1998 International Computer Symposium
Workshop on Artificial Intelligence
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

Inheritance Hierarchies and Partial Ordering

Hung-Tsow Chen

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
Email: chenht@dbl.iie.ncku.edu.tw
Tel: 886-6-2757575 x89297
Fax: 886-6-2747076

Abstract

In this paper, we present a set of inference rules for the property inheritance theories often encountered in commonsense reasoning. Then, we generalize the set of rules so that the monotonicity of function symbols or predicate symbols can be encoded in the inference rules. Consequently, we can replace cumbersome axioms by some succinct rules. We also discuss the completeness and soundness of these rules and compare the rule-based approach to other related works.

1. Introduction

There are issues in artificial intelligence over the reasoning about inheritance hierarchies, which occurs often in commonsense domains. Many researches have been done on the inheritance hierarchies with exceptions and their relationships with default logic (e.g. [4], [5] and [9]). It is cumbersome to specify for every property P inherited from one class to its subclass by every axiom, so the ordinary resolution rule could be modified so that the property inheritance mechanism can be embedded into the unification process. Although the framework, called partial order logic, provides a rigorous theoretical foundation for further research. It didn't really specify an inference mechanism for the property inheritance rule.

In this paper, we are going to make up for this deficiency and offer the reasoning mechanism that includes the property generalization and specialization rules.

In the following sections, we first describe the basic notions of partial order logic proposed by Chen and Warren ([3]). Then, we propose a paramodulation-like rule to reason about inheritance hierarchies. After presenting this rule, we discuss a set of more complex and powerful rules which can be used to reason about property generalization and specialization and provide a more flexible inference mechanism for inheritance hierarchies. Finally, we give a brief conclusion to explain what we have achieved.

2.Partial Order Logic

The syntax of partial order logic is the same as that of the first order predicate calculus except that the set C of constants is ordered and the set F_n of n-ary function symbols can be partitioned into two disjoint sets Mon_F_n and Non_F_n for every $n > 0$.

Let $\leq$ denote the partial order relation over C. It could be treated as "imply" for predicate symbols or as "value comparison" for function symbols, and then it can be extended to the whole set of terms in the following ways:

• $x \leq x$ for every variables x;

• $f(t_1, t_2, \ldots, t_n) \leq f(s_1, s_2, \ldots, s_n)$ iff $t_i \leq s_i$ for i from 1 to n, where $f \in Mon\_F_n$;

• $f(t_1, t_2, \ldots, t_n) \leq f(t_1, t_2, \ldots, t_n)$ where $f \in Non\_F_n$;

Obviously, the extended relation is still a partial ordering over terms.The partial ordering over terms can be further extended to atomic formula as follows:

• $P(t_1, t_2, \ldots, t_n) \leq P(s_1, s_2, \ldots, s_n)$ iff $s_i \leq t_i$ for i from 1 to n, where P is an n-ary predicate symbol.

Because the value of "true" is greater than that of "false", and $P(t_1, t_2, \ldots, t_n) \leq P(s_1, s_2, \ldots, s_n)$ means that $P(t_1, t_2, \ldots, t_n)$ logically implies $P(s_1, s_2, \ldots, s_n)$. For example, if we have nightingale $\leq$ bird, then fly(bird) logically implies fly(nightingale).

A semantic structure of partial order logic is a first order structure M which satisfies the following properties:

• |M| is a partial ordered set and let $\leq$ be the partial order over |M|;

note: |M| means the domain of M.

• $M[c_1] \leq M[c_2]$ if $c_1 \leq c_2$ for all $c_1, c_2 \in C$;

• M[f] is a monotonic function from $|M|^n$ to |M| if $f \in Mon\_F_n$, where the partial ordering over $|M|^n$ is the component-wise extension of the partial ordering over |M|;

• if $a_i \leq b_i$ ($1 \leq i \leq n$), where $a_i, b_i \in |M|$, and $(b_1, b_2, \ldots, b_n) \in M[P]$, then $(a_1, a_2, \ldots, a_n) \in M[P]$ for each n-ary predicate symbol P.

Finally, the partial order resolution rule is:

(r):

$$\frac{C_1 \vee A, \sim B \vee C_2}{(C_1 \vee C_2)\theta}$$

, where θ is a substitution satisfying $A\theta \leq B\theta$, and $C_1 \vee A$ and $\sim B \vee C_2$ without common variables.

This rule is actually an instance of Stickel's total theory resolution([7]) on property inheritance hierarchies. Stickel's rules are obtained by incorporating the background theories into the unification process of the ordinary resolution principle.

The refutation completeness for this rule is stated in the following theorem.

The following conditions are equivalent:

(a) The set S of clauses is unsatisfiable in all partial order semantic structure.

(b) There is a refutation of S (i.e. a derivation of empty clause from S) by using the partial order resolution rule (r) and the factoring rule of classical logic.

From the above presentation, we can find that partial order logic is a succinct logical system for reasoning about property inheritance theory, but it isn't a real inference mechanism. As we have pointed out above, the rule (r) of partial order logic is an instance of total theory resolution, and the completeness of the theory resolution in fact depends on the background theory.

Moreover, the partial order logic defined above still suffers some other defects. It seems too restrictive. Since the ordering is built into the language, we will be under more limitations to represent all possible ordering over terms. For example, we can't express the ordering between two terms with different functions symbols or interacts the partial ordering within different predicates.

On the other hand, if we use the classical axiomatic approach to manage the above issues. Since the ordering symbol is considered an ordinary binary predicate symbol, we can express any sentences with it.

Moreover, since the inference rule is the ordinary resolution principle, the syntactic unification algorithm can be used with no problem. However, going back to the axiomatic approach, we will lose the major advantages of partial order logic because we must write one property inheritance axiom for each predicate symbol and each function symbol in Mon_F_n.

To compensate for these deficiencies, we propose a schema so that we can encode as much regular information into the language as we can. At the same time, our language should remain flexible enough to express the irregular information.

3. Simple Property Inheritance System

In this section, we will present the first compromise between classical axiomatic approach and partial order logic to inheritance hierarchies.

The language we use is the first order predicate calculus with equality and ordering ($\leq$) symbols. The language is denoted by $L_{\leq}$. Let EQ and PO denote the axioms for equality and partial order theories respectively. Note: $\supset$ means imply.

EQ:

$x = x$ (reflexivity),

$x = y \supset y = x$ (symmetry),

$x_1 = y_1 \ \& \ x_2 = y_2 \ \&\ldots\& \ x_n = y_n \supset f(x_1,\ldots,x_n) = f(y_1,\ldots,y_n)$ for any n-ary function f,

$x_1 = y_1 \ \& \ x_2 = y_2 \ \&\ldots\& \ x_n = y_n \ \& \ P(x_1,\ldots,x_n) \supset P(y_1,\ldots,y_n)$ for any n-ary predicate P.

PO:

$x \leq x$ (reflexivity),

$x \leq y \ \& \ y \leq x \supset x = y$ (antisymmetry),

$x \leq y \ \& \ y \leq z \supset x \leq z$ (transitivity).

Moreover, we need the following property inheritance axioms.

IN:

$x_1 \leq y_1 \ \& \ x_2 \leq y_2 \ \&\ldots\& \ x_n \leq y_n \ \& \ P(y_1,\ldots,y_n) \supset P(x_1,\ldots,x_n)$ for any n-ary predicate P.

Let $\pi = EQ \cup PO \cup IN$, then a set of clauses S is called π -unsatisfiable iff $S \cup \pi$ is unsatisfiable.

Now, we give a set of inference rules which can derive the empty clause from S when S is π-unsatisfiable. First, let R_0 denote the set of classical resolution, paramodulation and factoring rules.

The rules to simulate PO and IN axioms are as follows:

(A):

$$\frac{s_1 \leq t_1 \vee C_1 \ , \ t_2 \leq s_2 \vee C_2}{(s_1 = t_1 \vee C_1 \vee C_2)\sigma}$$

where σ is the mgsu of (s_1, s_2) and (t_1, t_2) (i.e. $s_1\sigma = s_2\sigma$ and $t_1\sigma = t_2\sigma$).

(I):

$$\frac{P(s_1, s_2, \ldots, s_n) \vee C_1,\ t \leq u \vee C_2}{(C_1 \vee C_2 \vee P(s_1, s_2, \ldots, s_{i-1}, t, s_{i+1}, \ldots, s_n))}$$

where $\sigma = \mathrm{mgu}(s_i,u)$, and P is an n-ary predicate and i is an integer between 1 and n.

Let **R**= **R**$_0$ $\cup$ **{(A),(I)}**, then the completeness theorem of simple property inheritance theory can be stated as follows:

Theorem 3.1:

A set S of clauses is π-unsatisfiable iff S $\cup$ {x = x, x $\leq$x} $\vdash_{\mathbf{R}}$ empty clause, where $\vdash_{\mathbf{R}}$ means the derivation relation under rules in **R**.

Note that we have shifted the ordering symbol from metalanguage to the object first order language, so that we can express any irregular information about ordering over terms in our theory, and in the meanwhile, we encode the information about property inheritance into inference rules; hence we don't have to specify the axioms for them explicitly. If our first order language doesn't contain any monotonic function symbols, we don't sacrifice too much of the simplicity of partial order logic.

However, when we have many monotonic function symbols, our system is comparatively more complex than partial order logic we have to specify the monotonicity conditions explicitly by axioms for each of such function symbols. Therefore, to fulfill our previous promise to encode as much regular information about ordering into our logic as possible, apparently, the above system still leaves something to be desired.

The improved system is presented in the following section.

4.General Property Inheritance System

Observing the rule (I) in the preceding section, we can see that the inequality atom t $\leq$ u is applied to the atom P(s_1, s_2 ,..., s_n) only at the argument positions, but not at any subterm positions. This is because we don't know the truth value of P(s_1, s_2 ,..., s_n) changes with its arbitrary subterms. Furthermore, the exception that occurs when P is equality or ordering symbols also shows that the property inheritance mechanism may be disabled or inverted in some argument(or subterm) positions of some predicates. To facilitate the representation of these situations we must generalize the notion of subterm occurrences to specify the path from root to the subterm position when we represent a term as a labeled tree so that we can simultaneously specify the way how the changes of a subterm influence its superterm. First of all, we have to know a function(or predicate) symbol is monotonically increasing, decreasing, or neither with some of its argument.

Definition 4.1:

A monotonicity information function (m.i.f.) m is a partial function from ((P $\cup$ F $\cup$ {=,$\leq$})x N) to {-1,0,1} where P is the set of ordinary predicate symbols, F the set of function symbols and N the set of positive integers and m must satisfy the following requirements:

(i) m(s,i) is defined iff $1 \leq i \leq \mathrm{arity}(s)$,

(ii) m(=,1) = m(=,2) = 0,

(iii) m($\leq$,0) = -1, m($\leq$,2) = 1.

Note: Here, a predicate is considered as a function whose domain is BOOL and the ordering on BOOL is "false $\leq$ true". Then m(f, i) = 1 (resp. = -1) means that we must interpret f as follows:

When the ith argument of f increases monotonically and all the other arguments remain fixed, the value of f will increase (resp. decrease) under the ordering which is the interpretation of $\leq$.

Definition 4.2:

Given two words (terms or atoms) s and t, the occurrences of t in s are defined recursively as strings in $((P \cup F \cup \{=,\leq\}) \times N)^*$:

(i) t has ε (empty)occurrence in t,

(ii) if $s = f(t_1, t_2, \ldots, t_n)$ (or $P(t_1, t_2, \ldots, t_n)$), and there are occurrence λ in t_i, then there are occurrence $(f,i) \circ \lambda$ (or $(P,i) \circ \lambda$) in s, where $\circ$ means string concatenation.

Note: t may have several different occurrences in s.

If t has occurrence λ in s, we write $s[\lambda \leftarrow t]$ or s[t] for short if it is not confusing.

Now, we can extend a given m.i.f. m to the domain $((P \cup F \cup \{=,\leq\}) \times N)^*$:

(i) $m(\varepsilon) = 1$,

(ii) $m(\lambda) = m(\lambda_1) * m(\lambda')$, where $\lambda_1 \in ((P \cup F \cup \{=,\leq\}) \times N)$, $\lambda' \in ((P \cup F \cup \{=,\leq\}) \times N)^*$ and $\lambda = \lambda_1 \circ \lambda'$.

Example 4.1:

We give an example to explain the monotonicity information function. Let F contain one unary function symbol, succ, and two binary function symbols, + and -, and P contain two unary ordinary predicate symbols, Pos and Neg. Assume the intended meanings of succ, +, -, Pos, and Neg are successor function (on integers), integer addition, integer subtraction, "positive", and "negative" respectively. Then we can define the associated m.i.f. **m** as follows if the ordering symbol is interpreted as the ordinary "less than or equal to" relation on integers:

m(succ,1)=m(+,1)=m(+,2)=m(-,1)=m(Pos,1)=1,

m(-,2)=m(Neg,1)=-1.

Thus, for the given term x-succ(y), m((-,1))=1, and m(-,2)*m(succ,1) = -1. This means that, under the intended interpretations, the term x-succ(y) is interpreted as a function which is monotonically increasing on the first argument x, and decreasing on the second argument y.

Moreover, for a given **m**, we can define the following monotonicity and property inheritance axioms:

MO_m:

for all n>0, n-ary function f, and $1 \leq i \leq n$,

$x_i \leq y_i \supset f(x_1, \ldots, x_i, \ldots, x_n) \leq f(x_1, \ldots, y_i, \ldots, x_n)$ if $m(f,i) = 1$,

$y_i \leq x_i \supset f(x_1, \ldots, x_i, \ldots, x_n) \leq f(x_1, \ldots, y_i, \ldots, x_n)$ if $m(f, i) = -1$.

IN_m:

for all n >0, n-ary predicate P, and $1 \leq i \leq n$,

$x_i \leq y_i \supset f(x_1, \ldots, x_i, \ldots, x_n) \leq f(x_1, \ldots, y_i, \ldots, x_n)$ if $m(f, i) = 1$,

$y_i \leq x_i \supset f(x_1, \ldots, x_i, \ldots, x_n) \leq f(x_1, \ldots, y_i, \ldots, x_n)$ if $m(f,i) = -1$.

Let π_m= **EQ** $\cup$ **PO** $\cup$ **IN_m** $\cup$ **MO_m**, then any model of π_m is called a π_m-interpretation.

A set S of clauses is called π_m-unsatisfiable iff there don't exist π_m-interpretations which are the models of S. That is S is π_m-unsatisfiable iff $S \cup \pi_m$ is unsatisfiable.

Note:

The simple property inheritance theory π is just a special case which satisfies the following requirements: m(P, i) = 1 for all ordinary predicates P and m(f,i)=0 for all function symbols f.

According to these definitions, we come up with the following lemma.

Lemma 4.1:

Given an m.i.f. **m** (and its extension to the string domain) and any π_m -interpretation I, and t, t' are two terms, then we have:

(1) if s is a term, t has occurrence λ in s, and $I[t] \leq I[t']$, then $I[s[t]] \leq I[s[t']]$ when m(λ)=1, and $I[s[t']] \leq I[s[t]]$ when **m**(λ)=-1, and

(2) if s is an atom, t has occurrence λ in s, and $I[t] \leq I[t']$, then I[s[t]] implies I[s[t']] when m(λ)=1, and I[s[t']] implies I[s[t]] when m(λ) = -1.

Here, $\leq$ is the interpretation of ordering symbol $\leq$ under I.

Proof: by induction on the structure of s.

Note: m(λ)=0 means that the subterm occurrence in λ influences the value of its superterm irregularly, so the lemma doesn't hold for m(λ)=0.

Now, we will give the main inference rules of the general property inheritance theory.

Assume a given **m**, the property generalization rule is

(G_m):

$$\frac{C_1 \vee P[\lambda \leftarrow s],\ t \leq u \vee C_2}{(C_1 \vee C_2 \vee P[\lambda \leftarrow u])\sigma}$$

where σ =mgu(s,t) and m(λ)=1;

the property specialization rule is

(S_m):

$$\frac{C_1 \vee P[\lambda \leftarrow s],\ t \leq u \vee C_2}{(C_1 \vee C_2 \vee P[\lambda \leftarrow t])\sigma}$$

where σ =mgu(s,u) and m(λ)=-1.

Note:
The inequality atom $t \leq u$ is applied only to atoms (i.e. positive literals) P in the rules above.

Let $\mathbf{R}_m = \mathbf{R}_0 \cup \{(A), (G_m), (S_m)\}$. Following lemma 4.1, we can easily come to the soundness lemma as follows:

Lemma 4.2:

If the empty clause can be derived from a set S of clauses by the rules in $\mathbf{R}_m$, then S is π_m -unsatisfiable.

Unfortunately, the completeness theorem of $\mathbf{R}_m$ doesn't hold, as is shown by the following example.

Example 4.2:

Assume our language has two constant symbols a, b and an unary function symbol f, and m is m(f,1)=1 and undefined else where, then the set S ={ $a \leq b$, $\sim(f(a) \leq f(b))$} is obviously π_m -unsatisfiable, but we have no way to derive the empty clause since (**G_m**), and (**S_m**) are only applicable to positive literals.

There are two possible solutions to this problem. The first is to include additional axioms into our theory.

Define the weak functionally reflexive axioms as follows:

WFR= $\{f(x_1,\ldots,x_n) \leq f(x_1,\ldots,x_n) \mid f \in F$, arity(f)=n$\}$, then we have:

Theorem 4.3:

A set S of clauses is π_m -unsatisfiable iff the empty clause can be derived from S $\cup$ **WFR** $\cup$ $\{x = x, x \leq x\}$ by the rules in $\mathbf{R}_m$.

Although the theorem holds, it is quite difficult to prove it since we don't require the existence functionally reflexive axioms for equality. Therefore, we need the technique suggested by Peterson[6] to prove theorem 4.3 even though we include **WFR** in our axiom set. Because of the complexity of Peterson's technique, we would prefer to use functionally reflexive axioms rather than the weak ones.

Let $\mathbf{R}_m$' be $\mathbf{R}_m$ with paramodulation being replaced by hyperparamodulation[2] and **FR** denote the set of functionally reflexive axioms, then we have:

Theorem 4.4:

A set S of clauses is π_m -unsatisfiable iff the empty clause can be derived from S $\cup$ **FR** $\cup$ $\{x = x, x \leq x\}$ by the rules in $\mathbf{R}_m$'.

The theorem can be proved by following the method suggested by Slage[8].

Note that **WFR** can be derived from **FR** and $x \leq x$ by hyperparamodulation, so it is unnecessary to include **WFR** into our axiom set, and since hyperparamodulation is a restricted form of paramodulation, it is more efficient.

Coming back to example 4.2, we can see the axioms in **WFR** (or **FR**) are indeed used in the derivation of the empty clause.

Example 4.2(continued):

Let P in rule $(\mathbf{G}_m)$ be $f(x) \leq f(x)$, and λ be $(\leq,2)(f,1)$, then apply the rule with $\sigma = \{x \leftarrow a\}$ to $\{f(x) \leq f(x), a \leq b\}$, we get $f(a) \leq f(b)$. This, resolved with $\sim(f(a) \leq f(b))$, produces the empty clause. In a similar way, we can set $\lambda = (\leq,1)(f,1)$, $\sigma = \{x \leftarrow b\}$ and apply $(\mathbf{S}_m)$ to get the same result.

Another approach to solve the completeness issue is to augment our rule set without addition of WFR (or FR).

The following rules are contraposition of $(\mathbf{G}_m)$ and $(\mathbf{S}_m)$ respectively.

$(\mathbf{CG}_m)$:

$$\frac{C_1 \vee \sim P[\lambda \leftarrow s],\ t \leq u \vee C_2}{(C_1 \vee C_2 \vee \sim P[\lambda \leftarrow t])\sigma}$$

where $\sigma = \mathrm{mgu}(s,u)$ and $m(\lambda) = 1$;

the property specialization rule is

$(\mathbf{CS}_m)$:

$$\frac{C_1 \vee \sim P[\lambda \leftarrow s],\ t \leq u \vee C_2}{(C_1 \vee C_2 \vee \sim P[\lambda \leftarrow u])\sigma}$$

where $\sigma = \mathrm{mgu}(s,t)$ and $m(\lambda) = -1$;

These two rules can also be absorbed into $(\mathbf{G}_m)$ and $(\mathbf{S}_m)$ by extending the definition of subterm occurrences in words to literals and require $m(\sim,1) = -1$ by considering ~ as an unary operator.

Now, Let $\mathbf{AR}_m = \mathbf{R}_m \cup \{ (\mathbf{CG}_m), (\mathbf{CS}_m)\}$, then the completeness theorem can be reformulated in the following way.

Theorem 4.5:

A set S of clauses is π_m -unsatisfiable iff the empty clause can be derived from S $\cup$ $\{x = x, x \leq x\}$ by the rules in $\mathbf{AR}_m$.

5. Conclusion

To compensate for the inadequacy of the classical approach and partial order logic on reasoning about inheritance hierarchies, we propose the inference rules for simple and general property inheritance theory in this paper.

In this section, we show how we have achieved our claimed goal, i.e. to encode the most regular information about function and predicate symbols in the inference rules and meanwhile keep our language flexible enough to express the other irregular information.

Given a partial order logic language $L_p = \{V, C, F_n(n>0), P_n(n>0)\}$ and a set of clauses S_p, let L_g be L_p with equality and ordering symbols but considering C as an unordered set, and the m.i.f. **m** associated with L_g is as follows:

$m(P,i)=-1$, if $P \in \cup_{n>0} P_n$ and $1 \leq i \leq arity(P)$,

$m(f,i)=1$, if $f \in \cup_{n>0} Mon\_F_n$ and $1 \leq i \leq arity(f)$,

$m(f,i)=0$, if $f \in \cup_{n>0} Non\_F_n$ and $1 \leq i \leq arity(f)$.

Let $S_g = S_p \cup \{ c_1 \leq c_2 \mid c_1 \leq c_2$ in the ordered set $C\}$, then, obviously, we have:

Theorem 5.1:

S_p is unsatisfiable under partial order semantic structures iff S_p is π_m -unsatisfiable, where **m** is defined as above.

Note that the ordering no C is irregular information, so we translate it into clause set instead of rule set. This also has the advantage of not making the meaning of constant symbols fixed.

The theorem shows that our theory has at least the same expressive power as that of the partial order logic, and as we have noticed in section 2, there are some sentences (e.g. those represent the interactions between ordering predicate symbol and other ones) which can't be expressed by the partial order logic, but are obviously expressible as the first order axioms in our theory. Thus, our theory is indeed more expressible and flexible than partial order logic.

Moreover, we don't represent everything in the clause set as the classical approach. In fact, the information about function and predicate symbols encoded into an m.i.f can be handled by our inference rules.

References:

[1]Ait-Kaci, H. and Smolka, G., Inheritance Hierarchies: Semantics and Unification, Journal of Symbolic Computation (1989), Special Issue on Unification.

[2] Chang, C.L. and Lee, R.C.T., Symbolic Logic And Mechanical Theorem Proving, Academic Press, New-York,1973.

[3]Chen, W. and Warren, D.S., Partial Order Logic, Unpublished Manuscript, Department of Computer Science, State University of New York at Stony Brook, NY 11794, 1990

[4]Etherington, D., Formalizing Nonmonotonic Reasoning Systems, AI 31,1987.

[5]Etherington, D. and Reiter, R., On Inheritance Hierarchies with Exceptions, Proceedings AAAI-83, Washington, D.C.,1983.

[6]Peterson, G.E.,A Technique for Establishing Completeness Results in Theorem Proving with Equality, SIAM J. Comput, Vol. 12, No.1, February 1983.

[7]Stickel, M.E., Automated Deduction by Theory Resolution, J. of Automated Reasoning, Vol. 1 No.4 1985.

[8]Slage, J.R., Automatic Theorem Proving with Built-in Theories Including Equality, Partial Ordering, and Sets., JACM, Vol. 19, No. 1, Jan. 1972.

[9]Touretzky, D.S., Implicit Ordering of Defaults in Inheritance Systems, Proceedings AAAI-84, Austin, TX. 1984.

[10]Chang, Chin-Liang, Introduction to Artificial Intelligence Techniques, JMA Press, Austin, TX.

[11]Genesereth, M.R., Nilsson, N. J., Logical Foundations of Artificial Intelligence, Morgan Kaufmann Publishers, CA. 1986.

[12]Lloyd, J.W., Foundations of Logic Programming, Springer-Verlag, New York