人　工　知　能 79－5
（1991. 11. 26）

仮説に基づく意味ネットワークの管理法

佐藤 雅弘　　伊藤英則

名古屋工業大学

本稿では，信念に基づき知識間の性質を非単調多重継承する意味ネットワークの表現法とその管理法を提案する．この意味ネットワーク内では，常に成り立つと信じている知識と，常に成り立つとは信じていない知識とを区別して表現する．また，この意味ネットワーク内の上位知識の性質を下位知識へ継承する際は，その性質の否定はとりあえず信じない仮説推論方法をとる．さらに，意味ネットワークの非冗長性・無矛盾性の維持のための知識同化と知識削除，および，下位概念に共通する性質を上位概念に伝搬する知識調節を行なう知識管理方法について述べる．なお，これらの管理では意味ネットワーク内の推論履歴情報である根拠を用いる．

"Hypothesis-based Management of a Semantic Network."

Masahiro SATOH,　　Hidenori ITOH

Nagoya Institute of Technology

Gokiso, Syohwa-ku, Nagoya 466, Japan

This paper describes a semantic network based on belief and its management method for knowledge with nonmonotonic multiple-properties inheritance. In this semantic network, we separately express the knowledge which is always believed to be valid and the knowledge which isn't always believed to be valid. Higher-level properties which are inherited at the lower-level are not always believed to be valid as a hypothesis. The semantic network's management method consists of knowledge assimilation, knowledge deletion, and knowledge accommodation. Knowledge assimilation and knowledge deletion maintain the redundancies and inconsistencies of the semantic network. Knowledge acccommodation propogates properties which are common among lower knowledge levels to higher knowledge levels. Furthermore, the information of inference histories works efficiently in this management method.

1. はじめに

信念を含まない知識からなる知識ベース，および，仮説推論による知識ベースの知識同化 (knowledge assimilation)，知識削除 (knowledge deletion) と知識調節 (knowledge accommodation) による管理手法は知られている[1,2]．

本稿では，信念に基づく意味ネットワークの表現法とその意味ネットワークを知識ベースとし非単調多重継承を行なう管理法を提案する[3]．ここでは，非単調多重継承 (nonmonotonic multiple inheritance) とは複数の上位階層から例外を含む性質継承を行なうことをいう．この意味ネットワーク内では，常に成り立つ知識を“その知識が常に成り立つと信じている”と，とりあえず常識的には成り立つ知識，すなわち常に成り立つとは限らない知識を“その知識の否定はまだ信じない”と表現することにより，二つの型の知識を区別して扱う．また，意味ネットワークの知識の同化および知識の調節を行なう知識管理アルゴリズムを示す．新しい知識を意味ネットワークに同化するには意味ネットワークの非冗長性および無矛盾性を維持するために知識の追加と知識の削除を行なう．知識の調節は下位階層に共通する知識を上位階層に伝搬する．また，これらの管理では意味ネットワーク内の推論履歴情報である根拠[4]を用いる．

以下では，2章で信念に基づいた知識の意味ネットワーク表現を定義する．また，3章で意味ネットワーク管理システムの構成について述べ，4章で意味ネットワークの性質継承アルゴリズムについて述べ，5章で意味ネットワークの管理法について述べる．さらに，6章で本稿で述べる非単調多重継承方式と他方式との比較を行なう．

なお，本文中のアルゴリズム等の表記は IF Prolog に準拠している．

2. 信念に基づいた知識の意味ネットワーク表現

ここでは，is-a リンクと P リンクを定義し，これらを用いて知識を表現する．

定義 2.1 is-a リンク (is-a 関係の知識)

is-a 関係をもつ知識を以下に示す記法の is-a リンクで表現する．

$$is\text{-}a(concept, upper\_concept).$$

ここに，

$concept$: 概念，

$upper\_concept$: 概念の上位概念．□

定義 2.2 P リンク (概念の性質に関する知識)

is-a 関係以外の知識で概念の性質に関する知識を以下に示す記法の P リンクで表現する．

$$link(i, af/neg, attrib, concept, value, J, belief).$$

ここに，

i : P リンク番号 $(i = 1, \cdots, m)$

af/neg : af(肯定) または neg(否定)

af : $attrib(concept, value)$

neg : $\neg attrib(concept, value)$

$attrib$: 属性

$concept$: 概念

$value$: 属性値

J : $belief$(信念) を支持する P リンク番号のリスト (リスト J をリンク i の根拠[4] と呼ぶ)

$belief$: in または out

in : af/neg が真であると信ずる，

out : af/neg が真であると信じない

なお，$J = [\emptyset]$(空リスト) である P リンクを前提 P リンク，および $J = [\alpha]$ $(\alpha \neq \emptyset)$ の P リンクを根拠 α をもつ仮説 P リンクと呼び，$belief = in$ の P リンクを in P リンク，および $belief = out$ の P リンクを out P リンクと呼ぶ．

さらに，$belief = in/out$ の前提 P リンクを前提 in/out P リンク，および $belief = in/out$ の仮説 P リンクを仮説 in/out P リンクと呼ぶ．

また，P リンクを単に $link(i)$ と表すこともある．□

定義 2.3 意味ネットワーク

意味ネットワーク S は is-a リンクと P リンクの集合である．ただし，S 内では is-a リンクは推移律が成り立ち，非循環 (acyclic) とする．すなわち，

$$is\text{-}a(X, Y) \wedge is\text{-}a(Y, Z) \supset is\text{-}a(X, Z). \quad (1)$$

ただし，$X \neq Z$ である．□

定義 2.4 性質の推論

P リンク内の概念の性質判定には以下の推論方法をとる．

$$link(i, \mathbf{af}, attrib, concept, value, J, \mathbf{in}) \rightarrow attrib(concept, value). \quad (2)$$

$$link(i, \mathbf{neg}, attrib, concept, value, J, \mathbf{in}) \rightarrow \neg attrib(concept, value). \quad (3)$$

$$link(i, \mathbf{neg}, attrib, concept, value, J, \mathbf{out}) \rightarrow attrib(concept, value). \quad (4)$$

$$link(i, \mathbf{af}, attrib, concept, value, J, \mathbf{out}) \rightarrow \neg attrib(concept, value). \quad (5)$$

(2) ～ (5) の推論により得られる → の右辺を概念 ($concept$) の性質と呼ぶ．ここに，(2)，(3) は常に成り立つ概念の性質であると判定し，(4)，(5) はとりあえず常識的に成り立つ概念の性質であると判定するときに用いる．

また，性質 $attrib(concept, value)$ と $\neg attrib(concept, value)$ が同時に推論されるときこれらを互いに排反するという．□

次に，意味ネットワークの例を示す．

例 2.1

- 鳥 (bird) は飛ぶこと (fly) ができないと信じていない (= 鳥は普通, 飛ぶ).
- ペンギン (penguin) は飛ぶことができないと信ずる (= ペンギンは飛べない).
- ペンギンは鳥である.

上の記述は以下のようにis-aリンクとPリンクからなる意味ネットワーク S で表現できる.

$S = \{link(1, neg, can, bird, fly, [], out),$
$link(2, neg, can, penguin, fly, [], in),$
$is\text{-}a(penguin, bird)\}$. □

次に, 意味ネットワーク S とPリンクの冗長と矛盾を定義する.

定義 2.5 冗長Pリンク

$link(i, af/neg_{(i)}, attrib, concept, value, J_{(i)}, belief_{(i)}) \in S$ および $link(j, af/neg_{(j)}, attrib, concept, value, J_{(j)}, belief_{(j)}) \in S$ に対し,

$$((af/neg_{(i)} = af/neg_{(j)}) \wedge (belief_{(i)} = belief_{(j)}))$$
$$\vee((af/neg_{(i)} \neq af/neg_{(j)})$$
$$\wedge(belief_{(i)} = in) \wedge (belief_{(j)} = out))$$

が成り立つとき, 意味ネットワーク S は**冗長であるといい**, $link(j)$ を $link(i)$ の**冗長Pリンク**と呼ぶ. □

定義 2.6 矛盾Pリンク

$link(i, af/neg_{(i)}, attrib, concept, value, J_{(i)}, belief_{(i)}) \in S$ および $link(j, af/neg_{(j)}, attrib, concept, value, J_{(j)}, belief_{(j)}) \in S$ に対し,

$$((af/neg_{(i)} = af/neg_{(j)}) \wedge (belief_{(i)} \neq belief_{(j)}))$$
$$\vee((af/neg_{(i)} \neq af/neg_{(j)})$$
$$\wedge(belief_{(i)} = in) \wedge (belief_{(j)} = in))$$

が成り立つとき, 意味ネットワーク S は**矛盾する**といい, $link(j)/link(i)$ を $link(i)/link(j)$ の**矛盾Pリンク**と呼ぶ. □

なお, af(肯定) のoutPリンクと neg(否定) のoutPリンクにおいては, "冗長と矛盾については未定義" とする. すなわち, 肯定も否定も信じないということを矛盾していると判断しない. これは, 人間が通常行なっている曖昧さを残す判定法と一致しているといえる.

3. 意味ネットワーク管理システム

意味ネットワーク管理システム (semantic network management system) は2章で定義した信念に基づく意味ネットワーク内の非単調多重継承性質の決定と知識の管理を行なう.

図1に示す意味ネットワーク管理システムは知識ベースと性質決定システム, および, 知識管理システムから構成される.

知識ベース K は意味ネットワーク S, 冗長知識ベース R, 矛盾知識ベース C から構成される. 意味ネットワーク S はis-a関係が非循環 (acyclic) となるis-aリンクと, ある知識が同

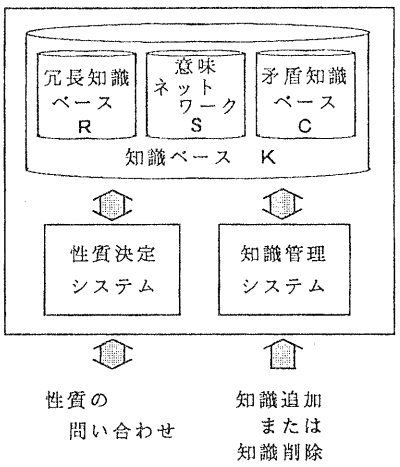

図1：意味ネットワーク管理システム

化される以前は非冗長性かつ無矛盾性が保証されているPリンクの集合である. R, C はそれぞれある知識を同化する際に S から派生した冗長Pリンクと矛盾Pリンクの集合である.

性質決定システムは概念の性質に関する外部からの問い合わせに応じて K を参照し, 非単調多重継承を行ない定義2.4に従い概念の性質を決定する.

また, 知識管理システムでは, 新しいis-aリンクおよびPリンクを S に追加するときに派生する冗長Pリンクと矛盾Pリンクを検出して, これを S から削除する. 冗長Pリンクは R に, 矛盾Pリンクは C に入れ保持する.

4. 性質決定システム

性質決定システムは継承性質候補選出アルゴリズムと継承性質決定アルゴリズムからなる.

4.1 継承性質候補選出アルゴリズム

継承性質候補選出アルゴリズムは上位概念から継承する性質の候補を選出する. そこでまず, S でのPリンクの無矛盾性を表すために, オペレータ M^*_S を次のように定義する.

定義 4.1 M^*_S (無矛盾オペレータ)

$\mathsf{M}^*_S\, link(\_, af/neg, attrib, concept, value, \_, belief)$.
は $link(\_, af/neg, attrib, concept, value, \_, belief)$ の矛盾リンクが S に存在しないことを表す. □

次に, S でのPリンクの継承性質候補の選出規則を定義する.

定義 4.2

S のPリンクにおける選出規則を (6)~(9) とする.

$pi([is\text{-}a(X, Y),$
$link(I, af, Attrib, Y, Z, J, in),$
$\mathsf{M}^*_S\, link(\_, neg, Attrib, X, Z, \_, out)] \Rightarrow$

$[link(N, neg, Attrib, X, Z, [I], out)]).$ (6)

$pi([is\text{-}a(X, Y),$
$link(I, neg, Attrib, Y, Z, J, in),$
$\mathsf{M}^*_S\ link(\_, af, Attrib, X, Z, \_, out)] \Rightarrow$
$[link(N, af, Attrib, X, Z, [I], out)]).$ (7)

$pi([is\text{-}a(X, Y),$
$link(I, af, Attrib, Y, Z, J, out),$
$\mathsf{M}^*_S\ link(\_, af, Attrib, X, Z, \_, out)] \Rightarrow$
$[link(N, af, Attrib, X, Z, [I], out)]).$ (8)

$pi([is\text{-}a(X, Y),$
$link(I, neg, Attrib, Y, Z, J, out),$
$\mathsf{M}^*_S\ link(\_, neg, Attrib, X, Z, \_, out)] \Rightarrow$
$[link(N, neg, Attrib, X, Z, [I], out)]).$ (9)

上の規則 (6)〜(9) は S で $\Rightarrow$ の左側が成り立てば右側を生成し選出する．これらは，上位概念 Y の性質 Z が下位概念 X でも常識的に成り立つと考えることが無矛盾ならば，X の性質として選出することを表す．

なお，選出された P リンク $link(N)$ はリスト $[I]$ を根拠とする仮説 out P リンクとなる．□

以下に，意味ネットワーク S 上の概念 $Concept$ に対し継承性質候補の選出を行なう手続き prop_inher を示す．

procedure prop_inher(input: *Concept*; output: PI):
{is-a 階層の上位概念から *Concept* に継承され，選出される P リンクと推移律により成り立つ *Concept* の is-a リンクのリスト PI を返す}
begin
  upper_concepts(*Concept*,[ ];Uppers);
  Uppers′ ← Uppers;
  while Uppers′ ≠ [ ] **do begin**
    Uppers′ = [Up|Tail];
    List ← [*is-a*(*Concept*,Up)|List];
    Uppers′ ← Tail
  end;
  Uppers ← Uppers ∪ [*Concept*];
  List ← [ ];
  while Uppers ≠ [ ] **do begin**
    Uppers = [*Concept*′|Tail];
    apply_pi_rule(*Concept*′,List;Pred);
    List ← Pred;
    Uppers ← Tail
  end;
  PI ← Pred
end.

procedure upper_concepts(input: *Concept*,List;
output: Uppers):
{*Concept* の is-a 階層の上位概念のリスト Uppers を返す}
begin
  Uppers ← List;
  while $\exists X((is\text{-}a(Concept, X) \in S) \wedge (X \notin \text{Uppers}))$ **do begin**
    upper_concepts(*X*,[*X*|Uppers]; Uppers′)
    Uppers ← Uppers′
  end
end.

procedure apply_pi_rule(input: *Concept*,List;
output: Pred):
{Uppers の要素に対し最上位概念から順に選出規則を適用し，生成される P リンクをリスト Pred を返す}
begin
  while ∃ 未適用の適用可能な選出規則
    $pi([is\text{-}a(Concept, \_)|\_] \Rightarrow$[P]) **do begin**
    List ← [P|List]
  end;
  Pred ← List
end.

例 4.1

次のような常識的な性質継承が成り立たない意味ネットワークでの継承性質候補の選出を考える．

- 象は普通，灰色である．
- ロイヤル象は象である．
- ロイヤル象は灰色ではない．
- *Clyde* はロイヤル象である．

上の記述の意味ネットワーク S は以下のようになる．

$S = \{link(1, neg, color, elephant, gray, [\,], out),$
$is\text{-}a(royal\_elephant, elephant),$
$link(2, neg, color, royal\_elephant, gray, [\,], in),$
$is\text{-}a(clyde, royal\_elephant)\}.$

手続き prop_inher により *clyde* が継承する知識 PI は次のように求められる．
$? - prop\_inher(clyde, PI).$

$PI = [is\text{-}a(clyde, elephant),$
$link(3, af, color, clyde, gray, [2], out)]$

ここでは以下の処理が行なわれた．

まず，is-a リンクの推移律から $is\text{-}a(clyde, elephant)$ が成り立つ．次に，$link(1)$ に適用できる選出規則はない．また，$link(2)$ に選出規則 (7) が適用でき $link(3)$ が選出される．□

4.2 継承性質決定アルゴリズム

性質決定システムは，選出された継承すべき多重の性質の候補の中から定義 4.3 に示す P リンクの優先順位に従って継承する性質を決定する．

定義 4.3 P リンクの決定優先順位

以下の順に優先順位が高いとし，この優先順位により性質を決定する．

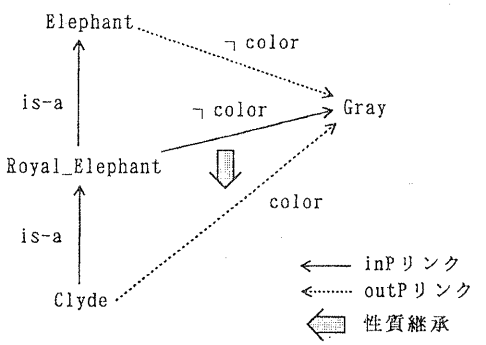

図 2：継承性質候補の選出

1. in P リンク (前提 P リンクと仮説 P リンク)
2. 前提 out P リンク
3. in P リンクから継承された仮説 out P リンク
4. out P リンクから継承された仮説 out P リンク □

なお，ここで 3 を 4 より優先させることは信じている上位性質から継承された P リンクを優先させることを意味する．また，優先順位が同じ P リンク同士からは性質を判定しない．

以下に，継承性質候補から性質を決定する手続き prop_decision を示す．

```
procedure prop_decision(input: Concept, PI;
                        output: Properties):
{継承性質候補のリスト PI を入力し，概念 Concept の性質
Properties を返す}
begin
  L ← [概念が Concept である S 内のすべての知識のリスト];
  PI' ← [概念が Concept である PI 内の
                          すべての知識のリスト];
  L ← L ∪ PI';
  Properties ← [L 内のすべての is-a リンクのリスト];
  L ← L − Properties;
  while L ≠ [] do begin
    L = [link(_,_,Attrib,Concept,Value,_,_)|Tail];
    if (link(_,af,Attrib,Concept,Value,_,in) ∈ L)
        ∧(link(_,neg,Attrib,Concept,Value,_,in) ∈ L)
                                        then
      Properties ← Properties;
    else if link(_,af,Attrib,Concept,Value,_,in) ∈ L
                                        then
      Properties ← [Attrib(Concept,Value)|Properties]
    else if link(_,neg,Attrib,Concept,Value,_,in) ∈ L
                                        then
      Properties ← [¬Attrib(Concept,Value)|
                                    Properties]
    else if (link(H,af,Attrib,Concept,Value,Jh,out)
    ∈ L) ∧ (link(I,neg,Attrib,Concept,Value,Ji,out)
                                        ∈ L) then
      if (Jh = []) ∧ (Ji ≠ []) then
        Properties ← [¬Attrib(Concept,Value)|
                                    Properties]
      else if (Jh ≠ []) ∧ (Ji = []) then
        Properties ← [Attrib(Concept,Value)|
                                    Properties]
      else begin
        compare_strength(link(H),link(I),
                                Properties; List);
        Properties ← List
      end
    else if link(_,af,Attrib,Concept,Value,_,out)
                                        ∈ L then
      Properties ← [¬Attrib(Concept,Value)|
                                    Properties]
    else if link(_,neg,Attrib,Concept,
                        Value,_,out) ∈ L then
      Properties ← [Attrib(Concept,Value)|
                                    Properties]
    del(Tail,Attrib,Concept,Value;L');
    L ← L'
  end
end.

procedure compare_strength(input: link(H),
                    link(I),List; output: Properties):
{推論される性質が排反する out P リンク link(H)，link(I)
の根拠をさかのぼり，in P リンクから継承された方の性質を
優先する}
begin
  trace_justi(link(H);Lh);
  trace_justi(link(I);Li);
  if ((in P リンク)∈ Lh) ∧((in P リンク)∉ Li) then
    Properties ← [¬Attrib(Concept,Value)|List]
  else if ((inP リンク)∉ Lh) ∧((inP リンク)∈ Li) then
      Properties ← [Attrib(Concept,Value)|List]
    else Properties ← List
end.

procedure trace_justi(input: link(I,_,_,_,_,J,_);
                                output: L):
{link(I) の根拠の P リンク番号のリストを推移的に求める}
begin
  L' ← [];
  J' ← J;
  while J' ≠ [] do begin
    J' = [J1|Tail];
    L' ← [link(J1)|L'];
    trace_justi(link(J1);L'');
    L ← L' ∪ L'';
    J' ← Tail
  end
end.
```

procedure del(input: L, $Attrib, Concept, Value$; output: L'):
{リストLから属性，概念，属性値が $Attrib$， $Concept$， $Value$ であるPリンクを削除する}
begin
List ← [$link(\_,\_,Attrib,Concept,Value,\_,\_) \in$ LであるすべてのPリンクのリスト];
L' ← L − List
end.

例 4.2

Sandewallの Type−1c 問題と呼ばれる次のような非単調多重継承の例を考える．

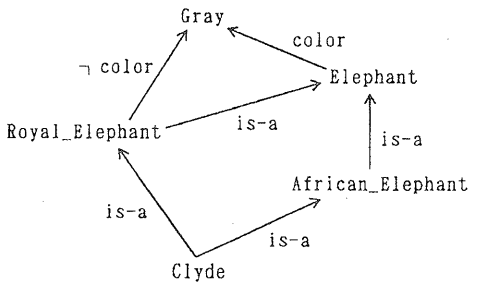

図 3：Type−1c 問題

性質 $color(elephant, gray)$ が常に成り立つとは限らない性質であるとすると，上の例の意味ネットワーク S は以下のように表現できる．

$S = \{is\text{-}a(clyde, royal\_elephant),$
$is\text{-}a(clyde, african\_elephant),$
$is\text{-}a(royal\_elephant, elephant),$
$is\text{-}a(african\_elephant, elephant),$
$link(1, neg, elephant, color, gray, [\,], out),$
$link(2, neg, royal\_elephant, color, gray, [\,], in)\}$

性質継承手続き prop_inher，および性質決定手続き prop_decision を実行することにより， $clyde$ の性質 Properties は以下のよう求められる．

$?- prop\_inher(clyde; PI),$
$prop\_decision(clyde, PI; Properties).$

$PI = [is\text{-}a(clyde, elephant),$
$link(3, neg, african\_elephant, color, gray, [1], out),$
$link(4, neg, clyde, color, gray, [3], out),$
$link(5, af, clyde, color, gray, [2], out)]$
$Properties = [is\text{-}a(clyde, royal\_elephant),$
$is\text{-}a(clyde, african\_elephant),$
$is\text{-}a(clyde, elephant),$
$\neg color(clyde, gray)]$

ここで，手続き prop_inher により得られた $link(4)$，$link(5)$ は推論される性質が排反する．さらに，いずれも仮説outPリンクである．そこで，二つのPリンクの根拠をたどると，$link(4)$ の根拠 $link(1)$ はoutPリンク，$link(5)$ の根拠 $link(2)$ はinPリンクである．従って，定義4.3の優先順位に従い $link(4)$ から $\neg color(clyde, gray)$ と決定される．すなわち，$clyde$ の性質として“$gray$ でない”のみを得る．□

5. 知識管理システム

ここでは，信念を含む意味ネットワーク S の知識同化，知識削除，および，知識調節の各機構について述べる．

5.1 知識管理

知識管理システムは以下に示す知識同化手続き k_assimilation，知識削除手続き k_deletion により，知識ベース K に知識同化，知識削除を行ない S の非冗長性，無矛盾性を維持する．また，手続き k_accommodation により知識調節を行なう．すなわち，is-a 関係の下位概念に共通する性質でかつ無矛盾なものを仮説Pリンクとして生成し，上位概念に伝搬する．

procedure k_assimilation(input: $link(i)$):
begin
assimilation($link(i)$);
redundant_manage
end.

procedure assimilation(input: Link):
begin
if Link = $is\text{-}a(x, y)$ **then**
if $is\text{-}a(x, y) \in S$ **then**
$R \leftarrow R \cup \{is\text{-}a(x, y)\}$
else begin
$S \leftarrow S \cup \{is\text{-}a(x, y)\}$;
if S のis-a関係が循環 (cyclic) **then begin**
$S \leftarrow S - \{is\text{-}a(x, y)\}$;
$C \leftarrow C \cup \{is\text{-}a(x, y)\}$
end
end
else begin
$link(I, Af/Neg, Attrib, Concept, Value, \_, Belief)$ = Link;
$S \leftarrow S \cup \{link(I)\}$;
if $link(I)$ の冗長Pリンク $link(M) \in S$ **then begin**
$S \leftarrow S - \{link(M)\}$;
$R \leftarrow R \cup \{link(M)\}$
end
else begin
while $\exists H((link(H) \in S) \vee (link(H) \in R)) \wedge$
($link(H)$ は $link(I)$ の矛盾Pリンク) **do begin**
k_deletion($link(H)$);
$C \leftarrow C \cup \{link(H)\}$
end;
accommodation($link(I)$)
end;

```
    Uppers ← [is-a(Concept,Up) ∈ S であるすべての Up のリスト];
    while Uppers ≠ [] do begin
      Uppers = [Up|Tail];
      if ∃M(((link(M, _, Attrib,Up, Value, J, _) ∈ S)
        ∨(link(M, _, Attrib,Up, Value, J, _) ∈ R)) ∧ (J ≠ [])
         ∧(link(M) は link(I) の冗長 P リンクでない)) then
        k_deletion(link(M));
      Uppers ← Tail
    end
  end
end.

procedure k_deletion(link(I)):
begin
  delete(link(I));
  redundant_manage
end.

procedure delete(input: Link):
begin
  if Link ∈ C then
    C ← C − {Link}
  else begin
    if Link∈ S then
      S ← S− {Link}
    else
      R ← R− {Link};
    if Link = is-a(x, y) then begin
      L ← [S の非環状性を守る C 内のすべての is-a 関係のリスト];
      while L ≠ [] do begin
        L = [is-a(a, b)|Tail];
        k_assimilation(is-a(a, b));
        L ← Tail
      end
    end else begin
      link(I) ← Link;
      while ∃H (link(I) の矛盾 P リンク link(H) ∈ C)
                                          do begin
        C ← C − {link(H)};
        k_assimilation(link(H))
      end;
      while ∃M (((link(M, _, _, _, _, J, _) ∈ S)
            ∨(link(M, _, _, _, _, J, _) ∈ R)
             ∨(link(M, _, _, _, _, J, _) ∈ C)) ∧ (I ∈ J)) do
        k_deletion(link(M))
    end
  end
end.

procedure accommodation(input:link(I, Af/Neg,
                Attrib, Concept, Value, _, Belief)) :
begin
  Uppers ← [is-a(Concept,Up) ∈ S であるすべての Up のリスト];
  while Uppers ≠ [] do begin
    Uppers =[Up|Tail];
    if M*S link(_, Af/Neg, Attrib,Up, Value, _, Belief)
                                          then begin
      Lowers ← [is-a(Low,Up)∈ S であるすべての Low のリスト];
      Lowers′ ← Lowers;
      Lowers′ = [Low′|Tail′];
      while (Lowers′ ≠ [])∧
       (link(_, Af/Neg, Attrib,Low′, value, _, Belief) ∈ S)
                                          then begin
        Lowers′ ← Tail′;
        Lowers′ = [Low′|Tail′]
      end;
      if Lowers′ = [] then
        assimilation(link(_, Af/Neg, Attrib,Up, Value,
                                          Lowers, Belief))
    end;
    Uppers ← Tail
  end
end.

procedure redundant_manage:
begin
  while ∃I ∃H ((link(I) ∈ R)
       ∧(link(I) の冗長 P リンク link(H) ∉ S)) do begin
    R ← R − {link(I)};
    S ← S ∪ {link(H)}
  end;
  while ∃I ∃H ((link(I) ∈ S)
       ∧(link(I) の冗長 P リンク link(H) ∈ S)) do begin
    S ← S − {link(I)};
    R ← R ∪ {link(H)}
  end
end.
```

5.2 実行例

信念を含む意味ネットワークの知識管理システムによる実行例を以下に示す.

例 5.1

知識ベース K が以下の S, R, C であるとする.

$K = (S, R, C)$

$S = \{is\text{-}a(man, mammal),$
$\quad is\text{-}a(dog, mammal),$
$\quad is\text{-}a(horse, mammal),$
$\quad link(1, neg, can, mammal, swim, [], out),$
$\quad link(2, af, can, man, swim, [], in),$
$\quad link(3, af, can, dog, swim, [], in),$
$\quad link(4, af, can, horse, swim, [], out)\}$

$R = \emptyset$

$C = \emptyset$

S に次の知識 $link(5)$ を同化する.

$link(5, af, can, horse, swim, [\,], in)$.

5.1節で示した知識同化手続き k_assimilation を実行する.

$? - k\_assimilation(link(5, af, can, horse, swim, [\,], in))$.

$S = \{is\text{-}a(man, mammal),$
$\quad is\text{-}a(dog, mammal),$
$\quad is\text{-}a(horse, mammal),$
$\quad link(2, af, can, man, swim, [\,], in),$
$\quad link(3, af, can, dog, swim, [\,], in),$
$\quad link(5, af, can, horse, swim, [\,], in),$
$\quad link(6, af, can, mammal, swim, [2, 3, 5], in)\}$
$R = \{link(1, neg, can, mammal, swim, [\,], out)\}$
$C = \{link(4, af, can, horse, swim, [\,], out)\}$

ここでは，次の処理が行なわれた.

$link(5)$ を S に追加すると，$link(4)$ は矛盾Pリンクとなり S から削除され C に追加される．次に，知識調節手続き accommodation により $horse$ の上位概念 $mammal$ に対し下位概念のすべての知識を調べる．$mammal$ の下位概念 man, dog, $horse$ はいずれも“泳ぐことができると信ずる”という知識を持つことから，Pリンク $link(6)$ を生成し $mammal$ に伝搬する．ここで，$link(6)$ を S に追加すると，$link(1)$ は冗長Pリンクとなるので S から削除され R へ追加される.

以上で，知識同化処理が完了する．□

例 5.2

例 5.1の知識ベースに次の知識 $link(7)$ を同化する.

$link(7, neg, can, man, swim, [\,], out)$.

知識同化手続き k_assimilation を実行する.

$? - k\_assimilation(link(7, neg, can, man, swim, [\,], out))$.

$S = \{is\text{-}a(man, mammal),$
$\quad is\text{-}a(dog, mammal),$
$\quad is\text{-}a(horse, mammal),$
$\quad link(1, neg, can, mammal, swim, [\,], out),$
$\quad link(3, af, can, dog, swim, [\,], in),$
$\quad link(5, af, can, horse, swim, [\,], in),$
$\quad link(7, neg, can, man, swim, [\,], out)\}$
$R = \emptyset$
$C = \{link(2, af, can, man, swim, [\,], in),$
$\quad link(4, af, can, horse, swim, [\,], out)\}$

ここでは，次の処理が行なわれた.

$link(7)$ を S に追加すると，$link(2)$ は矛盾Pリンクとなり S から削除され C に追加される．このとき，$link(2)$ を根拠に持つ仮説Pリンク $link(6)$ は根拠を失うので S から削除される．さらに，$link(1)$ は冗長Pリンクではなくなるので R から削除され S に追加される．□

ここで，例 5.2は R の知識が S に復活し追加される例である[†].

[†]本稿で提案したアルゴリズムでは C の知識は S に追加されない.

6. 非単調多重継承の他方式との比較

デフォルト論理[5]を用いた非単調性質継承の形式化として文献6), 7) がある．ここでは，4章で述べた性質決定システムの非単調多重継承処理とこれらの方式との比較を述べる.

先に述べた例 4.2の場合，文献6), 7) の方式では $African\_elephant$ の経路から継承される性質 $color(clyde, gray)$ と $Royal\_elephant$ の経路から継承される性質 $\neg color(clyde, gray)$ を矛盾すると判定する．この後，文献6) では二つの経路の性質継承を排他的に行ない二種類の解を出力し，文献7) では二つの経路の性質継承を並列に行ない $color(clyde, gray)$ と $\neg color(clyde, gray)$ を同時に出力する.

一方，本方式では得られる性質は性質継承経路の順序に依存しない．また，根拠となるPリンクをたどり，その信念を比較することによりinPリンクから継承された性質“$gray$ でない”のみを出力する.

なお，本方式は文献7) の方式と同様に，性質継承規則が意味ネットワークとは独立に定めているため管理が容易で簡潔に記述することができる.

7. おわりに

信念を用いて常に成り立つ知識と常に成り立つとは限らない知識を意味ネットワーク表現する方法を示した．また，性質継承された知識を常に成り立つとは限らない知識であるとし，多重継承の場合には根拠をたどることにより常に成り立つ知識から継承された知識を優先するという非単調多重継承方法を示した．さらに，この意味ネットワークの知識同化，知識削除，および知識調節の方法を示した.

参考文献

1) 国藤，北上，宮地，古川：論理プログラミング言語 Prolog による知識ベース管理システム，情報処理，Vol.26, No.11 (1985).

2) 松田，石塚：仮説推論システムの拡張知識表現と概念学習機構，人工知能学会誌，Vol.3, No.1 (1988).

3) 佐藤，伊藤：例外知識を含む意味ネットワークの仮説に基づく管理法，第43回情報処理学会全国大会 (1991).

4) Doyle,J.: A Truth Maintenance System, Artificial Intelligence, 12, (1979).

5) Reiter,R.: A Logic for Default Reasoning, Artificial Intelligence, 13, (1980).

6) Etherington,D.W.: Formalizing Non-monotonic Reasoning Systems, Artificial Intelligence 31, (1987).

7) 坂間，奥村：非単調並列継承ネットワーク，Proc. of LPC (1988).