複数概念の選言表現の逐次的学習のための複合多重集約アルゴリズム†

田代 勤†† 薦田憲久††

ルール型制御システム，エキスパートシステム等のルール処理技術を用いた実用システムが多数開発され効果を上げている．しかし，大規模なシステムでは，矛盾のない質の良いルールの作成に多大な工数を要し，ルール作成工数の削減が重要課題となっている．これに対処する一手段として，機械学習による自動的ルール獲得の手法が注目を集めている．この手法によれば，幾つかの典型的対象状態と結論の対からなる実例を与えるだけで多くの場合に適用できる一般的なルールを自動的に矛盾なく導けるため，上記問題の解決に役立つことが期待される．しかしながら，より広範な現実の問題への適用を考える時，(1)複数の結論を並列に学習できる，(2)結論ごとに独立したルールを学習できる，(3)条件の選言表現を学習できる，(4)逐次的に学習を行えることが要求される．本論文では，複数概念の独立した選言表現の逐次的な学習を可能とする新しいアルゴリズムを提案する．本アルゴリズムでは，概念ごとの複数のバージョン空間および各バージョン空間内の選言表現のための複数のサブ空間を同時に矛盾なく管理し狭めて行くことでもれなく求める概念記述を探索することが可能であり，広範な対象の現実問題を扱うシステムを構築できることが期待される．

1. はじめに

ソフトウェアの生産性，保守性向上の観点から，ルール型制御システム[1),2)]等の知識処理技術を用いたプログラミングシステムが開発され効果を上げている．これは，知識処理アーキテクチュアを取り入れることで，従来の手続型言語における処理手順の記述を不要とし，ソフト開発，変更の容易化を図ることを主眼としている．しかし，ルールの作成，十分性，無矛盾性の検証はすべて人手によっており，大規模，複雑な対象ではこれらに多くの工数を要している．このためルールを効率的に作成できる方法が求められている．

一方，各種エキスパートシステムの開発に際して，知識獲得のボトルネック[3)]が問題となってきている．知識工学の概念が定着し種々の実用エキスパートシステムが開発されるにつれ，有効な多くの知識を獲得するには多大な労力が必要なことが認識されつつあり，知識獲得の支援に対する要望が強まっている．

以上に対処する一手段として，機械学習による自動的ルール獲得の手法[4)]が注目を集めるようになってきている．この手法によれば，幾つかの典型的対象状態と結論の対からなる実例を与えるだけで多くの場合に適用できる一般的なルールを自動的に矛盾なく導けるため，上記問題の解決に役立つことが期待される．例えば，病気の診断ルールを医者の診断例から導く，文字認識ルールを人間の認識結果から導く等の形での利用が考えられ，限られた範囲ではあるが実際の応用[5),6)]も試みられつつある．

しかしながら，より広範な現実の問題への適用を考える時，以下を満足できる手法が要求されてくる．

(1) 複数の結論を並列に学習できる．

病気の診断では，多くの病名の内から患者の症状に対応するものを結論として選ぶ．このような現実問題では対象に応じた複数の結論が取り扱われる．様々な結論を持つ実例から，それぞれの結論ごとのルールを同時に導ける方式が必要となる．

(2) 結論ごとに独立したルールを学習できる．

文字認識，制御等では対象状態ごとに一つの結論を導くことが必要となる．学習したルールの条件の表現に重複があると一つの対象状態に対して複数の結論が下されるため，そのままでは使用できない．競合のない独立したルールを学習することが要求される．

(3) 条件の選言表現を学習できる．

通常，ルールの条件は「色が赤かつ多面体かつ重い」のように属性の連言表現（"and" でつながれた条件）で記述されるが，一つの連言表現だけでは対象の状態を規定する範囲が限られる．実際の問題では，ある対象状態と別の対象状態に対して同じ結論となる場合のように分離した複数のルールが必要となる．『「色が赤かつ多面体かつ重い」あるいは「色が黄かつ多面体かつ軽い」』等の連言の選言（"or" でつながれた条件）からなる条件表現も扱える必要がある．

† Combined Multiple Convergence Algorithm for Incremental Disjunctive Multiple Concept Learning by TSUTOMU TASHIRO and NORIHISA KOMODA (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

（4） 逐次的に学習を行える．

必要十分な実例を最初から与えることは，現実には困難である．幾つかの実例をまず与え，その範囲で学習されたルールを試用してみて誤った結論が導かれれば，正しい結論とした実例を与えるといったサイクルを繰り返すのが一般的である．学習状況に応じて実例を増やしていくというように，逐次的に学習を進めることのできる方式が必要である．

しかし，従来のアルゴリズムは単一概念の連言表現を学習するもの[4),7)~9)]がほとんどである（機械学習では，例えばアーチを判定するルールの学習をアーチの概念の学習といい，ルールの結論を概念，条件をその一般表現あるいは単に表現と呼ぶことが多い．以後これに従う）．これに対し，Michalski らは複数概念の独立な選言表現を学習する方式[10)]を与えているが，すべての実例を同時に与えねばならず逐次的な利用はできない．また，並列でなく各概念ごとに順に学習を進める方式のため，先の概念の表現ほど一般的で後のものほど特殊な表現となってしまいバランスのとれた学習を行えない．一方，Murray は逐次的に選言表現を学習する方式[11)]を与えているが，単一概念の場合しか扱っていない．

前記要求をすべて満足する方法として，最近 Gross の提案[12)]がなされた．ここでは，Michalski の概念の記述法 APC[10)] の下で，正の実例（その概念であることを示す実例）から学習する場合を示している．

本論文では，より広い範囲の概念および実例の記述法を前提にして，正および負の実例（その概念でないことを示す実例）から学習するという一般的枠組みの下で，前記要求をすべて満足する新たな学習アルゴリズムを提案する．本アルゴリズムは，求める学習結果をもれなく探索することが可能であり広範な応用が期待できる．

なお，問題によっては，可能性のある幾つかの結論をルールで下し，後の判断を別の基準に任せる等，独立したルールよりも条件に重複のあるルールを求めたい場合も考えられる．本論文では，独立でないルールを学習する場合についても考察を加える．

2. 問題の記述

初めに，本論文で扱う問題を明確に記述しておく．複数概念の独立した選言表現の逐次的な学習の問題は，以下のように規定される．

[Given]

①概念集合 $q_i \in Q$, $i=1,\cdots,n$：学習すべき概念（ルールの結論に相当）の集合．

②概念記述言語 CL：概念の一般表現（ルールの条件に相当）を記述する言語．その文を概念記述と呼ぶ．

③実例記述言語 EL：対象の状態（事実に相当）を記述する言語．その文を実例記述と呼ぶ．

④照合関係 $e \rightarrow c$：EL の実例記述（事実）e が CL の概念記述（条件）c を満足するか否かを判定する関係．

$e \rightarrow c$ の時，c は e を包含するという．c が e を包含しないことを $\sim(e \rightarrow c)$ と書く．

⑤一般化関係 $c < c'$：CL の概念記述の一般化（あるいは逆に特殊化）を与える半順序関係．

c' が c より一般的 $c<c'$ とは，c' に包含される実例記述の集合 $E' \equiv \{e \mid e \rightarrow c'\}$ が c に包含される実例記述の集合 $E \equiv \{e \mid e \rightarrow c\}$ を含む $E' \supseteq E$ 場合をいう．$c<c'$ の時 c' は c より大きい，c は c' より小さいという．

⑥概念ごとの正の実例の集合 $P_1, P_2, \cdots, P_n$：EL の実例記述 e が概念 q_i の具体例であることを示す対 $+[e, q_i]$（正の実例と呼ぶ）からなる集合 $+[e, q_i] \in P_i$.

⑦概念ごとの負の実例の集合 $N_1, N_2, \cdots, N_n$：EL の実例記述 e が概念 q_i の具体例でないことを示す対 $-[e, q_i]$（負の実例と呼ぶ）からなる集合 $-[e, q_i] \in N_i$.

以上において，概念記述言語 CL は実例記述言語 EL のすべての文を区別するのに十分な記述性を持つとする．すなわち，少なくとも任意の実例記述に対し，これのみを包含する要素概念記述が存在するとする．なお，CL＝EL とする単一表現トリックはこの条件を満たす．

[Find]

$P_1, P_2, \cdots, P_n$ および $N_1, N_2, \cdots, N_n$ の要素を逐次ランダムに入力し，以下を満足する概念記述 c の集合 $C_1, C_2, \cdots, C_n$ を求める．

（i） 各 C_i は，概念 q_i のすべての正の実例 $+[e, q_i]$ の実例記述 e をカバーする（完全性）．すなわち，

$$^{\forall}i, {}^{\forall}e : e \in P_i ; {}^{\exists}c : [e \rightarrow c,\ c \in C_i].$$

（ii） 各 C_i は，概念 q_i のどの負の実例 $-[e, q_i]$ の実例記述 e をもカバーしない（無矛盾性）．すなわち，

$$^{\forall}i, {}^{\forall}e : e \in N_i ; {}^{\forall}c : [\sim(e \rightarrow c),\ c \in C_i].$$

(ⅲ) 各 C_i は共通部分を持たない（独立性）．すなわち，$j \neq k$ なる C_j, C_k の任意の要素 c_j, c_k の組合せが共通部分を持たない．

概念記述 c, c' が共通部分を持たない $c \cap c' = \phi$ とは，$E \equiv \{e \mid e \rightarrow c\}$, $E' \equiv \{e \mid e \rightarrow c'\}$ が $E \cap E' = \phi$ となる場合をいう．逆に，$E \cap E' \neq \phi$ の時 c と c' は共通部分を持つ $c \cap c' \neq \phi$ という．

上記は，Subramanian らの表記法[13]に習い，拡張された学習問題を記述したものである．**複数概念を同時に扱う点**（①），複数の概念記述で正の実例をカバーする，すなわち選言表現を扱う点（ⅰ），相互に共通部分のない概念記述を得る点（ⅲ）が，従来の単一概念の連言表現の学習と比べ新規な点である．

以上の理解の助けとするため，飛翔体に関する4種類の概念の学習問題（**図 1**）をここで説明しておく．概念および実例の記述は，パワー属性とウイング属性の組合せからなり，それぞれ図に示す属性値を取る．なお，概念記述の方がより一般的内容の属性値（下線）を含んでいる．照合関係，一般化関係は，図に示すように属性値の階層を表す半順序関係≪に基づき定義される．

学習内容：飛翔体の分類

概念集合：{宇宙船 (SS), 飛行機 (AP), グライダ (GD), その他 (UFO)}

概念記述言語：

[x (パワー属性), y (ウイング属性)]

$x \in$ {レシプロエンジン (re), ジェットエンジン (je), ロケットエンジン (ro), エンジンなし (no), 空気エンジン (ae), 任意パワー (*)}

$y \in$ {主翼のみ (mo), 主翼と補助翼 (ms), 補助翼のみ (so), 翼なし (nw), 主翼あり (hm), 主翼なし (nm), 任意翼 (*)}

実例記述言語：

[x (パワー属性), y (ウイング属性)]

$x \in$ {レシプロエンジン (re), ジェットエンジン (je), ロケットエンジン (ro), エンジンなし (no)}

$y \in$ {主翼のみ (mo), 主翼と補助翼 (ms), 補助翼のみ (so), 翼なし (nw)}

照合関係 → および一般化関係 ＜：

$[x_1, y_1]$, $[x_2, y_2]$ が下記の半順序において

$x_1 \ll x_2$ かつ $y_1 \ll y_2$ なる時

$[x_1, y_1] \rightarrow [x_2, y_2]$ あるいは $[x_1, y_1] < [x_2, y_2]$

≪（上に行くほど大きい）：

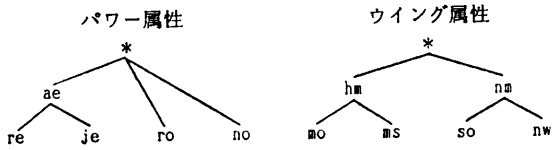

図 1 飛翔体に関する概念の学習問題
Fig. 1 A learning problem in flying objects.

3. 複合多重集約アルゴリズム

3.1 バージョン空間法

前記学習問題は概念記述言語の張る空間内を学習条件を満たす記述を見い出すべく探索する問題といえる．このような探索を扱う枠組みに，単一概念の連言表現の逐次的学習に使用され，全空間をもれなく探索できるバージョン空間法[14]がある．本論文で提案する複合多重集約アルゴリズムも基本的にはこの枠組みを採ることから，この方法に触れておく．

初めに，バージョン空間の定義を示す．

【定義 1】 最小概念記述，最小一般化

実例記述 $e_1, e_2, \cdots, e_k$ を包含する最小の概念記述，すなわち，

$$e_1 \rightarrow c, e_2 \rightarrow c, \cdots, e_k \rightarrow c$$

なる $c \in CL$ のうちで，半順序関係 ＜ の下で最も小さい（一般に複数存在する）概念記述 c を，その実例記述を包含する最小概念記述と呼ぶ．すべての最小概念記述を求めることを，その実例記述を最小一般化するという．また，ある概念記述 c より大きい範囲で，同様の最小記述を求めることを概念記述 c を実例記述 $e_1, e_2, \cdots, e_k$ を含むように最小一般化するという．

【定義 2】 最大概念記述，最小特殊化

実例記述 $e_1, e_2, \cdots, e_k$ を包含しない最大の概念記述，すなわち，

$$\sim(e_1 \rightarrow c), \sim(e_2 \rightarrow c), \cdots, \sim(e_k \rightarrow c)$$

なる $c \in CL$ のうちで，半順序関係＜の下で最も大きい（一般に複数存在する）概念記述 c を，その実例記述を排除する最大概念記述と呼ぶ．ある概念記述 c より小さい範囲で，そのようなすべての最大概念記述を求めることを概念記述 c を実例記述 $e_1, e_2, \cdots, e_k$ を排除するように最小特殊化するという．

【定義 3】 バージョン空間

学習の途中までに与えられた，すべての正の実例の実例記述を包含する最小概念記述の集合を $C\min$ とする．さらに，すべての負の実例の実例記述を排除する最大概念記述の集合を $C\max$ とする．$C\max$ の各要素が $C\min$ のすべての要素よりも大きい，あるいは逆に，$C\min$ の各要素が $C\max$ のすべての要素よりも小さいという条件を満足する時，$C\max$ と $C\min$ にはさまれる空間をバージョン空間，$C\max$ をその上限，$C\min$ をその下限と呼ぶ．

定義3から分かるように，バージョン空間は学習によって求める概念記述のすべての候補を包含している．

バージョン空間法では実例を得る度に上限を最小特殊化，下限を最小一般化により変更して行き，最終的に上限と下限が一致した際の概念記述を学習結果として得る．バージョン空間法の概要を，図1の飛行機の概念の学習を例に**図2**に示す．なお，学習すべき概念は1種類なので図には明記せず，実例記述のみを示す．

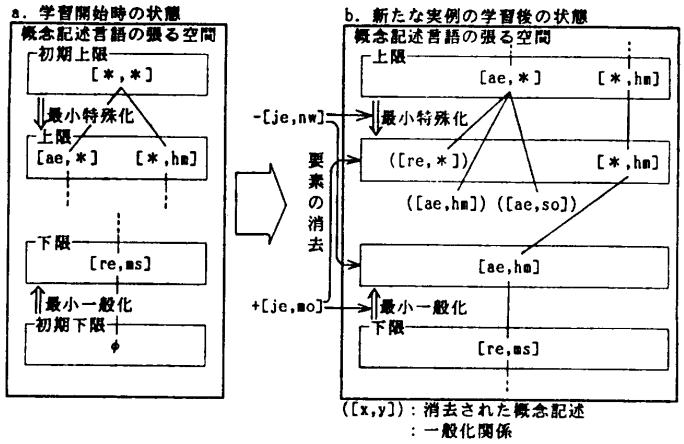

図2 バージョン空間法の概要
Fig. 2 Outline of version space method.

初めに，バージョン空間を概念記述言語の張る全空間とする．すなわち，図2 a. のように上限を概念記述言語の表現し得る最も一般的な概念記述 [*, *] に，下限を空 φ にセットする．その後，実例を一つずつ入力し上限と下限を変更していく．図2 a. の初期上限，下限は，実例 −[ro, nw]，+[re, ms] をこの順で入力した後同図の上限，下限となる．さらに，図2 b. のように学習が進む．

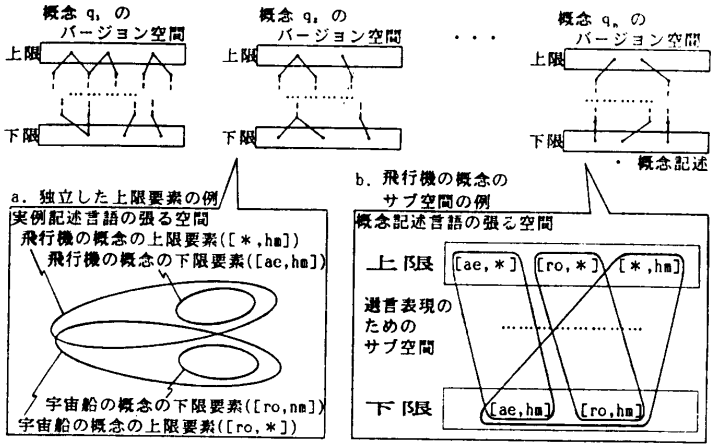

図3 複合多重集約アルゴリズムの枠組み
Fig. 3 Framework of combined multiple convergence algorithm.

ここで負の実例 −[je, nw] が入力されると，矛盾を解消するために，その実例の実例記述を包含する下限の要素をすべて取り除く．次にその実例を排除するように上限の各要素を最小特殊化し（図2の [re, *]，[ae, hm]，[ae, so]，[*, hm])，そのうち下限のすべての要素より大きい（図2の [re, *]，[ae, hm]，[*, hm]）最大の概念記述のみを新たな上限の要素として残す（図2の [re, *]，[*, hm]).

さらに正の実例 +[je, mo] が入力されると，同様に入力された実例の実例記述を包含しない上限の要素をすべて取り除く（図2の [re, *]). 次にその実例の実例記述を包含するように下限の各要素を最小一般化し，そのうち上限のすべての要素より小さい最小の概念記述のみを新たな下限の要素として残す（図2の [ae, hm]).

以上のように，学習条件を満足しない概念記述を消去しながら最終的に一つの概念記述を得る．なお，途中で上限あるいは下限が空となった場合は，求める概念記述がないことを報告する．

以上のアルゴリズムから分かるようにバージョン空間法は，(1)逐次的学習を扱える，(2)可能性のあるすべての概念記述を考慮できる，(3)学習終了時点が分かる，(4)早く収束するといった利点を備えている．

3.2 アルゴリズムのアイデア

3.2.1 アルゴリズムの枠組み

本論文の複合多重集約アルゴリズムもバージョン空間法の枠組みを採るが，複数概念を同時に扱えるように，学習する概念ごとにバージョン空間を持つ（**図3**).

さらに，概念間の独立性を保証するため，上限の要素を，そのバージョン空間で学習する概念のすべての負の実例の実例記述を一つも包含しないように管理するという従来の手法に加え，他のバージョン空間のどの下限要素とも共通部分を持たないように管理する．すなわち，あるバージョン空間の下限の要素の一般化

と同時に，一般化後の下限要素と共通部分を持たないように他のバージョン空間の上限の要素を特殊化する．例えば，図 3 a. に示すように，飛行機の概念の上限要素 [∗, hm] は，宇宙船の概念の下限要素 [ro, nm] が包含する実例記述 [ro, so]，[ro, nw] を包含しないように管理される．このことにより，上限と下限にはさまれる空間は，最終的に共通部分を持たない一つの概念記述に収束し，独立した概念記述を得ることができる．

また，バージョン空間法ではすべての正の実例の実例記述を包含する一つの概念記述（連言表現）を探索するため，一つでも実例記述を包含しない概念記述は捨てていたのに対し，複合多重集約アルゴリズムでは選言表現を扱えるよう，一つでも正の実例の実例記述を包含する概念記述は候補として残しておく．このため，各バージョン空間は上限の要素ごとに幾つかの下限の要素（すべてでなくてよい）を包含するサブ空間を形成する．図 3 b. の例に示す学習途中の飛行機の概念のバージョン空間では，実例記述 [ro, mo]，[re, ms]，[je, mo]，[je, ms] を包含する下限要素 [ae, hm] および上限要素 [ae, ∗] が残され，[ro, mo]，[ro, ms] を包含する下限要素 [ro, hm]，上限要素 [ro, ∗] が残されている．さらに，以上の実例記述すべてを包含する上限要素 [∗, hm] が残されており，それぞれのサブ空間を形成している．複合多重集約アルゴリズムは，これらのサブ空間を狭めて行くことにより最終的に複数の概念記述によってすべての正の実例をカバーする概念の選言表現を得る．

このようにサブ空間ごとに収束を図ることが多重集約の名の由来となっている．一方，複合とは複数のバージョン空間を同時に扱うことを意味する．

複合多重集約アルゴリズムが扱う拡張されたバージョン空間は以下のように定義される．

【定義 4】 拡張バージョン空間

各概念 q_i に対して学習の途中までに与えられたすべての正の実例の集合を $P_i' \subset P_i$，負の実例の集合を $N_i' \subset N_i$ とする．P_i' の幾つかの実例の実例記述を包含する最小概念記述の集合で P_i' をカバーするものを $C\min_i$ とする．すなわち，

$$\forall e : [e, q_i] \in P_i' ; \exists c : [e \rightarrow c, c \in C\min_i].$$

さらに，N_i' のすべての実例の実例記述を排除する最大概念記述の集合で，$j \neq i$ なるすべての j に対する上記 $C\min_j$ と共通部分を持たないものを $C\max_i$ とする．すなわち，

$$\forall e : [e, q_i] \in N_i' ; \forall c : [\sim(e \rightarrow c), c \in C\max_i]$$

かつ

$$\forall j, c : c \in C\min_j, j \neq i ; \forall c' : [c \cap c' = \phi, c' \in C\max_i].$$

各 $C\max_i$，$C\min_i$ の組において $C\min_i$ の各要素より大きい $C\max_i$ の要素が必ず存在する（すべての要素が大きくなくてもよい）時，$C\max_i$ と $C\min_i$ にはさまれる空間を概念 q_i の拡張バージョン空間，$C\max_i$ をその上限，$C\min_i$ をその下限と呼ぶ．また，$c' < c, c' \in C\min_i$，$c \in C\max_i$ なる時，c' を c の配下要素，c のすべての配下要素と c にはさまれる空間を c のサブ空間と呼ぶ．

3.2.2 バックトラック操作

以上の枠組みで従来のバージョン空間法同様に最小一般化，特殊化操作を用いて学習を進める際，次の問題の発生する場合がある．

（1） ある上限の要素を最小特殊化した時に，その要素の配下要素のうち，特殊化後の上限要素より小さくないものが発生する．

（2） 受け取った正の実例の実例記述を包含する要素がその実例の概念の上限に存在しない．

（3） 受け取った負の実例に対し，その実例以外の概念の上限において必要がないのにその実例を排除してしまっている．

複合多重集約アルゴリズムでは，求める概念記述となり得る可能性のあるすべての概念記述を最後まで追及するという立場から，これらの問題を回避するように下記操作を用いてバックトラックを発生させながら学習を進める．

【操作 1】 最大分配

概念記述 c，c' より小さい最大の概念記述を c，c' の最大共通概念記述と呼ぶ．概念記述 $c_1, c_2, \cdots, c_k$ の各々に対し，概念記述 c との最大共通概念記述を求めることを，概念記述 c を $c_1, c_2, \cdots, c_k$ に最大分配するという．

【操作 2】 最大一般化

実例記述 $e_1, e_2, \cdots, e_k$ を排除する最大概念記述のうち，実例記述 e を包含し，概念記述の集合 $C_1, C_2, \cdots, C_l$ のどの要素とも共通部分を持たないすべての最大概念記述を求めることを，実例記述 e を上記実例記述および概念記述の集合を排除する範囲で最大一般化するという．

前述の問題に対し，これらの操作を用いて以下のバックトラックを発生させる．

(1)の場合，小さくない配下要素を特殊化後の上限要素に最大分配してその配下要素として登録し，新たなサブ空間で概念の選言表現として学習を続ける．

(2)の場合，この問題が発生するのは，他の概念の下限が，受け取った実例の実例記述を包含するように既に一般化されているためである．そこで，まずその実例記述を排除するように他の概念の上限を最小特殊化して下限に最大分配操作を発生させる．これで他の概念の下限が狭まり問題となっている上限を拡げることが可能となる．したがって，受け取った実例の実例記述をその概念のこれまでの負の実例の実例記述および他の概念の下限を排除する範囲で最大一般化した要素を上限に加え上限を拡げる．その後実例記述を最小一般化してその配下要素として登録し，新たなサブ空間を形成して学習を続ける．

(3)の場合，上記と同様に実例を排除している上限を拡げる．すなわち，前記(3)となる概念ごとに，その概念のこれまでの負の実例の実例記述および他の概念の下限を排除する範囲で受け取った負の実例の実例記述を最大一般化した要素を上限に追加することで問題を解消する．

以上の理解の助けとして図1の学習問題の学習過程で発生する上記操作の例を図 4，5 に示しておく．

図4は飛行機の概念の上限要素が最小特殊化された際に発生する最大分配操作の例を示している．実例記述 [no, ms] を排除するために，上限要素 [＊, hm] が [ae, hm]，[ro, hm]，[＊, mo] に特殊化される．この際，その配下要素 [＊, hm] がこれら特殊化後の要素に最大分配され，新たな配下要素 [ae, hm]，[ro, hm]，[＊, mo] として登録されている．

図5はグライダの概念の上限が新たな実例記述を受け入れるように最大一般化操作を発生させる場合の例を示している．実例記述 [no, mo] を受け入れるように，この記述をこれまでの負の実例の実例記述（この例にはない）および他の概念の下限要素 [ae, nm]，[ae, hm]，[ro, nm]，[ro, hm] を排除する範囲で最大一般化した要素 [no, ＊] を上限に追加している．さらに，同実例記述を最小一般化した要素 [no, hm] をその配下要素として登録し，新たなサブ空間を形成している（古いサブ空間は吸収される）．

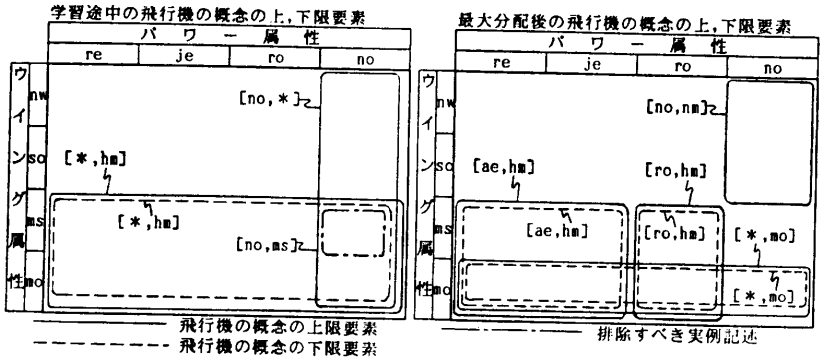

図 4 最大分配操作の例
Fig. 4 An example of maximum distribution operation.

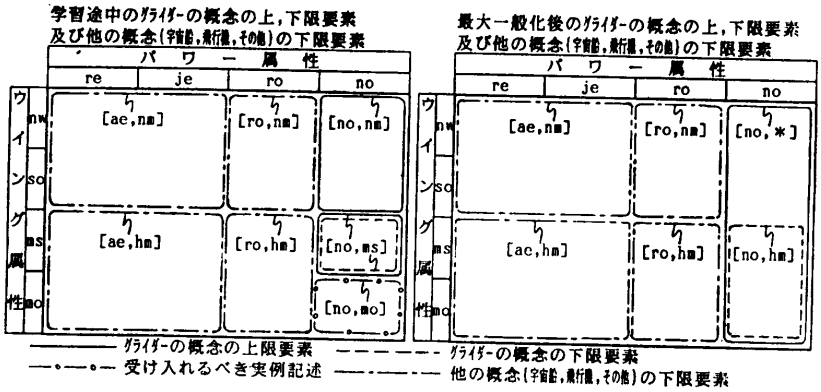

図 5 最大一般化操作の例
Fig. 5 An example of maximal generalization operation.

複合多重集約アルゴリズムは，従来の最小一般化，特殊化操作と上記操作を用いて各拡張バージョン空間の上限と下限を変更して行き，最終的に概念ごとの独立した選言表現を得る．

3.3 アルゴリズムの詳細

複合多重集約アルゴリズムの処理フローを**図6**に示す．図中，下線部分が前節で述べた問題を回避するためのバックトラック処理である．以下，各処理を説明する．

［処理1］ 各概念の拡張バージョン空間の上限を最も一般的な概念記述に，下限を空に初期化する．すなわち，$\forall i$; $C\max_i$ = {概念記述言語 CL の最も一般的な文}，$C\min_i = \phi$.

［処理2］ 新しい一つの実例 $\pm[e, q_m]$ を受け取る．

［処理3］ q_m の拡張バージョン空間に対して以下の処理を行う．

①eを包含し配下要素を持たない$C\max_m$の全要素 c_j に対し，e を最小一般化し，このうち c_j より小さいものを c_j の配下要素として $C\min_m$ に登録する．

②e を包含し配下要素 c_k を持つ $C\max_m$ の全要素 c_j に対し，各 c_k を e を含むように最小一般化し，このうち c_j より小さく最小のものを新たな c_j の配下要素として $C\min_m$ に登録する．

［処理4］ q_m 以外のすべての拡張バージョン空間（添字 i）のそれぞれの $C\max_i$ の要素のうち $C\min_m$ の要素と共通部分を持つものを c_j，持たないものを c'_j とし，各 c_j に対してそれぞれ以下の処理を行う．

①c_j を $C\min_m$ のすべての要素を排除するように最小特殊化し，このうち各 c'_j より小さくないもののみを新たな要素として $C\max_i$ に登録する（$c_{j_1}, c_{j_2}, \cdots$）．

②c_j の各配下要素 c_k をそれぞれ $c_{j_1}, c_{j_2}, \cdots$ に最大分配し，このうち各 $c_{j_1}, c_{j_2}, \cdots$ ごとに最大のもののみを，その配下要素として新たに $C\min_i$ に登録する．

［処理5］ q_m 以外のすべての拡張バージョン空間（添字 i）に対してそれぞれ処理4の処理を行う．ただし，$C\min_m$ に換え e を用いる．

［処理6］ q_m の拡張バージョン空間に対して以下の処理を行う．

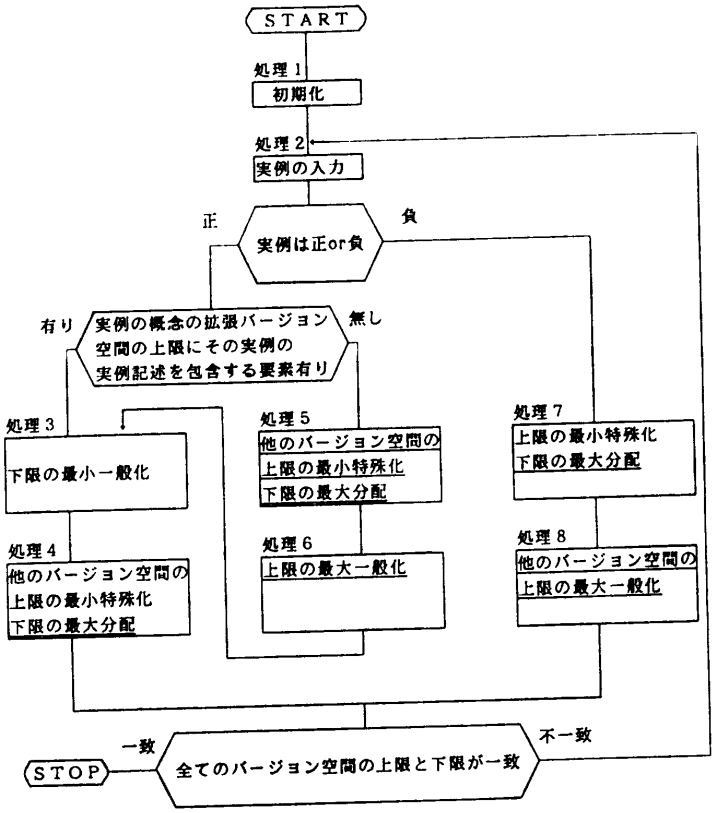

図6 複合多重集約アルゴリズムの処理フロー
Fig. 6 Flow chart of combined multiple convergence algorithm.

①e を q_m のこれまでのすべての負の実例 $-[e', q_m]$ の e' およびすべての $C\min_i$, $i \neq m$ を排除する範囲で最大一般化し，これを $C\max_m$ に登録する（$c_{m_1}, c_{m_2}, \cdots$）．この際，$c_{m_1}, c_{m_2}, \cdots$ より小さい $C\max_m$ の要素 c'_m を削除する．また，c'_m が配下要素を持っておれば，これを c'_m より大きな $c_{m_1}, c_{m_2}, \cdots$ の配下要素として $C\min_m$ に新たに登録する．

［処理7］ q_m の拡張バージョン空間に対して処理4の処理を行う．ただし，$C\max_i$ を $C\max_m$ に，$C\min_i$ を $C\min_m$ に，$C\min_m$ を e に換える．

［処理8］ q_m 以外のすべての拡張バージョン空間（添字 i）に対してそれぞれ処理6の①の処理を行う．ただし，添字 i と m を入れ換える．

3.4 アルゴリズムの性質

前節に示した各処理に対し以下がいえる．なお，$|C\max_i| \equiv \{e \mid e \to c, c \in C\max_i\}$，$|C\min_i| \equiv \{e \mid e \to c, c \in C\min_i\}$．

①最小一般化前，後のある概念の下限を $C\min_i$，$C\min_i'$ とすると，$|C\min_i| \subseteq |C\min_i'|$（処理3）．

②最小特殊化前，後のある概念の上限を $C\max_i$，C

max_i' とすると，$|C max_i'| \subseteq |C max_i|$（処理 4, 5, 7）.

③最大分配前，後のある概念の下限を $C min_i$, $C min_i''$，その概念のこれまでの正の実例の実例記述の集合を $\underline{P}i$ とすると，$\underline{P}i \subseteq |C min_i''| \subseteq |C min_i|$（処理 4, 5, 7）.

④最大一般化前，後のある概念の上限を $C max_i$, $C max_i''$，その概念のそれまでの負の実例の実例記述の集合を $\underline{N}i$，その概念以外の全下限がカバーする実例記述の集合を $\cup |C min_j|$，さらに実例記述言語のすべての文の集合を $|EL|$ とすると，$|C max_i| \subseteq |C max_i''| \subseteq |EL| - \underline{N}i - \cup |C min_j|$（処理 6, 8）.

⑤任意の時点の上，下限に対し，$|C min_i| \subseteq |C max_i|$.

①，②は明らかであろう．③の右側の⊆は最大分配の定義から明らか．また，アルゴリズム上各 $|C min_i|$ は共通部分を持たないように管理されることから，処理 4，5，7 の最小特殊化の排除する実例記述は，$\underline{P}i$ の要素でないものに限られる．よって，要素概念記述（2 章）の存在を仮定しているので，最小特殊化後の上限は $\underline{P}i$ をカバーする（少なくとも排除する実例記述を包含しない概念記述の候補として $\underline{P}i$ の要素に対する要素概念記述がある）．したがって，$\underline{P}i \subseteq |C min_i|$ ならば，最大分配する上，下限要素の共通概念記述として少なくとも $\underline{P}i$ の要素に対する要素概念記述が存在するので，$\underline{P}i \subseteq |C min_i''|$ がいえる．一方，初めて最大分配が発生する時の下限は，$\underline{P}i \subseteq |C min_i|$ となっているので③が常に成り立つ．④の左側の ⊆ は最大一般化の定義から明らか．一方右側の ⊆ は，処理 4，7 により上限は常に $|C max_i| \subseteq |EL| - \underline{N}i - \cup |C min_j|$ に保たれること，ならびに最大一般化の際の排除範囲が $\underline{N}i$ および他の概念の下限であることから明らか．⑤は，バックトラック処理（図 6 下線）により保証される．

これらのことから次の性質が導かれる．

（1） アルゴリズムは収束し，空でない解を得る．

上記から一つの実例の学習処理後常に，$\underline{P}i \subseteq |C min_i| \subseteq |C max_i| \subseteq |EL| - \underline{N}i - | \cup |C min_j|$ が成り立つ．ここで，正の実例が入力される度に $\underline{P}i$ は大きく，負の実例および他の概念の正の実例が入力される度に $|EL| - \underline{N}i - | \cup |C min_j|$ は小さくなる．したがって，最終的には上，下限が一致し空でない解を得る．

（2） 解の完全性，無矛盾性が保証される．

$\underline{P}i \subseteq |C min_i| \subseteq |C max_i| \subseteq |EL| - \underline{N}i$ より明らか．

（3） 得られた解が独立．

任意の i, j に対し，$|C min_i| \subseteq |C max_i| \subseteq |EL| - |C min_j|$ となることから明らか．

以上機能面の性質を述べた．性能面では，次の点から，応用対象によってはかなりの計算の必要なことが予想される．

（a） バージョン空間法は，問題の規模に伴って組合せ的に計算量が増加する．

（b） 提案アルゴリズムは，バックトラック処理の計算も必要．

処理性能が問題となる応用システムでは，対象の特性に合せたヒューリスティクスの導入，インプリメント上の工夫，並列計算機の導入等の検討を行うことが重要と考える．

3.5 非独立な概念記述の学習法および類似手法との関連について

前節のように，提案アルゴリズムにおいて解の独立は，3.3 節の処理 4 および最大一般化時の排除範囲から保証される．この処理の条件をゆるめることで相互に独立でない概念記述を得ることもできる．

例えば，処理 4 の①にて c_j の最小特殊化時の排除対象を $C min_m$ に換え e とすると，独立ではないが他の概念の正の実例の実例記述を排除している解が得られる．

また，処理 4 を廃止すると，実際には 3.2.2 項の（2），（3）の問題が発生せず，処理 5，6，8 も必要なくなる．これは，Murray の方法[11]で個別に各概念の学習を進める場合と一致し，概念ごとの選言表現で，一般には相互にオーバラップした解が得られる．

さらに，最大分配も廃止し矛盾する要素を上，下限から取り除く操作を行うと，バージョン空間法そのものにて個別に各概念の学習を進める場合と一致し，解は相互にオーバラップした連言表現となる．この際，必ず解が得られる保証はない．

なお，下限要素のみを保持し，正の実例を得る度にその概念の下限要素の最小一般化，他の概念の下限要素との間の最大分配を行う方式（処理 3，4 相当）とすると，Gross の方法[12]に相当するアルゴリズムとなる．この場合，独立な選言表現の解を得るが，学習の終了判定に上限と下限の一致以外の基準が必要となる．

4. 適用事例

通常バージョン空間法は，正の実例のみでは容易には収束しない．負の実例による上限の特殊化と下限の

一般化とが相まって収束の速い学習が可能となる．しかし，複合多重集約アルゴリズムでは，概念間の独立性を保証する観点から正の実例を得た場合でも，他の概念の拡張バージョン空間の上限が特殊化される．したがって，正の実例のみでも十分収束の速い学習が可能である．以下では，正の実例のみの場合に対して学習例を示しておく．負の実例が存在する場合も同様である．

図7に，図1の飛翔体の概念の学習問題に対する学習例を示す．上段に示す正の実例を順に入力した場合

入力実例：
i+{[re,ms],AP}, ii+{[re,nw],UFO}, iii+{[je,mo],AP}, iv+{[ro,ms],AP}, v+{[ro,nw],SS}, vi+{[ro,so],SS},
vii+{[je,so],UFO}, viii+{[no,ms],GD}, ix+{[no,mo],GD}, x+{[no,nw],UFO}, xi+{[no,so],UFO}

学習経過：

初期状態
宇宙船のバージョン空間
上限 {[*,*]}
下限 {}
飛行機のバージョン空間
上限 {[*,*]}
下限 {}
グライダーのバージョン空間
上限 {[*,*]}
下限 {}
その他のバージョン空間
上限 {[*,*]}
下限 {}

実例 i
宇宙船のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,nm]}
下限 {}
飛行機のバージョン空間
上限 {[*,*]}
下限 {[re,ms]}
グライダーのバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,nm]}
下限 {}
その他のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,nm]}
下限 {}

実例 ii
宇宙船のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,so]}
下限 {}
飛行機のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,hm],[*,so]}
下限 {[re,ms]}
グライダーのバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,so]}
下限 {}
その他のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,mo],[*,nm]}
下限 {[re,nw]}

実例 iii
宇宙船のバージョン空間
上限 {[je,nm],[ro,*],[no,*],[*,so]}
下限 {}
飛行機のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,hm],[*,so]}
下限 {[je,mo],[ae,hm]}
グライダーのバージョン空間
上限 {[je,nm],[ro,*],[no,*],[*,so]}
下限 {}
その他のバージョン空間
上限 {[ro,*],[no,*],[*,nm]}
下限 {[re,nw]}

実例 iv
宇宙船のバージョン空間
上限 {[je,nm],[ro,nm],[no,nm],[*,so]}
下限 {}
飛行機のバージョン空間
上限 {[je,*],[ro,*],[no,*],[*,hm],[*,so]}
下限 {[je,mo],[ro,ms],[*,hm]}
グライダーのバージョン空間
上限 {[je,nm],[ro,nm],[no,nm],[*,so]}
下限 {}
その他のバージョン空間
上限 {[*,nm]}
下限 {[re,nw]}

実例 v
宇宙船のバージョン空間
上限 {[je,nm],[ro,nm],[no,nm],[*,so]}
下限 {[ro,nw]}
飛行機のバージョン空間
上限 {[je,*],[no,*],[*,hm],[*,so]}
下限 {[je,mo],[*,hm]}
グライダーのバージョン空間
上限 {[je,nm],[no,nm],[*,so]}
下限 {}
その他のバージョン空間
上限 {[ae,nm],[ro,so],[no,nm]}
下限 {[re,nw]}

実例 vi
宇宙船のバージョン空間
上限 {[je,nm],[ro,nm],[no,nm],[*,so]}
下限 {[ro,nm],[ro,so]}
飛行機のバージョン空間
上限 {[je,*],[no,*],[*,hm],[ae,so]}
下限 {[je,mo],[*,hm]}
グライダーのバージョン空間
上限 {[je,nm],[no,nm],[ae,so]}
下限 {}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[re,nw]}

実例 vii
宇宙船のバージョン空間
上限 {[ro,nm],[no,nm]}
下限 {[ro,nm]}
飛行機のバージョン空間
上限 {[no,*],[*,hm]}
下限 {[*,hm]}
グライダーのバージョン空間
上限 {[no,nm]}
下限 {}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[ae,nm]}

実例 viii
宇宙船のバージョン空間
上限 {[ro,nm],[no,nm]}
下限 {[ro,nm]}
飛行機のバージョン空間
上限 {[no,nm],[*,mo],[ae,hm],[ro,hm]}
下限 {[*,mo],[ae,hm],[ro,hm]}
グライダーのバージョン空間
上限 {[no,ms],[no,nm]}
下限 {[no,ms]}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[ae,nm]}

実例 ix
宇宙船のバージョン空間
上限 {[ro,nm],[no,nm]}
下限 {[ro,nm]}
飛行機のバージョン空間
上限 {[no,nm],[ae,hm],[ro,hm]}
下限 {[ae,hm],[ro,hm]}
グライダーのバージョン空間
上限 {[no,*]}
下限 {[no,hm]}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[ae,nm]}

実例 x
宇宙船のバージョン空間
上限 {[ro,nm],[no,so]}
下限 {[ro,nm]}
飛行機のバージョン空間
上限 {[no,so],[ae,hm],[ro,hm]}
下限 {[ae,hm],[ro,hm]}
グライダーのバージョン空間
上限 {[no,hm],[no,so]}
下限 {[no,hm]}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[ae,nm],[no,nw]}

実例 xi
宇宙船のバージョン空間
上限 {[ro,nm]}
下限 {[ro,nm]}
飛行機のバージョン空間
上限 {[ae,hm],[ro,hm]}
下限 {[ae,hm],[ro,hm]}
グライダーのバージョン空間
上限 {[no,hm]}
下限 {[no,hm]}
その他のバージョン空間
上限 {[ae,nm],[no,nm]}
下限 {[ae,nm],[no,nm]}

学習結果の文章表現：
・ロケットエンジンを持ち，
主翼の無い飛翔体は
⇒宇宙船である．
・空気エンジンを持ち，
主翼のある飛翔体
あるいは，
ロケットエンジンを持ち，
主翼のある飛翔体は
⇒飛行機である．
・エンジンを持たず，
主翼のある飛翔体は
⇒グライダーである．
・空気エンジンを持ち，
主翼の無い飛翔体
あるいは，
エンジンを持たず，
主翼の無い飛翔体は
⇒何だか分からない．

図7 飛翔体に関する概念の学習例
Fig. 7 An example of learning in flying objects.

の拡張バージョン空間の変化の様子を学習過程に示してある．なお，サブ空間を構成する上限と下限の要素を上下に対応させて示している．各概念の拡張バージョン空間のサブ空間が実例の入力と共に縮小して行く様子が良く分かる．実例 viii, ix の入力時には，実例を受け入れることのできる上限の要素が存在しなくなっており，3.3 節の処理 5，6 の操作により問題を解消し拡張バージョン空間を再構成して学習を続けている．その後，問題なく学習が進み，最終的に各概念に対する独立した選言表現を得ている．

以上のように，バックトラックを発生（上記では実例 viii, ix の入力時）させつつ，各サブ空間内の上限の要素と下限の要素が近づき結果を導けることが分かる．

5. おわりに

機械学習の手法の実用化に向け，複数概念の独立した選言表現の逐次的な学習を可能とする新しいアルゴリズムを提案した．

本アルゴリズムは，概念ごとの複数のバージョン空間および各バージョン空間内の選言表現のための複数のサブ空間を矛盾なく管理し狭めて行くことでもれなく求める概念記述を探索することができる．本アルゴリズムは，機械学習手法の実際問題への応用可能性を大きく拡げたものといえる．

機械学習の実際の問題への応用では，本論文で扱った以外にも実例の誤りの取り扱い[15]等の問題がある．今後，本アルゴリズムと共に広範な対象の現実問題を扱うシステムを構築できることが期待される．

謝辞 本研究の機会を与えていただいた，(株)日立製作所情報事業部次長川崎惇博士，同システム開発研究所堂免信義所長，同研究所副所長春名公一博士ならびに同研究所第 1 部部長中尾和夫博士に感謝いたします．

参考文献

1) 田代ほか：ルール型制御ソフトウェアシステム SCD (Station Coordinator) の開発，情報処理学会論文誌，Vol. 27, No. 5, pp. 552-561 (1986).
2) 都島ほか：流れ作業ライン制御へのルール型制御方式の適用—製鉄所のビレット精製ライン制御への適用，計測自動制御学会論文集，Vol. 21, No. 10, pp. 1113-1120 (1985).
3) 上野ほか：特集：エキスパートシステム，情報処理，Vol. 28, No. 2, pp. 146-236 (1987).
4) Michalski, R. S.: A Theory and Methodology of Inductive Learning, in Michalski, R. S. et al. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. 1, pp. 83-134, Tioga Pub. (1983).
5) Michalski, R. S. et al.: PLANT/ds: An Expert Consulting System for the Diagnosis of Soybean Diseases, *Proc. of European Conf. on Artificial Intelligence ECAI '82*, pp. 133-138 (1982).
6) Michalski, R. S. et al.: The Multi-purpose Incremental Learning System AQ 15 and Its Testing Applicaion to Three Medical Domains, *Proc. of AAAI '86*, pp. 1041-1045 (1986).
7) Mitchell, T. M.: Generalization as Search, *Artif. Intell.*, Vol. 18, pp. 203-226 (1982).
8) Dietterich, T. G. and Michalski, R. S.: A Comparative Review of Selected Method for Learning from Examples, in Michalski, R. S. et al. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. 1, pp. 41-81, Tioga Pub. (1983).
9) Bundy, A. et al.: An Analytical Comparison of Some Rule-Learning Programs, *Artif. Intell.*, Vol. 27, pp. 137-181 (1985).
10) Cohen, P. R. and Feigenbaum, E. A. (eds.): AQ 11, in *The Handbook of Artificial Intelligence*, Vol. 3, pp. 423-427, William Kaufmann Inc. (1982).
11) Murray, K. S.: Multiple Convergence: An Approach to Disjunctive Concept Acquisition, *Proc. of IJCAI '87*, pp. 297-300 (1987).
12) Gross, K. P.: Incremental Multiple Concept Learning Using Experiments, *Proc. of 5th Int. Conf. on Machine Learning*, pp. 65-72 (1988).
13) Subramanian, D. and Feigenbaum, J.: Factorization in Experiment Generation, *Proc. of AAAI '86*, pp. 518-522 (1986).
14) Mitchell, T. M.: Version Spaces: A Candidate Elimination Approach to Rule Learning, *Proc. of IJCAI '77*, pp. 305-310 (1977).
15) 田代，鷹田：自動的ルール獲得における誤り実例の排除法，第 36 回情報処理学会全国大会論文集，pp. 1481-1482 (1988).

(昭和 63 年 9 月 28 日受付)
(平成 元 年 6 月 13 日採録)

田代 勲（正会員）

昭和28年生．昭和51年東京工業大学工学部制御工学科卒業．53年同大学大学院総合理工学研究科システム科学専攻修士課程修了．同年(株)日立製作所入社．システム開発研究所にて，ソフトウェア開発技法，ルール型制御システム，トランザクション処理システム等の事象駆動型システムの制御，学習，知識獲得等の知識処理システムの研究に従事．現在同研究所研究員．昭和60年～61年 Edinburgh 大学 AI 学科に留学．IEEE，電子情報通信学会，計測自動制御学会各会員．

薗田 憲久（正会員）

昭和25年生．昭和47年大阪大学工学部電気工学科卒業．49年同大学院修士課程修了．同年(株)日立製作所に入社．システム開発研究所にてシステム計画技法，システム構造化技法，ペトリネットなどの事象駆動型システム，生産・流通業向情報処理システムなどに関する研究に従事．現在同研究所第1部主任研究員．56～57年 UCLA に留学．工学博士．IEEE，電気学会，電子情報通信学会，計測自動制御学会などの会員．61年度計測自動制御学会論文賞，62年度計測自動制御学会技術賞受賞．