PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1988,
edited by ICOT.

Generating Rules with Exceptions

Jun ARIMA

ICOT Research Center, Institute for New Generation Computer Technology,
4-28, Mita 1-Chome, Minato-ku, Tokyo 108 Japan
Phone: +81-3-456-4365, Csnet:arima%icot.jp@relay.cs.net
Junet: arima@icot.junet

ABSTRACT

Commonsense knowledge varies according to the characteristics of the person who possesses it. When we consider an intelligent system which makes commonsense reasoning, we can not ignore the cost of changing commonsense knowledge according to the circumstances. Commonsense knowledge must be acquired.

This paper describes the first step to formalization of the whole process from acquiring commonsense knowledge to doing commonsense reasoning using it.

1 INTRODUCTION

When we lack information, commonsense reasoning will often lead us to useful conclusions by making use of *default assumptions* that hold generally. For instance, we can jump to the conclusion, "Tweety flies", given the fact, "Tweety is a bird", using a default assumption, "Birds *normally* fly". Major objectives of the research of nonmonotonic reasoning are to formalize such commonsense reasoning and to realize intelligent systems that can use commonsense knowledge. This kind of research usually starts from the assumption that default assumptions are given. However, default assumptions vary widely according to the nationality, class and temperament of the person who possesses them as well as depending on time and place. Changing the set of default assumptions in an intelligent system according to the various circumstances in which the system works would probably be very expensive, so we should also try to develop an intelligent system that learns commonsense knowledge by itself. This paper attempts to formalize the whole process from acquiring default assumptions to doing commonsense reasoning using them.

In formalizing the whole process of such default reasoning, the problems that arise and our solutions to them are as follows.

(1) Representation of default assumption

Default assumptions change depending on a (sub-)domain. Consider a bird-world which consists only of birds. In this world, a default assumption with respect to flying will be such a statement as "x normally flies ('Fly(x)')", then we say, 'Fly' has *positive directivity* in the bird-world in the sense that we assign 'true' as truth values to 'Fly(x)' if possible.

Now, consider a penguin-world which consists only of penguins. In this world, the default assumption will be "x normally does not fly ('$\neg$Fly(x)')", then we say, 'Fly' has *negative directivity* in the contrary sense. The directivity of 'Fly' has to be changed according to each domain.

How do we represent such default assumptions? For this purpose, we propose a general form of *(parallel) circumscription* (McCarthy 1980, Lifschitz 1985), *partially directional (Pd-)circumscription*. Circumscription makes the extension of certain predicates (symbols) closer to the extension of 'false' (its extension means empty set) as far as possible, whereas Pd-circumscription makes the extension of properties closer to the extensions of our intended predicates in each (sub-)domain. Pd-circumscription is defined in the second section.

(2) Representation of criterion for directivity of the concerning property

We must still choose positive or negative directivity of the concerning property in each domain. How should we judge the direction? Our treatment is very natural. If entities that are shown to have the concerning property are *much more (much less)* than entities that are shown not to have the concerning property in some domain, the directivity of the property should be positive (negative) in the domain. That is, if many more birds fly than do not, we consider that birds normally fly. This is the key idea.

For this treatment, we introduce an binary order over predicate names, called *the surpassing relation*. The surpassing relation represents whether entities that satisfy a predicate are much more than entities which satisfy another predicate. Depending on the relations, we select the directivity of the concerning property. The entities *that are shown to have a property* are interpreted as elements of minimum extension of the property under given knowledge.

The third section presents these details, and then constructs our objective form which generates default assumptions and makes commonsense reasoning using them. We call this form *the majority generalization*, which is expressed a second-order formula. The fourth section explores the possibility of collapsing it into first-order logic and the fifth describes how to use it and how it works.

2 PARTIALLY DIRECTIONAL CIRCUMSCRIPTION

As we have already described the objective of this paper and the problems that arise, we would like to focus on explanation of our solutions in the later sections. The main problem to solve in this section is to provide a way to represent default assumptions that are allowed to vary depending on a sub-domain (sub-class). We can use circumscription to provide such a way.

J.McCarthy provides circumscription (McCarthy 1980, 1986) as a form of non-monotonic reasoning. Formula circumscription is one version of circumscription. It is a general formulation in that a wff is minimized, whereas the earlier form minimizes some predicates. Formula circumscription is a powerful way to express non-monotonic reasoning, but in order to use it, we must specify many predicate parameters. The wff to be minimized is the most important one to be given because to give the wff means to characterize reasoning by giving predicates the intended interpretation. However, unfortunately, it is the most difficult, because there are no constraints to decide it and the problem is left entirely in our hands. Hence, how to specify the wff is a problem. In this paper, we take the way of extending the earlier form, called predicate circumscription.

Consider predicate circumscription, and consider what character the formulation has. Predicate circumscription can make the extension of property P minimal. When we consider that it makes the extension of P closer to the extension of 'false' as far as possible, we can extend predicate circumscription for our objectives. That is, what we want is a formula which, according to the characteristics of the sub-domain, can give an extension of P so that the extension will be closest to the extensions of our intended predicates. With such a formulation, even if new information contradicts an old theorem, the theorem would be revised because the formulation does not fix the extension of P to our intended extension, but simply makes the extension closest to ours. Now we get our form.

In this paper we simply write x instead of a tuple of finite terms for brevity. By n-ary predicate, we mean an expression, $\lambda x.(\alpha(x))$, where x is a tuple of n variables and $\alpha(x)$ is a wff in which x occurs free and no other variables occur free. That is, a predicate is obtained from a formula by λ-abstracting all of the free variables in it.

Let P be a tuple of distinct predicate symbols, $P1,\cdots,Pn$, and Ψ a tuple of predicates, $\Psi1,\cdots,\Psi n$, where Pi and Ψi have the same arity. $[\Psi/P]$ means a substitution, representing $[\Psi1/P1,\cdots,\Psi n/Pn]$ and usually abbreviated $[\Psi]$. We write $\alpha(x)[\Psi/P]$ for the result of replacing simultaneously each occurrence Pi in $\alpha(x)$ by Ψi. And $\forall x.(P(x) \equiv \Psi(x))$ means $\forall x.(P(x) \supset \Psi(x)) \wedge \forall x.(P(x) \subset \Psi(x))$.

Definition [*partially directional circumscription (Pd-circumscription)*].

Let P and Z be tuples of distinct predicate symbols and disjoint each other, and Φ and Λ be tuples of predicates. Let A be a formula. The *partially directional circumscription of* P *to* Λ *inside* Φ with variable Z is

$$A[P,Z] \wedge \forall p,z.(A[p,z] \wedge \wedge_i \forall x.(\Phi i(x) \supset ((Pi(x) \equiv \Lambda i(x)) \supset (pi(x) \equiv \Lambda i(x)))) \supset \wedge_i \forall x.(\Phi i(x) \supset (Pi(x) \equiv pi(x)))), \quad (2.1)$$

where Pi, Λi, Φi and pi are elements of P, Λ, Φ and p respectively. This formula is denoted by Pd-circum(A; P ~ Λ / Φ ; Z).

It asserts that the extension of P cannot be made closer to the extension of Λ inside an extension of Φ, even if allowing Z to vary, while they satisfy A. That is, intuitively, it can, under A, give P the closest extension to Λ's extension with allowing Z to vary in the Φ-domain. In this sense, we say P *is directional to* Λ *inside* Φ *in* A.

In this paper, ⊥ represents a tuple of property, λx.(false), with respect to which for all tuples of entities the false value is assigned constantly, and similarly, ⊤ represents a property, λx.(¬false), with respect to which the true value is assigned constantly.

Proposition 1.
1) Pd-circum(A; P ~ ⊥ / ⊤ ; Z) ≡ Circum(A; P ; Z), where Circum(A; P ; Z) is the parallel circumscription of P in A.
2) If P consists of a single element, Pd-circumscription is a specialized formulation of formula circumscription.

Proofs.
1) By predicate calculus.
2) Pd-circumscription is equivalent to the formula circumscription that minimizes the wff ¬(Φ(x) ⊃ (P(x) ≡ Λ(x))).

Proposition 1 says that parallel circumscription (which predicate circumscription is a special form of) is to make some predicates directional to false in a whole domain.

3 MAJORITY GENERALIZATION: A FORM OF GENERATING RULES WITH EXCEPTIONS

Now, we have a tool for representation of default assumptions. In this section, we explore the way to generate adequate assumptions, and then a form of generating rules with exceptions is proposed.

Consider a property, to fly (predicate, 'Fly'). For any domain, we can consider the following three cases; the individuals of the domain 1) generally fly, 2)generally do not, or 3) otherwise. The case that all individuals fly (do not fly) is a special case of 1) (2)). It would be adequate to generate assumptions which have the positive directivity of 'Fly' in the case of 1) and negative in the case of 2). In the case of 3), it would not be adequate to generate assumptions. To represent such cases, we still need some preparations.

We assume that the syntax of the underlying language includes names for predicates. That is, if α is a predicate, let our language include a constant term, α, as the predicate name. Now, we introduce a comparative relation over predicate (names), called *the surpassing relation* which is denoted by '>>'. When 'α >> β' holds, we say that α *surpasses* β. Its intended meaning is that the number of entities that satisfying property, α, is much greater than the number of entities that satisfy the other, β. Based on this intended meaning, let the following sentence hold with respect to the predicate and its predicate name.

For all predicates α, β and γ (and for their predicate names):

I $(\alpha >> \beta) \wedge (\beta >> \gamma) \supset (\alpha >> \gamma)$ (3.1)
II $\neg(\alpha >> \alpha)$ (3.2)
III $(\alpha >> \beta) \wedge \forall x.(\alpha(x) \supset \gamma(x)) \supset (\gamma >> \beta)$ (3.3)
IV $(\alpha >> \beta) \wedge \forall x.(\gamma(x) \supset \beta(x)) \supset (\alpha >> \gamma)$ (3.4)

The following two formulas are not essential, but here, we consider these as axioms.

V a. $\alpha \neq \bot \supset (\alpha >> \bot)$ (3.5)
VI b. $\alpha \neq \top \supset (\top >> \alpha)$, (3.6)

where α = β denotes ∀x.(α(x) ≡ β(x)) and α ≠ β denotes ¬∀x.(α(x) ≡ β(x)). The conjunction of these formulas (3.1)~(3.6) is represented as 'Surp'. 'Surp' is handled as an axiom.

Other axioms with respect to surpassing relation '>>' may be given by using a function to count up entities that satisfy certain conditions and by using some adequate evaluate functions for '>>', or may be given directly based on our intuition.

Now we introduce a formulation on generating rules with exceptions (default assumptions). Its intuitive idea is that, in a certain domain Φ,

entities normally have a certain property P if, as far as we know, there are much more entities that satisfy the property than do not. And we assume that a set of entities that we know to satisfy a predicate P corresponds to the minimum extension of P which satisfy given knowledge. To brief description, the formula,

$$A[\llcorner P \lrcorner] \wedge \forall p.\,(A[p] \supset \forall x.\,(\llcorner P \lrcorner(x) \supset p(x))), \quad (3.7)$$

is represented by '$\mathrm{Min}_P(\llcorner P \lrcorner)$'.

$\mathrm{Min}_P(\llcorner P \lrcorner)$ means that $\llcorner P \lrcorner$ has a extension of P which is smaller than the extension of any predicate p satisfying the conditions satisfied by P. That is, intuitively, $\llcorner P \lrcorner$ has the minimum extension of P. In this sense, we say $\llcorner P \lrcorner$ is the *minimum candidate of* P in A when $\mathrm{Min}_P(\llcorner P \lrcorner)$ holds. Conversely,

$$A[\ulcorner P \urcorner] \wedge \forall p.\,(A[p] \supset \forall x.\,(p(x) \supset \ulcorner P \urcorner(x))). \quad (3.8)$$

is represented by '$\mathrm{Max}_P(\ulcorner P \urcorner)$' and we say the *maximum candidate of* P in A when $\mathrm{Max}_P(\ulcorner P \urcorner)$ holds.

Then, *the majority generalization* of P *inside* Φ *with variable* Z *in* A is

$$A[P]$$
$$\wedge\, \forall \llcorner P \lrcorner, \ulcorner P \urcorner.(\,\mathrm{Min}_P(\llcorner P \lrcorner) \wedge \mathrm{Max}_P(\ulcorner P \urcorner) \wedge \lambda x.(\Phi(x) \wedge \llcorner P \lrcorner(x)) >> \lambda x.(\Phi(x) \wedge \neg \ulcorner P \urcorner(x)) \supset \text{Pd-circum}(\,A\,;P \sim \top / \Phi; Z\,)\,)$$

$$\wedge\, \forall \llcorner P \lrcorner, \ulcorner P \urcorner.(\,\mathrm{Min}_P(\llcorner P \lrcorner) \wedge \mathrm{Max}_P(\ulcorner P \urcorner) \wedge \lambda x.(\Phi(x) \wedge \neg \ulcorner P \urcorner(x)) >> \lambda x.(\Phi(x) \wedge \llcorner P \lrcorner(x)) \supset \text{Pd-circum}(\,A\,;P \sim \bot / \Phi; Z\,)\,). \quad (3.9)$$

This formula is denoted by 'Major(A; P / Φ; Z)'. We use this majority generalization with axioms on the surpassing relation, that is, we consider deduction from 'Surp $\wedge$ Major(A; P / Φ; Z)'.

$\lambda x.(\,\Phi(x) \wedge \llcorner P \lrcorner(x))$ expresses the minimum set of entities that exist inside Φ and satisfy P. $\lambda x.(\,\Phi(x) \wedge \neg \ulcorner P \urcorner(x)\,)$ expresses the minimum set of entities that exist inside Φ and do not satisfy P. That is, (3.9) declares that under A, P should be directional to $\top$ inside Φ (P has the positive directivity inside Φ) if there are more entities that satisfy P than not inside Φ so far as we know. And conversely, (3.9) also declares that P should be directional to $\bot$ inside Φ (P has the negative directivity inside Φ) if there are more entities that do not satisfy P than satisfy inside Φ so far as we know. Of course, it means that entities that satisfy Φ (do not) have a property P normally, so it is a declaration that anything that has a property Φ should be considered (not) to have the property P if there is nothing in knowledge A that prevents it from doing so.

Example 1:

In a database, DB, three birds: Tweety, Jack and P-suke, are registered, and the information, "P-suke cannot fly" is given. This may be represented as follows:

$$\mathrm{Bird}(\text{tweety}) \wedge \mathrm{Bird}(\text{jack}) \wedge \mathrm{Bird}(\text{p-suke}) \wedge \neg \mathrm{Fly}(\text{p-suke}). \quad (E1.1)$$

Also, assume that enough information on surpassing relation '>>' is given, for instance, assume the following knowledge:

$$\lambda x.(x = \text{tweety}) << \lambda x.(x = \text{jack} \vee x = \text{p-suke}) \wedge \lambda x.(x = \text{jack}) << \lambda x.(x = \text{p-suke} \vee x = \text{tweety}) \wedge \lambda x.(x = \text{p-suke}) << \lambda x.(x = \text{tweety} \vee x = \text{jack}). \quad (E1.2)$$

Let A be such information mentioned above and consider Surp $\wedge$ Major(A; Fly / Bird). In this case, $\bot$ clearly satisfies the condition of $\llcorner \mathrm{Fly} \lrcorner$ as the minimum candidate of 'Fly', and $\lambda x.(\neg x = \text{p-suke})$ for $\ulcorner \mathrm{Fly} \urcorner$ as the maximum candidate of 'Fly', that is, $\mathrm{Min}_{\mathrm{Fly}}(\bot)$ and $\mathrm{Min}_{\mathrm{Fly}}(\bot)$ hold[1]. Therefore, using (E1.1),

$$\lambda x.(\,\mathrm{Bird}(x) \wedge \llcorner \mathrm{Fly} \lrcorner(x)) = \bot, \quad (E1.3)$$

$$\lambda x.(\,\mathrm{Bird}(x) \wedge \neg \ulcorner \mathrm{Fly} \urcorner(x)) = \lambda x.(x = \text{p-suke}). \quad (E1.4)$$

Now from (3.5)

$$\lambda x.(x = \text{p-suke}) >> \bot \quad (E1.5)$$

follows. So, using (E1.3), (E1.4), (3.3), (3.4) and Major(A; Fly / Bird), it gives Pd-circum(A ; Fly $\sim$ $\bot$ / Bird), that is,

$$A[\mathrm{Fly}] \wedge \forall p.(A[p] \wedge \forall x.(\mathrm{Bird}(x) \supset (\neg \mathrm{Fly}(x) \supset \neg p(x)))$$

[1] How to compute these generally is beyond the scope of this paper. However, the next section will partly solve the problem.

$$\supset \forall x.(Bird(x) \supset (Fly(x) \equiv p(x)))). \quad (E1.6)$$

Here, substituting $\bot$ for p, we obtain

$$\forall x.(\, Bird(x) \supset \neg Fly(x)\,). \quad (E1.7)$$

Recall the three cases mentioned in the first part of this section. The above case corresponds to case 2) and (E1.7) is a result of the negative directivity of 'Fly'.

Also, (E1.7) shows generalization of knowledge. From (E1.7) and (E1.1) we can see that both Tweety and Jack may be unable to fly.

Now, we add new information to the DB, "Tweety can fly (Fly(tweety))". Then $\llcorner Fly \lrcorner$ is $\lambda x.(x = tweety)$ and $\ulcorner Fly \urcorner$ is unchanged. Under this circumstance, we cannot obtain either '$\lambda x.(\, Bird(x) \wedge \llcorner Fly \lrcorner(x)) >> \lambda x.(\, Bird(x) \wedge \neg \ulcorner Fly \urcorner(x))$' or '$\lambda x.(\, Bird(x) \wedge \neg \ulcorner Fly \urcorner(x)) >> \lambda x.(\, Bird(x) \wedge \llcorner Fly \lrcorner(x))$'. Hence, (E1.7) is not a theorem of the DB any more. This case corresponds to case 3). However, if the DB also knows "Jack can fly (Fly(jack))", the theorems of DB will change more dramatically. In this case, using (E1.2) $\lambda x.(\, Bird(x) \wedge \llcorner Fly \lrcorner(x)) >> \lambda x.(\, Bird(x) \wedge \neg \ulcorner Fly \urcorner(x))$ follows, that is, this case corresponds to c). Therefore, the directivity of 'Bird' changes and using Major(A $\wedge$ Fly(tweety) $\wedge$ Fly(jack); Fly / Bird),

$$\forall x.(\, Bird(x) \supset (Fly(x) \equiv \neg x = p\text{-suke}) \quad (E1.8)$$

is obtained. This means "P-suke is the only bird that cannot fly," and P-suke comes to be considered to be abnormal with respect to flying inside the bird-world.

4 CONSIDERATIONS ON A FIRST-ORDER FORMULATION

An discontented point of the majority generalization is on computational aspects which originate from its expression by second order logic. Let us try to collapse it into first order logic, giving some constraints to given formulas A. The way this paper treats the problem is to restrict A to being a formula, called *the symmetrically solitary formula*, that has (speaking intuitively) both the only minimal model (that is, *minimum model*) and the only maximal model (*maximum model*) with respect to P.

Theorem 1.a.

Let A be a given formula and $\llcorner P \lrcorner$ be a tuple of predicates such that no predicate in P occurs in $\llcorner P \lrcorner$, and that $A[\llcorner P \lrcorner] \wedge \forall x.(\llcorner P \lrcorner(x) \supset P(x))$ follows from A, then

$$\forall p.(\, Min_P[p] \equiv \forall x.(p(x) \equiv \llcorner P \lrcorner(x))\,). \quad (4.1)$$

Proof.

From the assumption, $A \vdash A[\llcorner P \lrcorner] \wedge \forall x.(\llcorner P \lrcorner(x) \supset P(x))$ for some $\llcorner P \lrcorner$. So, $A[p] \vdash \forall x.(\llcorner P \lrcorner(x) \supset P(x))[p]$ (See (Kleene 1971)). Here, no predicate in P occurs in $\llcorner P \lrcorner$, therefore, $\llcorner P \lrcorner[p] = \llcorner P \lrcorner$. So, $A[p] \vdash \forall x.(\llcorner P \lrcorner(x) \supset p(x))$. Therefore, $\vdash \forall p.(A[p] \supset \forall x.(\llcorner P \lrcorner(x) \supset p(x))$. From this and $A \vdash A[\llcorner P \lrcorner]$, $Min_P[\llcorner P \lrcorner]$ holds.

(Right to Left) Obvious from the fact that $Min_P[\llcorner P \lrcorner]$ holds.

(Left to Right) Assume that $Min_P[p']$ holds for some p'. From the fact that $Min_P[\llcorner P \lrcorner]$ holds, $\forall p.(A[p] \supset \forall x.(\llcorner P \lrcorner(x) \supset p(x)))$ holds. Substituting p' for p, $A[p'] \supset \forall x.(\llcorner P \lrcorner(x) \supset p'(x))$. From the assumption that $Min_P[p']$ holds, A[p'] follows. Therefore, $\forall x.(\llcorner P \lrcorner(x) \supset p'(x))$ holds. Similarly, $\forall x.(p'(x) \supset \llcorner P \lrcorner(x))$ holds.

Theorem 1.b.

Let A be a given formula and $\ulcorner P \urcorner$ be a tuple of predicates such that no predicate in P occurs in $\ulcorner P \urcorner$ and that $A[\ulcorner P \urcorner] \wedge \forall x.(\ulcorner P \urcorner(x) \supset P(x))$ follows from A. Then

$$\forall p.(\, Max_P[p] \equiv \forall x.(p(x) \equiv \ulcorner P \urcorner(x))\,). \quad (4.2)$$

Proof. Similar to that of Theorem 1.a.

Theorem 1 shows that if we can find $\llcorner P \lrcorner$ and $\ulcorner P \urcorner$ that satisfy the condition of theorem 1, we can leave out second order formulas, $Min_P[\llcorner P \lrcorner]$ and $Max_P[\ulcorner P \urcorner]$, and the quantifier $\forall \llcorner P \lrcorner$, $\ulcorner P \urcorner$. from (3.9). How to find such $\llcorner P \lrcorner$ and $\ulcorner P \urcorner$, for a certain class of given formulas, A, can be easily shown.

Lifschitz shows that if A[P] is transformed into a certain class of formula, called *solitary formula* (Lifschitz 1985), circumscription of P can be collapsed into a first order sentence. Though we need some refinement, we can basically use his results.

Definition [*symmetrically solitary formula*].

A formula, A, is an *symmetrically solitary formula with respect to* P if A can be transformed into the following form,

$$U \wedge \forall x.(L(x) \supset P(x)) \wedge \forall x.(P(x) \supset G(x)), \quad (4.3)$$

where no predicate in P occurs in U, L(x) or G(x).

Lemma 1.
Let a formula, A, be a symmetrically solitary formula with respect to P, $U \wedge \forall x.(L(x) \supset P(x)) \wedge \forall x.(P(x) \supset G(x))$. Then, L is the minimum and G is the maximum candidate of P, that is,

$$\mathrm{Minp}[L] \wedge \mathrm{Maxp}[G] \quad (4.4)$$

holds.

Proof.
L and G satisfies the condition of Theorem 1.a. and Theorem 1.b. respectively.

Theorem 2.
Let a formula A be a symmetrically solitary formula with respect to P such that $U \wedge \forall x.(L(x) \supset P(x)) \wedge \forall x.(P(x) \supset G(x))$. Then,

$$\begin{aligned} &\mathrm{Major}(A; P / \Phi) \equiv \\ &\quad A \\ &\wedge (\lambda x.(\Phi(x) \wedge L(x)) >> \lambda x.(\Phi(x) \wedge \neg G(x)) \\ &\qquad \supset \forall x.(\Phi(x) \supset (P(x) \equiv G(x)))) \\ &\wedge (\lambda x.(\Phi(x) \wedge \neg G(x)) >> \lambda x.(\Phi(x) \wedge L(x)) \\ &\qquad \supset \forall x.(\Phi(x) \supset (P(x) \equiv L(x)))) \end{aligned} \quad (4.5)$$

Proof. Using Proposition 1 and the result which Lifschitz shows (Lifschitz 1985), we obtain

$$\text{Pd-circum}(P\sim \perp / \top) \equiv A \wedge \forall x.(P(x) \equiv L(x)).$$

Using expansion of this result and Lemma 1, we obtain this theorem.

Proposition 2.
Let A be a symmetrically solitary formula with respect to P and Φ be a predicate which does not contain P. Then, the majority generalization of P inside Φ without variables is also a symmetrically solitary formula.

Proof. If both A and B are symmetrically solitary formulas, $A \wedge B$ is also a symmetrically solitary formula. From this fact and Theorem 2, we obtain this proposition.

Example 1 (reviewed):
A is a symmetrically solitary formula with respect to Fly, because A can be transformed into the following form,

$$\begin{aligned} &U \wedge \forall x.(\perp \supset \mathrm{Fly}(x)) \\ &\quad \wedge \forall x.(\mathrm{Fly}(x) \supset \neg x = \text{p-suke}), \end{aligned} \quad (E1.8)$$

where 'Fly' does not occur in U. Using Lemma 1,

$$\mathrm{Min}_{\mathrm{Fly}}(\perp) \wedge \mathrm{Max}_{\mathrm{Fly}}(\lambda x.(\neg x = \text{p-suke})) \quad (E1.9)$$

holds. Then, from Theorem 1 and 'Surp',

$$\begin{aligned} &\mathrm{Major}(A; \mathrm{Fly} / \mathrm{Bird}) \equiv \\ &\quad A \\ &\wedge (\lambda x.(\mathrm{Bird}(x) \wedge \perp) >> \lambda x.(\mathrm{Bird}(x) \wedge x = \text{p-suke}) \\ &\qquad \supset \forall x.(\mathrm{Bird}(x) \supset (\mathrm{Fly}(x) \equiv \neg x = \text{p-suke}))) \\ &\wedge (\lambda x.(\mathrm{Bird}(x) \wedge x = \text{p-suke}) >> \lambda x.(\mathrm{Bird}(x) \wedge \perp) \\ &\qquad \supset \forall x.(\mathrm{Bird}(x) \supset (\mathrm{Fly}(x) \equiv \perp))) \\ &\equiv A \wedge \forall x.(\mathrm{Bird}(x) \supset \neg \mathrm{Fly}(x)). \end{aligned} \quad (E1.10)$$

So we can mechanically obtain the sentence, "birds can not fly", from the majority generalization of Fly and Surp.

5. AN APPLICATION TO IS-A HIERARCHY

Consider a simple example of an is-a hierarchical system.

Example 2.
Let A be

$$\begin{aligned} &\mathrm{Sparrow} \Rightarrow \mathrm{Bird} && \text{(E2.1)} \\ &\wedge \mathrm{Penguin} \Rightarrow \mathrm{Bird} && \text{(E2.2)} \\ &\wedge \mathrm{Bird} \Rightarrow \mathrm{Animate} && \text{(E2.3)} \\ &\wedge \mathrm{Reptilian} \Rightarrow \mathrm{Animate} && \text{(E2.4)} \\ &\wedge \forall x.(\mathrm{Sparrow}(x) \supset \mathrm{Bird}(x)) && \text{(E2.5)} \\ &\wedge \forall x.(\mathrm{Penguin}(x) \supset \mathrm{Bird}(x)) && \text{(E2.6)} \\ &\wedge \forall x.(\mathrm{Bird}(x) \supset \mathrm{Animate}(x)) && \text{(E2.7)} \\ &\wedge \forall x.(\mathrm{Reptilian}(x) \supset \mathrm{Animate}(x)) && \text{(E2.8)} \\ &\wedge \neg \exists x.(\mathrm{Sparrow}(x) \wedge \mathrm{Penguin}(x)) && \text{(E2.9)} \\ &\wedge \neg \exists x.(\mathrm{Bird}(x) \wedge \mathrm{Reptilian}(x)) && \text{(E2.10)} \end{aligned}$$

$$\begin{aligned} &\wedge \forall x.(\mathrm{Sparrow}(x) \supset \mathrm{Fly}(x)) && \text{(E2.11)} \\ &\wedge \forall x.(\mathrm{Penguin}(x) \supset \neg \mathrm{Fly}(x)) && \text{(E2.12)} \\ &\wedge \forall x.(\mathrm{Reptilian}(x) \supset \neg \mathrm{Fly}(x)) && \text{(E2.13)} \end{aligned}$$

$$\begin{aligned} &\wedge \mathrm{Sparrow} >> \mathrm{Penguin} && \text{(E2.14)} \\ &\wedge \mathrm{Reptilian} >> \mathrm{Bird}, && \text{(E2.15)} \end{aligned}$$

where the intended meaning of binary relation $\Rightarrow$ is '*is-a*'.

Now, let us consider majority generalization of Fly inside each class of this hierarchical system. As we define the interpretation of Fly using the majority generalization inside each class, the obtained definition of Fly inside a class must influence the definition of Fly inside other classes that are obtained later. Therefore, the order in defining has significant meaning. In an *is-a* hierarchical system, we define Fly from lower class to upper (from leaves to root), and we can obtain natural results. For instance, in this example, if we define Fly inside Animate previously to inside Bird, we obtain

$$\forall x.\,(\mathrm{Animate}(x) \wedge \neg \mathrm{Sparrow}(x) \supset \neg \mathrm{Fly}(x)). \qquad \text{(E2.16)}$$

However, rather than (E2.16), we prefer

$$\forall x.\,(\mathrm{Animate}(x) \wedge \neg \mathrm{Bird}(x) \supset \neg \mathrm{Fly}(x)) \quad \text{(E2.17)}$$

as a result in which we generalize a rule with respect to Fly inside Bird previously and then using the rule obtained we generalize it inside Animate. (This result is illustrated latter.) Hence, we use the majority generalization in the following way:

$$\mathrm{Major}(\mathrm{Major}(\cdots \mathrm{Major}(A; P/C_1)\cdots ; P/C_{n-1}); P/C_n), \qquad (5.1)$$

where $\neg(C_i \Rightarrow C_j)\ (1 \leq j \leq i \leq n)$. (5.1)is denoted by 'Major(A; P / C_1, $\cdots$, C_{n-1}, C_n)'.

That is,

$$\mathrm{Major}(A; P/C_1, \cdots, C_{n-1}, C_n) = \mathrm{Major}(G_n; P/C_n), \qquad (5.2)$$

where $G_0 = A$, $G_i = \mathrm{Major}(G_{i-1}; P/C_{i-1})$. This formula, Major(A; P / C_1, $\cdots$, C_{n-1}, C_n), is called the *prioritized majority generalization*.

Example 2 (continued):

From the definition of prioritized directional generalization,

$$\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}, \mathrm{Animate}) \equiv \mathrm{Major}(\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}); \mathrm{Fly}/\mathrm{Animate}) \qquad \text{(E2.18)}$$

Here, A is a symmetrically solitary formula with respect to Fly, so, using Theorem 2 (the minimum candidate of Fly in A is Sparrow and the maximum candidate of Fly in A is $\lambda x.\neg(\mathrm{Penguin}(x) \vee \mathrm{Reptilian}(x))$), we can easily obtain the result,

$$\begin{aligned}&\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}) = \\ &\quad A \\ &\wedge (\lambda x.(\mathrm{Bird}(x) \wedge \mathrm{Sparrow}(x)) \\ &\quad >> \lambda x.(\mathrm{Bird}(x) \wedge (\mathrm{Penguin}(x) \vee \mathrm{Reptilian}(x))) \\ &\quad \supset \forall x.(\mathrm{Bird}(x) \supset \\ &\qquad (\mathrm{Fly}(x) \equiv \neg(\mathrm{Penguin}(x) \vee \mathrm{Reptilian}(x))))) \\ &\wedge (\lambda x.(\mathrm{Bird}(x) \wedge (\mathrm{Penguin}(x) \vee \mathrm{Reptilian}(x))) \\ &\quad >> \lambda x.(\mathrm{Bird}(x) \wedge \mathrm{Sparrow}(x)) \\ &\quad \supset \forall x.(\mathrm{Bird}(x) \supset \\ &\qquad (\mathrm{Fly}(x) \equiv \mathrm{Sparrow}(x)))) \\ &\equiv A \wedge \forall x.(\mathrm{Bird}(x) \wedge \neg \mathrm{Penguin}(x) \supset \mathrm{Fly}(x)) \end{aligned} \qquad \text{(E2.19)}$$

Similarly, the minimum candidate of Fly in Major(A; Fly / Bird) is $\lambda x.(\mathrm{Bird}(x) \wedge \neg \mathrm{Penguin}(x))$ and the maximum candidate of Fly is $\lambda x.\neg(\mathrm{Penguin}(x) \vee \mathrm{Reptilian}(x))$. Therefore,

$$\begin{aligned}&\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}, \mathrm{Animate}) \equiv \\ &\quad A \\ &\wedge \forall x.(\mathrm{Bird}(x) \wedge \neg \mathrm{Penguin}(x) \supset \mathrm{Fly}(x)) \\ &\wedge \forall x.(\mathrm{Animate}(x) \wedge \neg \mathrm{Bird}(x) \supset \neg \mathrm{Fly}(x)) \end{aligned} \qquad \text{(E2.20)}$$

holds. (See (E2.17).)

Now, we add a new axiom 'Bird(tweety)' to A and assume that we want to know whether tweety can fly or not. This new axiom gives no change to above results, (E2.20). Obviously, we can not obtain the answer to the question from (E2.20). In this case, we must introduce predicates which are allowed to vary. We consider the majority generalization of P inside Ci with allowing all predicates that represent subclasses (descendants) of Ci to vary. That is, in this example, we consider

$$\begin{aligned}&\mathrm{Major}(\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}; \mathrm{Penguin}, \mathrm{Sparrow}); \\ &\qquad \mathrm{Fly}/\mathrm{Animate}; \mathrm{Bird}, \mathrm{Reptilian}). \end{aligned} \qquad \text{(E2.21)}$$

With respect to 'tweety', it is sufficient to see

$$\mathrm{Major}(A; \mathrm{Fly}/\mathrm{Bird}; \mathrm{Penguin}, \mathrm{Sparrow}). \qquad \text{(E2.22)}$$

Similarly, it yields

$$\mathrm{Pd\text{-}circum}(A; \mathrm{Fly}\sim\top/\mathrm{Bird}; \mathrm{Penguin}, \mathrm{Sparrow}), \qquad \text{(E2.23)}$$

and

$$\forall x.(Bird(x) \supset Fly(x)) \wedge Fly(tweety) \qquad (E2.24)$$

follows. This shows that this formalization generates a default assumption and makes default reasoning using the assumption.

6 CONSIDERATION AND REMARKS

We have considered formalization of the whole process from acquiring commonsense knowledge to doing commonsense reasoning using it and proposed a form, called the majority generalization. We have also considered the possibility of collapsing the form into a first-order formula and show an application. We will try to apply this idea to the acquisition of various types of commonsense knowledge.

This research may suggest a new view to the field of concept learning (Mitchell 1977). One aim of concept learning is to obtain some properties which, for some set of entities, *all* entities have in common. Here, more generally, we can consider a certain type of learning, in which one aim is to obtain some properties which, for some set of entities, *most* entities possibly have in common.

We hope this research extends the sphere of interest of researchers in non-monotonic reasoning and serves as a new stimulus to machine learning.

ACKNOWLEDGEMENTS

I would like to thank Dr. Koichi Furukawa and Dr. Masayuki Numao for their useful comments. Also, I wish to express my gratitude to Dr. Kazuhiro Fuchi, Director of the ICOT Research Center, who provided me with the opportunity to pursue this research.

REFERENCES

[1] Kleene,S.C.: *Introduction to Metamathematics*, North-Holland, 1971, CH. VII.

[2] Lifschitz,V.: Computing circumscription, in: *Proceedings of Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA (1985) 121-127.

[3] McCarthy,J.: Circumscription - a form of non-monotonic reasoning, *Artificial Intelligence* 13 (1980) 27-39.

[4] McCarthy,J.: Application of circumscription to formalizing common-sense knowledge, *Artificial Intelligence* 28 (1986) 89-116.

[5] Mitchell,T.M.:Version Spaces: A Candidate Elimination Approach to Rule Learning, *Proceedings of Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Mass. (1977) 305-310.