top) \rightarrow soa(X, Y)$.
$repr\_of\_reg(X : registration, Y : top) \rightarrow repr(X, Y)$.
$soa\_of\_reg(X : registration, Y : top) \rightarrow soa(X, Y$).

Sorted Specifications:

$repr : human\_act, top.$ $soa : human\_act, top.$
$repr\_of\_reg : human\_act, top.$ $soa\_of\_reg : human\_act, top.$

The falsity rule 2 have two predicates: one is soa_of_ctrct (state of affairs) and the other is $repr\_of\_ctrct$ (representation). The rule 2 decides the falsity of a contract if the arguments of soa_of_ctrct and $repr\_of\_ctrct$ are different. Since Rule 3 on registrations can decide both soa_of_reg and $repr\_of\_reg$ for registrations, a rule obtained by generalizing the falsify rule 2 for contracts will succeeds for our case of registrations with an aid of Rule 3. All the techniques carrying out such a task will be presented in the next section.

4 SG-Generalization

In this section, we present a formal definition of sorted generalization. Furthermore, a kind of SLD-refutation equipped with sorted generalization is also presented. We will show in Section 5 that the extended refutation works as a basis for legal analogical reasoning.

4.1 Sorted Generalization

Our generalization is simply defined as a substitution that replaces a variable of some sort with another variable of more general sort.

Definition 2 *(SG-generalization)*

1. *A substitution $\theta = \{x_j/y_j\}$ replacing variables with variables is called a SG-generalization, if $[x_j] \leq [y_j]$.*
2. *Let Eq be a unifiable set of well-sorted terms. Eq is said to have a SG-solution or to be a SG-unifiable set, if there exists a SG-generalization θ such taht $Eq\theta$ has at least one mgosu τ. In this case, $\theta\tau$ is called a SG-solution for Eq.*

¿From the definition, a SG-unifiable set is also unifiable, although it may not have any o.s. unifier. In such a case, the SG-unifiablity means that the set of terms becomes unifiable under our sort hierarchy, after generalizing sorts of some variables that make our sorted unification fail. Therefore, in order to analyze SG-unification, we need to know conditions under which sorted unifications fail.

Definition 3 *(Walther [11]) Let Eq be a unifiable set of terms.*

1. *A binary relation $\sim$ between terms is defined as: $t \sim s \underset{\text{def}}{\Leftrightarrow}$ t and s are unifiable by some mgu of Eq. Clearly $\sim$ is an equivalence relation.*
2. *An equivalence class M for $\sim$ is called a SU-equivalence class, if it contains at least one variable and its number of elements is greater than or equal to 2.*

Proposition 1 *(Walther [11]) A set Eq of w.s. terms has a mgosu iff, for each SU-equivalence class M for Eq, a maximal lower bound of $\{[t] | t \in M\}$ exists.*

¿From this proposition, a unifiable set that is not o.s. unifiable has a SU-equivalence class whose sort set has no lower bounds. We say that such a SU-class violates sort condition. The algorithm we present here operates on such SU-classes, generalizes sorts of variable that causes the violation of sort condition, and then makes the original set of terms unifible in the sense of order sorted unification.

The algorithm consists of two components: The first one is for SU-equivalence classes of only variables. The second one is for SU-equivalence classes containing proper terms [7].

SG-generalization $\mathcal{A}_1$ for SU-equivalence class of variables:

[7] A term that is not a variable.

procedure $sg\_for\_variables(Z, Z')$
input: a n-tuple $Z = (z_1, ..., z_n)$ of variables such that $[Z] = \{[z_1], ..., [z_n]\}$ has no lower bounds.
begin
  if $n = 2$ then
    begin
      let (z'_1, z'_2) be the first pair in the enumeration such that
        (1) $[z_j] \leq [z'_j]$ holds for $j = 1, 2$ and
        (2) $\{[z'_1], [z'_2]\}$ has a lower bound;
      retrun$((z'_1, z'_2))$
    end;
  call $sg\_for\_variables(\{z_1, ..., z_{n-1}\}, \{z'_1, ..., z'_{n-1}\})$;
  choose a new variable z_{up} of $mlb\{[z'_1], ..., [z'_{n-1}]\})$;
  choose a new variable z_{next_up} such that
    (1) $[z_{next\_up}] \geq [z_n]$,
    (2) $[z_{next\_up}]$ and $[z_{up}]$ has a lower bound, and
    (3) $[z_{next\_up}]$ is minimal w.r.t (1) and (2);
  return$((z'_1, ..., z'_n))$
end

Figure 1: An Algorithm $\mathcal{A}_1$ for SG-generalizing SU-class of variables

- problem specification:
  Given: a n-tuple $Z = (z_1, ..., z_n)$ of variables such that $[Z] = \{[z_1], ..., [z_n]\}$ has no lower bounds.
  Find:
  a minimal n-tuple of variables $Z' = (z'_1, ..., z'_n)$ such that
    (1) $[z'_j] \geq [z_j]$ for all j, and
    (2) $[Z']$ has a lower bound,
  where the minimal n-tuple of variables is determined by the quasi-ordering defined as:
  $x'_1, ..., x'_n \leq x''_1, ..., x''_n \underset{\text{def}}{\Leftrightarrow} [x'_j] \leq [x''_j]$ holds for any j.
  The desired SG-generalization θ is then defined as $\theta = \{z_j/z'_j | 1 \leq j \leq n\}$
- method: apply the algorithm in Fig. 1, where we assume an enumeration of pairs of variables such that (x_1, y_1) is enumerated faster than (x_2, y_2) whenever $(x_1, y_1) < (x_2, y_2)$. We use this enumeration to find every possible candidates of variables to which original variables are generalized. Moreover, in the algorithm, we use a non-deterministic instruction "*choose* $<$*object*$>$ *such that* $<$ *condition* $>$ " that selects arbitrary object satisfying the condition.

SG-generalization $\mathcal{A}_2$ for SU-equivalence class of terms:

Let x_j, t_i and $M = \{x_1, ..., x_n, t_1, ..., t_m\}$ be a variable, a proper term and a SU-equivalence class violating the sort condition, respectively. For M is unifiable, $[t_1] = ... = [t_m] = s$ holds for some sort $s \in S$. Hence there exists at least one variable, say x_j, such that $[x_j] \geq s$ does not hold. For any such a variable x_j, take a minimal upper-bound $s_j = mub\{s, [x_j]\}$, and introduce a new variable z_j of s_j. Then the desired SG-generalization is defined as a substitution replacing x_j with z_j.

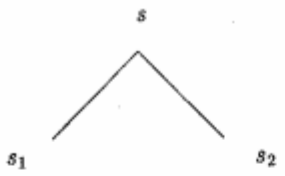

Figure 2: A simple sort hierarchy

The SG-generalization θ that makes Eq o.s. unifiable is now simply defined as a union of substitutions. Let $M_1, ..., M_k$ be all the SU-equivalence relations violating our sort condition. For each M_j, if it contains a proper term then apply $\mathcal{A}_2$ else apply $\mathcal{A}_1$. This produces SG-generalizations $\theta_1, ..., \theta_k$. Then the final SG-generalization is defined as:

$$\theta_{Eq} = \bigcup_{j=1}^{k} \theta_j \ .$$

Proposition 2 *Let Eq be a unifiable set of w.s. terms. If Eq has a SG-solution, then θ_{Eq} is a minimal SG-generalization such that $Eq\theta_{Eq}$ has mgosu.*

4.2 SLD-resolution equipped with SG-Generalization

Before describing the extended resolution, let us exemplify some problem found in applying SG-generalization to the standard resolution principle.

Suppose we have a sort hierarchy shown in Figure 2, goal clause $\leftarrow p(x : s_1)$ and a fact $p(b)$, where b is a constant symbol of sort s_2.

Since s_2 is not a subsort of s_1, our order sorted unification fails. If we generalize the sort s_1 to the common super sort s, then our order sorted unification succeeds, and an empty clause is derived with an answer substitution $\{x/b\}$. This is performed by SG-generalization $\{x : s_1/z : s\}$ followed by an order sorted unification $\{z : s/b : s_2\}$. However our standard operational semantics for goal clauses has changed, since the sort of variable in goal clause is generalized. The original goal means a question "Is there some instance of s_1 satisfying p". On the otherhand, the obtained answer says that "that instance you desire is b", not an instance of s_1. This kind of question-answering seems semantically confusing. For this reason, our SG-generalization is prohibited from generalizing sorts of variables in goal clauses. Only variables in rules or facts are allowed to be generalized. Under this restriction, we now describe our extended SLD-refutation equipped with a function of SG-generalizing variables in rules.

Extended SLD-Resolution: Basically our extended resolution is the order sorted SLD-resolution that we have briefly described in Subsection 2.1. In addition to that, we allow to use SG-generalization before applying order-sorted resolution, provided o.s. unification fails. More formally speaking, our derivation is a finite sequence $G_0, G_1, ..., G_n$ of well-sorted goal clauses with a sequence $< R_1, \theta_1 >, ..., < R_n, \theta_n >$ of pairs of defnite clauses R_j and substitutions θ_j. The latter sequence of pairs should satisfy exactly one of the following two conditions:

(C1) [standard o.s. SLD-resolution]:
The head B_j of R_j and a selected atom A_{j-1} in G_{j-1} is o.s. unifiable. θ_j is their mgosu.

(C2) [restricted SG-generalization followed by standard resolution]:

1. B_j and A_{j-1} is not o.s. unifiable,
2. $\theta_j = \tau\sigma$ for some SG-generalization τ and o.s. substitution σ satisfying the following requirement :
   τ is a possible output of our SG-generalization algorithm for B_j and A_{j-1} such that $x\tau = x$ whenever $x\theta_1...\theta_{n-1} = z\theta_1...\theta_{n-1}$ for a variable z appeared in the top goal G_0.
3. σ is mgosu of $B_j\tau$ and $A_{j-1}\tau$.

It should be noted here that we never derive a goal clause that is not well-sorted. If some candidate SG-generalization produces an ill-sorted goal clause, then it is rejected, and another SG-generalization is tried. Thus the well-sortedness is kept during the whole process of derivation. Moreover the restriction on SG-generalization means that SG-generalizations should be identical for any variable that is unified with some variable in top goal clause G_0.

The following proposition summarizes the arguments in this section.

Proposition 3 *Given a set of definite clauses P, let $Ext(P)$ be the set $\{C\theta | C \in P, \theta$ is a SG-generalization, and $C\theta$ is well-sorted $\}$. Then $Ext(P) \vdash \alpha \Leftrightarrow$ there exists our extended SLD-derivation of empty clause from $\leftarrow \alpha$.*

5 Order-sorted generalizations

Now we are ready to show how to utilize legal knowledge and how to carry out legal reasoning in an order-sorted symbol system for which order-sorted resolution and generalization are applied.

First suppose that we try to construct a legal argument to protect the right of good faith person p_c appeared in the case presented in Section 3. Since the plaintiff p_a claimed that p_c should do a cancellation procedure of passage of title, it suffices to show that he cannot set up p_c with respect to the passage of title.

Hence we first make the following goal expressed as an well-sorted goal clause:

$$\leftarrow cannot\_set\_up(p\_a, p\_c, reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X))$$

Our system is basically a backward reasoner just like a Prolog interpreter. It first applies order-sorted resolution to a given goal clause, whenever there remains an atom in the goal clause with which some rule or a fact can be resolved.

Control 1 *Let $\leftarrow A_1, ..., A_n$ be a given well-sorted goal clause. If there exists an atom A_j with which some rule or a fact can be resolved, then do order-sorted resolution to produce the next goal* [8]*. If otherwise, apply SG-generalization followed by order sorted resolution for an atom A_j and a rule $A \leftarrow W$.*

In our example case , Rule 1 and the atom (1) are unifiable. However they fails in order-sorted unification due to the type constraint for the third argument of *cannot_set_up*. Our SG-generalization is tried so that it weakens the type constraints and makes the rule applicable to the atom. For this purpose, we use SG-generalizations described in Section 4.

In the present case, we have three SU-equivalence classes:

$$M_1 = \{X : person, p\_a\}, \quad M_2 = \{Z : person, p\_c\},$$
$$M_3 = \{Ctrct2 : contract, reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X)\}.$$

Only M_3 is the class violating our sorted condition. Since $mub(\{contract, reg\_of\_ptitle\}$ is a singleton set $\{lawful\_act\}$, so the $Ctrct2$ should be replaced with a variable $Lawful\_act : lawful\_act$. Thus the desired SG-generalization is just $\theta_1 = \{Ctrct2 : contract/Lawful\_act : lawful\_act\}$. Moreover, using θ_1, our extended SLD-resolution has an effect of generalizing Rule 1 to the following rule, and then resolves the latter rule with the original goal to produce next goal clause.

Rule 4 (Article94, Clause2: generalized one)

$oc(Ctrct1 : contract),$
$\quad attr(Ctrct1, agent1, X : person),$
$\quad attr(Ctrct1, agent2, Y : person),$
$\quad attr(Ctrct1, agent1, Obj),$
$oc(Falsity : falsity),$
$\quad attr(Falsity, obj, Ctrct1),$
$oc(Lawful\_act : lawful\_act),$
$\quad attr(Lawful\_act, agent1, Y),$
$\quad attr(Lawful\_act, agent1, Z : person),$
$\quad attr(Lawful\_act, agent1, Obj),$
$good\_faith(Z, Falsity)$
$\quad \rightarrow cannot\_set\_up(X, Z, Lawful\_act)$

[8] Normally the left most atom is selected, if there exist several atoms that can be resolved with some rules.

Clearly Rule 4 is applicable to our top goal (1). Hence we have the following as the next goal clause:

$\leftarrow oc(Ctrct1 : contract),$
$\quad attr(Ctrct1, agt1, p\_a), attr(Ctrct1, agt2, Y),$ (1)
$\quad attr(Ctrct1, agt1, Obj : object),$ (2)
$\quad oc(Falsity : falsity),$
$\quad attr(Falsity, obj, Ctrct1),$ (3)
$\quad oc(reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X)),$ (4)
$\quad attr(reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X), agt1, Y),$ (5)
$\quad attr(reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X), agt2, p\_c),$ (6)
$\quad attr(reg\_of\_ptitle\_f(p\_b, p\_c, imm\_X), object, Obj),$ (7)
$\quad good\_faith(p\_c, Falsity)$

According to Control Rule 1, every possible resolution is tried first. In this case, for the atoms (1), (2), (3), (5), (6) and (7) are eliminated from the goal clause from the unit clauses defining the *attr* predicate. Furthermore the atom (4) is really a fact, so it is also eliminated. As a result, we have the following new goal clause after several steps of order-sorted resolutions:

$\leftarrow oc(contract\_f(p\_a, p\_b, imm\_X),$
$\quad oc(falsity\_f(contract\_f(p\_a, p\_b, imm\_X)),$ (8)
$\quad good\_faith(p\_c, falsity\_f(contract\_f(p\_a, p\_b, imm\_X))).$

Now from the legal theory rule in Rule 2, the atom (8) is resolved, and the goal clause becomes:

$\leftarrow oc(contract\_f(p\_a, p\_b, imm\_X),$
$\quad repr\_of\_ctrct(contract\_f(p\_a, p\_b, imm\_X), ReprCts),$
$\quad soa\_of\_ctrct(contract\_f(p\_a, p\_b, imm\_X), RealCts),$
$\quad ReprCts \neq RealCts,$
$\quad good\_faith(p\_c,$ (9)
$\quad\quad falsity\_f(contract\_f(p\_a, p\_b, imm_X)))$ (10)

It should be noted here that every atom in the above goal clause fails. Especially (10) cannot succeed from the definition of *good_faith*, for

$$oc(contract\_f(p\_a, p\_b, imm_X))$$

fails. From Control Rule 1, we should apply Sorted Generalization Rule. However, no rule is unifiable with any atom in the goal clause, so Generalization Rule also fails to produce a new hypothetical rule.

Thus we need another type of generalization to accomplish our task. Recall that our sorted generalization tries to generalize sorts of variables appeared in a rule. It is also possible to consider a "term generalization" that generalizes a term to a variable of more general sort. To investigate the condition for this type of generalization, let us consider a simple example. Suppose we have sorts s_3, s_4 of event type such that $s_3 \leq s_4$ and two functors $g : s_1 \rightarrow s_2$, and f with its sorted specification $f(l : s_2) : s_3$. Then the following is a logical deduction:

$$\frac{\exists z : s_1 \;\; oc(f(g(z)))}{\exists x : s_4 \;\; \exists z : s_1 \;\; oc(x) \wedge attr(x, l, g(z))}$$

The term $f(g(z))$ of sort s_3 is replaced with a variable x of more general sort s_4. Hence a goal derivation from $\leftarrow oc(f(g(z:s_1))$ to $\leftarrow oc(x:s_4), attr(x,l,g(z:s_1))$ should be a generalization of goal. Now we preset here our second generalization rule under some additional condition.

Generalization 1 (Term Generalization) *Suppose we have a goal clause*

$$\leftarrow oc(f(t_1,...,t_n)), Bs \tag{11}$$

where $f(l_1:s_1,...,l_n:s_n):s$. *Then find a super sort* s' *of* s [9] *such that a goal clause defined by*

$$\leftarrow oc(y:s') \wedge \bigwedge_{i=1}^{n} attr(y,l_j,t_j)$$

is provable by standard order-sorted resolution. If this succeeds with an answer substitution θ, *then remove the first atom* $oc(f(t_1,...,t_n))$ *from the original goal (11) and replace its all the occurrences in (11) with* $y\theta$.

In the present case we are examining, the goal

$$oc(X:lawful\_act), attr(X,agt1,p\_a), attr(X,agt2,p\_b), attr(X,object,imm\_X) \tag{12}$$

is generated $oc(contract\_f(p\_a,p\_b,imm\_X))$ and proved with the following answer [10]:

$$\theta = \{X/reg\_of\_ptitle\_(a,b,imm\_X)\}$$

Thus we have the next goal clause from the original goal (11), according to Generalization Rule 1:

$$\leftarrow repr\_of\_ctrct(reg\_of\_ptitlet\_f(p\_a,p\_b,imm\_X),ReprCts) \tag{13}$$
$$soa\_of\_ctrct(reg\_of\_ptitlet\_f(p\_a,p\_b,imm\_X), \tag{14}$$
$$RealCts), \tag{15}$$
$$ReprCts \neq RealCts,$$
$$good\_faith(p\_c, falsity\_f(reg\_of\_ptitle\_f(p\_a,p\_b,imm_X)))$$

Since our term generalization generalizes a term to a vaiable of more general sort, the replacement performed by the generalization may introduce some type errors. In the present case, the atoms (13) and (15) are not well-sorted, because the domain sort of $repr\_of\_ctrct$ is *contract*. Thus we need to have a type error elimination rule whenever the term generalization succeeds and some illegal terms become to appear in the new goal clause. The elimination is also carried out by a generalization.

Control 2 (Type Check) *We assume to check if a goal clause obtained by Generalization Rule 1 is well-sorted or not, whenever it is applied. If some ill-sorted expression is found, then execute Generalization Rule 2. Precisely speaking, suppose* $oc(f(t_1,...,t_n))$ *is replaced with* $y\theta$ *in Generalization Rule 1 to produce the next goal clause* G. *Let* $A_1(y\theta),...,A_n(y\theta)$ *be all the ill-sorted atoms appeared in* G. *Apply Generalization Rule 2, called Predicate Generalization, to a set of atoms:* $A_1(y\theta),...,A_n(y\theta)$.

The predicate generalization is originally introduced in [[2]] as a kind of abductive goal reduction rule, and is now extended so as to cope with well-sorted expressions. The major function of predicate generalization is to apply our (object-level) rules to the ill-sorted goal clause to eliminate the ill-sorted expressions. Before describing it formally, we present here an simple illustration.

Suppose symbols a and b denote

$$contract\_f(p\_a,p\_b,imm\_X) \text{ and}$$
$$reg\_of\_ptitle\_f(p\_a,p\_b,imm\_X),$$

respectively. Assume furthermore that an well-sorted atom

$$repr\_of\_cntrct(a,Cntrct) \tag{16}$$

holds as a hypothetical premise for predicate generalization. Recall that the ill sorted expression $repr\_of\_cntrct(b,Cntrct)$ is obtained from the premise (16) by replacing a with b. To the contrary, our predicate generalization first generalizes the premise so that the replacement for a generalized premise $B(a)$ produces an well-sorted expression $B(b)$.

(PG1): $B(a)$ is provable from (16)

(PG2): $B(b)$ is well-sorted.

¿From the conditions (PG1) and (PG2), both $B(a)$ and $B(b)$ are well sorted. Hence there exists a sort s such that

(PG3): $B(x:s)$ is well-sorted, where x is a variable,

(PG4): $contract \leq s$ and $registration \leq s$.

¿From the syntactic assumption for our symbol system, there exist just two atoms as candidates of B that satisfy (PG3) and (PG4):

$$oc(falsity\_f(X:lawful\_act)), \tag{17}$$
$$repr(X:lawful\_act,Cntrct) \tag{18}$$

[9] When there exist several sorts s' satisfying the condition, we assume to choose a minimal one.

[10] Strictly speaking, we assume the following additional rules to make the goal (12) succeed:

$$oc(reg\_of\_ptitle\_f(X:person,Y:person,W:imm\_prop) \leftarrow oc(reg\_of\_ptitle\_f(Y,Z:person,W)), ownership(X,W).$$

This rule is used to derive a hypothetical fact on the registrations. The registration of passage of title from Y to Z does not imply a passage from the real owner X to the nominal owner Y. However, from the standpoint of the third person Z, it can be assumable that the passage of title from X to Y was registered. Although we should distinguish two types of facts under a framework of "abductive logic programming", we leave this issue as a future work.

¿From the condition (PG1), we can conclude that the atom (17) is rejected and the atom (18) is accepted. As a result, our predicate generalization replaces all the occurrence of ill-sorted atom $repr\_of\_cntrct(b, Cntrct)$, appeared by Term Generalization, with the well-sorted $repr(b, Cntrct)$.

As a result, we have:

$$\begin{aligned} \leftarrow \quad & repr(reg\_of\_ptitlet\_f(p\_a, p\_b, imm\_X), ReprCts), \\ & soa(reg\_of\_ptitlet\_f(p\_a, p\_b, imm\_X), RealCts), \\ & ReprCts \neq RealCts, \\ & good\_faith(p\_c, falsity\_f(\\ & \qquad reg\_of\_ptitle\_f(p\_a, p\_b, imm_X))) \end{aligned}$$

Now it is clear that all the atoms in the above is refutable by standard order-sorted resolution.

Finally we present the generalization rule in its general form:

Generalization 2 (Predicate Generalization)
Let $A_1(a), ..., A_n(a)$ be a set of well-sorted atoms, where a is a term of event type. Furthermore suppose $A_1(b), ..., A_n(b)$ is ill-sorted, where b is an well-sorted term of event type. Then find a set of well-sorted atoms $B_1(a), ..., B_k(a)$ such that $k \leq n$, $A_1(a), ..., A_n(a) \vdash B_1(a), ..., B_k(a)$, and $B_1(b), ..., B_k(b)$ is well-sorted.

¿From the definition of predicate generalization, it is possible to consider two kinds of computational rules. One is to generate possible deductions until the ill sorted expression disappears. The other one is to enumerate atom sets $B_1(x), ..., B_n(x)$ that meet our syntactic constraints, and then check if they are provable or not. For our present example, these two methods pay similar computational costs.

6 Concluding remarks

We have defined an order-sorted symbol system, represented various types of legal knowledge, and showed how three generalization rules are applied. Now we breiefly discuss some important issues not yet mentioned in this paper.

First the sorted generalization has been introduced so as to make a unification failure recover by generalizing sorts of variables. For this purpose, we required that the generalized sorts have their common lower bound which will be a sort of unified terms after sorted generalization. Here we want to call the generalized sort and the sort of unified term a generalization point and a unification point, respectively. The process for our sorted generalization thus involves finding the generalization points for which a unification point exists. Such a process might become complex when we have a large sort hierachy in which multiple super sorts are allowed to exist. Hence we must have more efficient method to compute the generalization points. The author is now developing such an algorithm according to the following strategy:

1. Mapping the sort hierarchy into a set of primitive elements corresponding to minimal sorts in the hierarchy. This transforms the hierarchy into a partially ordered subset of a set boolean lattice.

2. Before finding the generalization points, computing candidates for the unification points by set theoretic operations.

3. Once such a unification point is found, it is easy to find the generalization points by simple algebraic operations.

The second problem we have to discuss is how we handle hypothetical rules. The rules obtained by sorted generalization is clearly hypothetical ones. In addition, we have used another type of hypothetical rules in Section 5 to infer hypothetical facts that are substantially assumable but are not explicitly recorded as approved facts. Furthermore one may assert that even our sort hierachy might be hypothetical. For instance, when we are talking about some issues about some legal concepts and cases, we may choose particlar super sorts to emphasize some aspects of the concepts. In such a case, other super sorts should be neglected or blocked to prevent useless and harmful arguments. The studies focussing their themes on such a problem is well known as default reasoing or as multiple inheritance.

Default reasoning is generally concerned with a situation that is incompeletly specified by a set of evidences supporting hypothetical facts or rules. The problem is to select some hypotheses from the evidences and to conclude that they might hold in the situation. Some evidence can distinguish some useful default rules from the others, and blocks some irrelevant hypotheticl rules. A mechanism which can behave in such a way is known as conditional entailment [22]. In addition to the function of conditional entailment, some experimental enviroment under which we can get additional evidences to test whether the hypothetical rules are relevant to the situation or not seems to be necessary. The author is now trying to design a system that is basically a conditional entailement system but has additional function to acheive such an experimental environment.

References

[1] Tanaka, H. (1974) *Introduction to the study of positive law* (in Japanese), University of Tokyo Press.

[2] Haraguchi, M. (1991) A form of analogy as an abductive inference, *Proc. 2nd Workshop on Algorithmic Learning Theory*, pages 266-274, Japanese Society for Artificial Intelligence.

[3] Haraguchi, M. (1992) What kinds of knowledge and inferences are needed to realize legal reasoning? (in Japanese), *Proc. 6th symposium on knowledge representation and legal reasoning system*, Legal Expert System Association in Japan.

[4] Muggleton, S. and Buntine, W. (1988) Machine invention of first-order predicates by inverting resolution, *Proc. Workshop on Machine Learning*, pages 339–352.

[5] Yoshino,H., Haraguchi,M.,Sakurai,S.,Kagayama,S. (1993) Towards a Legal Analogical Reasoning System: Knowledge Representation and Reasoning Methods, *Proc. 4th ICAIL*, 110-116, 1993

[6] Yoshino, H. (1987) Legal expert system LES-2, *Springer Lecture Notes in Computer Science, Logic Programming '86* pages 34-45.

[7] Aoumi, Z. (1989) *Introduction to Philosophy of Law* (in Japanese), Koubunn-dou.

[8] Guarino, N. (1991) A concise presentation of ITL, *Proc. of Processing Declarative Knowledge*, 141-190, Springer-Verlag.

[9] Beierle,C. et.al. (1992) An order-sorted logic for knowledge representation systems, *Artif. Intell.*, *55*, 149–191.

[10] Nebel, B. (1990) Reasoning and Revision in Hybrid Representation Systems, Springer LNAI, *422*, 270 pages

[11] Walther, C. (1988) Many sorted unification, *JACM*, *35*, 1, 1–17.

[12] Waragai,T. (1994) A natural extension of predicate calculus in which ISA relation is expressible, Ph.D. Thesis, Tokyo Institute of Technology.

[13] Muggleton, S. (1990) Inductive logic programming, in *Proc. 1st Workshop of Algorithmic Learning Theory*, pages 42–66.

[14] H. Gasyuu (1986) *Analogy in law* (in Japanese), Unpublished Lecture Note, Legal Expert Systems Association, Meiji Gakuinn Univ., Tokyo.

[15] Rouveirol C. (1991) *"ITOU: Induction of First Order Theories"*, in *Proceedings of the first Inductive Learning Programming Workshop*, Viana de Castelo.

[16] D.Greiner (1988) Abstraction-based Analogical Reasoning, in Analogical Reasoning, in *Analogical Reasoning*, Kluwer Academic Publishers, D.H.Helman ed., 147–180.

[17] Indurkya,B. (1990) On the Role of Interpretive Analogy in Learning, ALT'90, 174–189.

[18] Russell,S.J. (1986) Analogical and Inductive Reasoning, STAN-CS-87-1150, Dept. Computer Science, Stanford Univ.

[19] Harao,M (1993) Generalized-knowledge acquisition and reasoning based on similarity (in Japanese), in *Proc. of 7th Annual Conference of JSAI*, 41–44.

[20] Dierbach,C. et.al.(1992) A formal basis for analogical reasoning, *Proc. KR'91*, 139–150.

[21] Lloyd,J.W. (1984) *Foundation of Logic Programming*, Springer-Verlag.

[22] Geffner,H. and Pearl,J. (1992) Conditional Entailment: bridging two approaches to default reasoning, *Artif. Intell.* 53, 209–244.