論　文

例からのLISPプログラムの帰納的推論における典型例の検討

正員　犬塚　信博[†]　　正員　高橋　健一[†]　　正員　石井　直宏[†]

A Study on Representative Sample of Inductive Inference of LISP Program from Examples

Nobuhiro INUZUKA [†], Kenichi TAKAHASHI [†] *and* Naohiro ISHII [†], *Members*

あらまし　LISPのS式上の関数を，その有限個の入出力例から推測し，プログラムとして導出しようという問題は帰納的推論の基本的な問題である．本論文では，LISPのプログラムを推測するためのある種の推論アルゴリズムに対し，それに与えるのにふさわしい入出力例の集合(サンプル)を，典型的なサンプルとして特徴づける．この典型的なサンプルは，関数の各側面を記述する最も簡単な例の集合であり，全体として，関数を最も一般的に記述するものである．また，関数が典型性条件と呼ぶ条件を満たすことで，その関数の典型的な例の集合が常に存在することを示す．そのとき，典型的な入出力例の集合を推論アルゴリズムに与えることで，推論アルゴリズムでは意図する関数を出力することを示す．すなわち，典型性を満たす関数を推論アルゴリズムの領域としたとき，ある意味での部分正当性が成り立つことを示す．また，本論文の考え方を用いることで，与えられたサンプルが典型的なものであることを確認できる．これにより任意に与えられた入出力例を，会話的に典型的な例に変換する手続きについても述べる．

1. まえがき

帰納的推論は，いくつかの事例からそれらに潜む規則性を法則として見出す推論方式であり，高次推論の一つとして人工知能の重要な研究分野となっている[(1)]．また，帰納的推論は，知識が真に増加する推論であるという意味で，学習との関連も深い[(2)]．すなわち，帰納的推論は，機械に知的な振舞いをさせようという人工知能の目標には不可欠である．本論文では，帰納的推論の一つの具体的問題である，LISPのプログラムを，その入出力例から生成させるという問題をとりあげた．そこで与えるべき入出力例を特徴づけ，その入出力例を与えたとき生成される関数の性質を明らかにした．また，与えるべき入出力例を求める手続きを考案した．

LISPは記号処理を容易にするために開発されたプログラム言語である．このLISPのプログラムをいくつかの入出力例から推測し生成する問題は，帰納的推論の問題として早くから研究されてきており[(3),(4)]，P. D. SummersはLISPプログラムを例から生成するための一つのアルゴリズムを提案している[(5),(6)]．彼のアルゴリズムは，複数の入出力例を用いることによりそれらの間に規則性を見出すものであり，LISPのデータ構造であるS式の性質がうまく利用されている．この推論アルゴリズムの適用範囲は，LISPのS式を入力および出力とする関数であり，出力が入力の構造にのみ依存する関数である．そして，これをLISPのプログラムとして生成する．我々はこれまでに，このアルゴリズムの能力を拡大するため，彼のアルゴリズムの拡張を試みてきた[(7)]．

本論文では，これらの推論アルゴリズムに与えるのにふさわしいと考えられるサンプル(ある関数の入出力例の集合をその関数のサンプルと呼ぶ)を，典型的なサンプルとして定義する．すなわち，典型的なサンプルは，それを推論アルゴリズムに与えることにより，関数の最も一般的な記述を得ることができる入出力例の集合であり，そこに含まれる入出力例がある意味で最も簡単な入出力例となるものとして定義される．また，我々が提唱する典型性条件と呼ぶ条件を満たす関数に生成の目標とする関数を絞ることで，典型的なサンプルが存在することを示す．更に，典型的なサンプルを推論アルゴリズムに与えることで，推論アルゴリズム

† 名古屋工業大学工学部電気情報工学科，名古屋市
Faculty of Engineering, Nagoya Institute of Technology, Nagoya-shi, 466 Japan

は意図する関数を生成することを示す．

また，我々が考案した任意に与えられた典型的ではないサンプルを，典型的なサンプルに変換する手続きについても述べる．この手続きは，入出力例の欠如および重複を自動的に判定するものであり，また，欠如部分については会話的に外部に求める．

以下，**2.** では本論文で扱う推論アルゴリズムについて述べる．**3.** では関数に対する典型的なサンプルを定義し，関数が満たすべき典型性条件を提示する．また，この典型性条件の下での関数の振舞いについて調べる．**4.** では，任意のサンプルを **3.** で定義した典型的なサンプルへ変換する手続きを与える．

2. 例からの LISP プログラムの推論

本章では，与えられた入出力例から LISP プログラムを推論するアルゴリズムのあるクラスについて述べる．これがこれから検討を加える対象である．この推論アルゴリズムのクラスは，P. D. Summers の方法[(5),(6)]や，我々が提案したその方法の拡張[(7)]を含んでおり，これらの方法を抽象したものである．このクラスのアルゴリズムでは，ある LISP プログラムの複数個の入出力例が与えられることで，それらの間の規則性を見出し，その規則性を一般化することでプログラムを生成する．

2.1 推論の対象

LISP では，S 式と呼ばれるデータ構造を用いて一切のデータを扱う．S 式は，nil を含めた可算無限個のアトムの集合 A により，次のように構成される．すなわち，① $a \in A$ ならば a は S 式であり，② s_1，s_2 が S 式ならば $(s_1 . s_2)$ は S 式である．②による表現を点対と言う．また，点対で表された $(s . \mathrm{nil})$ を (s) と書く．更に，$(s_1 . (s_2 . \cdots . (s_n . \mathrm{nil}) \cdots))$ を $(s_1 s_2 \cdots s_n)$ と書く．S 式の集合を S と表す．

ここでは，S 式上の演算として car，cdr，cons のみを，そして S 式上の述語として atom のみを用いることにする．これらは通常どおりの意味，$\mathrm{car}[(s_1 . s_2)] = s_1$，$\mathrm{cdr}[(s_1 . s_2)] = s_2$，$\mathrm{cons}[s_1 ; s_2] = (s_1 . s_2)$，$\mathrm{atom}[s] = \mathrm{true} \Longleftrightarrow s \in A$ で用いる．

本論文で扱う推論アルゴリズムは，S 式から S 式への関数を対象とする．但し，その出力がただ入力の構造のみに依存するものを考え，入力した S 式に含まれるアトムの個々の性質に出力が依存するような関数は，ここでの対象としない．故に，アトムを数値として見る算術的関数や，リスト中に重複するアトムを削除する関数など，個々のアトムを識別する必要のある関数はここでの考察の範囲にない．従って，このとき述語 eq を用いる必要はない．また，基本関数 car，cdr，cons と，述語 atom のみを用い，合成と，条件分岐および再帰の制御構造を考慮するだけで計算論的に完全であることが知られている[(5)]．

2.2 推論方法

ある関数に対して，いくつかの入出力例が次のような形で与えられる．

$$
\begin{aligned}
EX = \{ & (\ \ ) && \rightarrow (\ \ ) ; \\
& (A) && \rightarrow (A) ; \\
& (A\ \ B) && \rightarrow (B\ \ A) ; \\
& (A\ \ B\ \ C) && \rightarrow (C\ \ B\ \ A) \}
\end{aligned}
$$

各 $x \rightarrow y$ は，この関数に x を入力として与えることで，y が出力されることを示している．このサンプル EX は，入力した S 式をリストとして反転するような関数を表わしていると考えられる．このとき，次に述べる手順に従うことで，こうした入出力例から関数をプログラムとして導くことができる．

手順を説明する前に若干の定義を行う．次の(1)～(4)のみで定まる集合 FF の要素を関数フラグメントと呼ぶ．

(1) $a \in A$ ならば，$\lambda x.a \in FF$．

(2) $\lambda x.x \in FF$．

(3) $f \in FF$ ならば，
$\lambda x.\mathrm{car}[f[x]] \in FF$，$\lambda x.\mathrm{cdr}[f[x]] \in FF$．

(4) $f_1, f_2 \in FF$ ならば，
$\lambda x.\mathrm{cons}[f_1[x] ; f_2[x]] \in FF$．

また，S 式 x と関数フラグメント f が独立であるとは，x に出現する nil 以外のどんなアトムも，f にアトムとして含まれないことを言う．更に，関数フラグメント f が入出力例 $x \rightarrow y$ に対して正規であるとは，① $y = f[x]$ を満たし，② x に出現するすべての nil をそれぞれある任意の S 式に置き換えた S 式を x' とすると，$f[x'] = y$ を満たすことを言う．

[推論アルゴリズム] 与えられたサンプル $\{x_i \rightarrow y_i \mid 1 \leqq i \leqq n\}$ に対して，次のステップ 1 からステップ 5 を行う．

(ステップ 1) 各入出力例 $x_i \rightarrow y_i$ に対して，正規で x_i に独立な関数フラグメント $f_i\ (1 \leqq i \leqq n)$ を求める．入出力例から関数フラグメントを求めることは，入力と出力との関係を記述することである．ステップ 3 で入出力間の関係に規則性を見つけるため，この関数フラグメントは最大限一般的なものを導いておく必要がある．入力と独立である関数フラグメントを導くのはこの理

由による．

(ステップ2)

$$p_i[x_j]=\begin{cases}\text{true} & (i=j)\\ \text{false} & (i<j)\\ (\text{その他の } i,\ j \text{ には任意})\end{cases}$$

なる述語 $p_1, \cdots, p_n$ を求める．すなわち，j 番目の入出力例の入力S式で x_j へ，述語 $p_1, p_2, \cdots$ を順に適用してゆくと，p_j で初めてtrueとなる．ここで，述語はステップ1の(2)，(3)のみによって導かれた関数フラグメントに，atomを合成することで作られる．

ステップ1および2によって求められた関数フラグメントおよび述語を，次の関数 F_n のようにマッカーシの条件式によって組み合わせる．これにより，与えられた例に関しては正しく計算できる関数が得られる．

$$\begin{aligned}F_n=\lambda x.[&p_1[x]\to f_1[x];\\ &p_2[x]\to f_2[x];\\ &\cdots\\ &p_n[x]\to f_n[x]]\end{aligned}$$

(ステップ3) 関数フラグメント $f_1, \cdots, f_n$ の間に繰返し的な規則性を見出す．

(ステップ4) 述語 $p_1, \cdots, p_n$ の間に繰返し的な規則性を見出す．

(ステップ5) ステップ3および4で求めた規則性が $i>n$ でも成り立つとして関数を一般化し，この関数と等価なプログラムを生成する．すなわち，得られる関数は次の関数と等価である．

$$\begin{aligned}F=\lambda x.[&p_1[x]\to f_1[x];\\ &p_2[x]\to f_2[x];\\ &\cdots\\ &p_n[x]\to f_n[x];\\ &\cdots\quad]\end{aligned}$$

□

今後このクラスの推論アルゴリズムを対象とする．このクラスの推論アルゴリズムが推論に成功して生成する関数は少なくとも与えられたサンプルに無矛盾である．このアルゴリズムが行う一般化については次章で述べる．

3. 例の典型性と推論アルゴリズム

本章では，**2.** で述べた推論アルゴリズムによって入出力例から関数を生成する場合，アルゴリズムに与えるべき例の基準を明らかにする．

3.1 準　　備

本論文では，S式の上の関数を推論対象とするアルゴリズムについて考えているが，議論の煩雑を避けるためS式に制限を加える．この制限付きS式の集合を S_0 とし，推論の対象を S_0 から S_0 への関数とする．すなわち S_0 は，次の(1)，(2)のみによって帰納的に定義される．

(1) $\text{nil}\in S_0$.

(2) $x\in S_0\cup A-\{\text{nil}\}$，$y\in S_0$ ならば $(x.y)\in S_0$.

すなわち，S_0 は，アトムを除く，点対による表現を用いなくとも表せるようなS式のすべて，およびnilを含み，それのみを含む．nilにはリストにおいて終点を示すという特別な働きがある．そのため，リストを扱う関数では，関数フラグメントをnilと独立なものとして書くことが一般にできない．S_0 への制限は，nilの使用をこのリストの終点を示す用途に限定するためのものである．

S式を S_0 へ制限することにより，S式上の関数として扱えないものもある．しかし，計算論的な関数の表現力としては変化ない．

S_0 の上の置換を $\{a_1/x_1, \cdots, a_n/x_n\}$ と表す．但し，$a_1, \cdots, a_n\in A-\{\text{nil}\}$，$x_1, \cdots, x_n\in S_0\cup A-\{\text{nil}\}$ である．こうした置換を $\sigma, \sigma_1, \sigma_2, \cdots$ 等で記す．置換 σ のS式 $s\in S_0$ への適用は $s\sigma$ と書き，s に出現する各アトム $a_1, \cdots, a_n$ を $x_1, \cdots, x_n$ へ置き換えることを意味する．σ_1 と σ_2 の合成は $\sigma_1\sigma_2$ と表し，$s(\sigma_1\sigma_2)=(s\sigma_1)\sigma_2$ である．この置換の適用は S_0 上で閉じている．つまり，任意の置換 σ に関して，$s\in S_0$ ならば $s\sigma\in S_0$ である．

次に，S_0 にS式の構造の簡単さを測る尺度を導入する．その前に写像 form[·] を考える．form[·] はすべてのアトムを同一視するために，単一の記号 ω に移す．これにより S_0 の構造的性質に注目する．この写像 form[·] による S_0 の像 $\{\text{form}[s] \mid s\in S_0\}$ を SF_0 と書く．また，S_0 ではcdr部にあるアトムは必ずnilであるので，SF_0 の要素もcdr部はnilと考え，やはり点対を用いずに表示する．特に，$\text{nil}\in S_0$ についてはform[nil]$=\omega$ を () で表示する．例えば，

$$\begin{aligned}&\text{form}[(A\quad(B)\quad C)]\\ &\quad=\text{form}[(A.((B.\text{nil}).(C.\text{nil})))]\\ &\quad=(\omega.((\omega.\omega).(\omega.\omega)))\end{aligned}$$

である．これを $(\omega(\omega)\omega)$ と書く．

そして，SF_0 の上に次のような関係 $\leqq$ を考える．これが半順序であることは容易に確認される．

[定義1] SF_0 の上の関係 $\leqq$ は，次の(1)，(2)のみで定まる．

(1) 任意の $t\in SF_0$ に対して，$\omega\leqq t$.

(2) 任意の $t_1, t_2, t_3, t_4 \in SF_0$ に対して，
$t_1 \leqq t_2$, $t_3 \leqq t_4$ ならば，$(t_1.t_3) \leqq (t_2.t_4)$. □

$t_1, t_2 \in SF_0$ は $t_1 \leqq t_2$ のとき，t_1 のある ω を $(\omega.\omega)$ に書き換える操作を繰り返し行うことで，t_1 から t_2 に移ることができる．このとき，この操作を行う回数を t_1 と t_2 の距離と呼ぶ．

また，SF_0 は $\leqq$ によって束をなす．すなわち，任意の $t_1, t_2 \in SF_0$ に対し，その上限 $t_1 \cup t_2$，下限 $t_1 \cap t_2$ を次のように求めることができる．

$$t_1 \cup t_2 = \begin{cases} t_1 & ; t_2 = \omega \\ t_2 & ; t_1 = \omega \\ (a \cup c.b \cup d) & ; t_1 = (a.b),\ t_2 = (c.d) \text{ のとき} \end{cases}$$

$$t_1 \cap t_2 = \begin{cases} \omega & ; t_1 = \omega \text{ or } t_2 = \omega \\ (a \cap c.b \cap d) & ; t_1 = (a.b),\ t_2 = (c.d) \text{ のとき} \end{cases}$$

この束 SF_0 のようすを図1に示す．

3.2 入力の型と典型的入力

推論アルゴリズムは，与えられた各入出力例から，それらの入出力例を規定する関数に関する情報を得る．従って，関数の情報を適切に効率よく伝えるには，この入出力例が代表的な例であるべきである．本節では典型的入力という概念を導入することで，無数にある入出力例から代表的な例を選択する基準を与える．

関数 $\boldsymbol{F}$ に対して，同じ振舞いをする入出力例を与えるようなS式ごとに，S_0 を分割する．

[定義2] 関数 $\boldsymbol{F}: S_0 \to S_0$ に関する S_0 上の関係 $=_F$ は，同じく関数 $\boldsymbol{F}$ に関する S_0 上の関係 $\simeq_F$ の推移・対称閉包である．但し，$s_1, s_2 \in S_0$ に対して，$s_1 \simeq_F s_2$ が成り立つのは次の場合であり，そのときに限る．

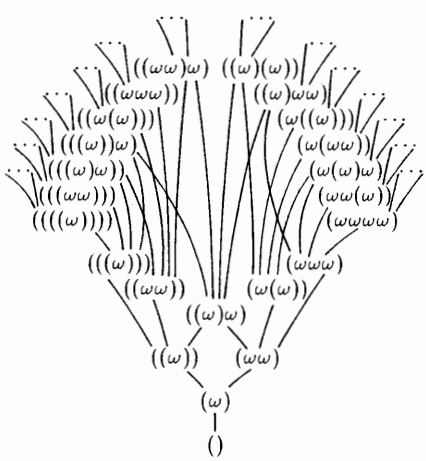

図1 束 SF_0
Fig. 1 Lattice SF_0.

$\exists \sigma, s_1\sigma \Rightarrow s_2$ かつ $\boldsymbol{F}[s_1]\sigma = \boldsymbol{F}[s_2]$

ここで，$s = s'$ は s と s' の文字面が全く等しいことを表す．また，$s \Rightarrow s'$ は s に出現するいくつかの nil をそれぞれある S_0 のS式に置き換えることで，s' が得られることを示す． □

例えば，前に挙げたサンプル EX で示唆されるような，入力S式をリストとして反転する関数 REV を考える．

REV[(A B C)]=(C B A)
REV[(D (E) F)]=(F (E) D)
REV[X X (Y Z))]=((Y Z) X X)

この関数の上の二つの入出力例は，置換 $\sigma = \{A/D, B/(E), C/F\}$ によって，関数 $=_{\mathrm{REV}}$ を満たすことがわかる．つまり，$(A\ B\ C) =_{\mathrm{REV}} (D\ (E)\ F)$ である．同様に，$(A\ B\ C) =_{\mathrm{REV}} (X\ X\ (Y\ Z))$ なので，$=_{\mathrm{REV}}$ の推移性，対称性より $(D\ (E)\ F) =_{\mathrm{REV}} (X\ X\ (Y\ Z))$ も成り立つ．

また，この関係 $=_F$ において，入力側は $s_1\sigma = s_2$ ではなく $s_1\sigma \Rightarrow s_2$ としたのは，リストの長さが異なっても同じ振舞いと見ることを可能にするためのものである．例えば，次の関数 SND を考える．SND は nil に対しては nil を，長さ1のリストに対しては第1要素を，長さ2以上の場合は第2要素を見る．これがアトムであればこのアトムを，リストで長さが1ならば第1要素を，長さが2以上であればその第2要素を見る，と定義される関数である．

SND[(A B C)]=B
SND[(A (B (C) D))]=C
SND[(A (B (C D) E F) G)]=D

このとき，例えば $\sigma = \{A/X, B/Y, C/Z\}$ とすると $(A\ (B\ C))\sigma \Rightarrow (X\ (Y\ Z\ U)\ V)$ であり，$\mathrm{SND}[(A\ (B\ C))]\sigma = C\sigma = Z = \mathrm{SND}[(X\ (Y\ Z\ U)\ V)]$ であるので，$(A\ (B\ C)) =_{\mathrm{SND}} (X\ (Y\ Z\ U)\ V)$ である．

関数 $\boldsymbol{F}$ に対するこのような関係 $=_F$ は，明らかに反射律を満たし，また定義から推移律，対称律を満たすので S_0 の上の同値関係である．この関係によって S_0 を同値類に分割する．

[定義3] 関数 $\boldsymbol{F}$ に関する同値関係 $=_F$ による S_0 の各同値類を，数 $\boldsymbol{F}$ の型 (type) と呼ぶ．$s \in S_0$ の属する $\boldsymbol{F}$ の型を $[s]_F$ と書く． □

型は一般に無限個存在する．例えば，前述の関数 REV，SND では，

$[(A\ B\ C)]_{\mathrm{REV}}$

$=\{(A\ \ B\ \ C),(E\ \ F\ \ G),(D\ \ (E)\ \ F),\cdots\}$

$[(A\ \ (B\ \ C))]_{\mathrm{SND}}$

$=\{(A\ \ (B\ \ C)),(X\ \ (Y\ \ Z\ \ U)\ \ V),\cdots\}$

であり，その他のS式も同様に無限個の型に分割される．また，REVやSNDの各型に属するS式は，それぞれ関数REV，関数SNDに対して同じタイプの振舞いを記述する入出力例を与えると考えられる．

型としてまとめられたS式の中から，一つの代表的なS式を選択するために，SF_0上の半順序 $\leqq$ を用いる．

[定義4] 関数 $\boldsymbol{F}$ における，ある型 $[s]_F$ のformによる像 $\mathrm{form}[[s]_F]$ に，半順序 $\leqq$ による最小元 t が存在する場合，$t=\mathrm{form}[\hat{s}]$ となる $\hat{s}\in S_0$ で，nil以外に同一のアトムが2回以上出現しないS式を，関数 $\boldsymbol{F}$ の型 $[s]_F$ における典型的入力(representative input)と言う．関数 $\boldsymbol{F}$ のすべての典型的入力の集合を T_F と表す．

$\hat{s}$ が典型的入力であるとき，入出力例 $\hat{s}\rightarrow\boldsymbol{F}[\hat{s}]$ を，関数 $\boldsymbol{F}$ の型 $[\hat{s}]_F$ における典型例(representative example)と言う． □

例えば，型 $[(A\ \ B\ \ C)]_{\mathrm{REV}}$ のformによる像は，

$\mathrm{form}[[(A\ \ B\ \ C)]_{\mathrm{REV}}]$

$=\{(\omega\ \ \omega\ \ \omega),((\omega)\ \ \omega\ \ \omega),$

$\cdots,(\omega\ \ (\omega)\ \ \omega),\cdots\}$

であり，最小元として $(\omega\ \ \omega\ \ \omega)$ をとる．従って，$(A\ \ B\ \ C)$ 等は典型的入力であり，例 $(A\ \ B\ \ C)\rightarrow(C\ \ B\ \ A)$ は典型例である(図2参照)．

3.3 典型的なサンプル

次に，考慮している推論アルゴリズムに与える際に，全体としてふさわしいと考えられる入出力サンプルを特徴づける．これは前節で挙げたサンプル EX に示されるような入出力例の集合である．そのため，サンプルが満足すべき条件を以下に列挙する．

(a) 関数の特殊な振舞いを網羅している．

(b) そこに含まれる各入出力例は，同じ振舞いをする例の中で最も簡単なものである．

(c) 同じ振舞いをする例を，たかだか一つ含む．

(d) 各例を構造の簡単さの順序で並べたとき，途中に抜けがない．

(e) 推論アルゴリズムが，(可能であれば)規則性を見つけられる程度に十分多くの例を含む．

これらの条件のうち，(b)は前節で述べた典型例の定義に含まれる．そこで，これ以外の条件，および推論アルゴリズムに特有な条件を考慮し，次のように規定することで典型的なサンプルを条件づける．

[定義5] 関数 $\boldsymbol{F}$ のサンプル $EX=\{x_i\rightarrow y_i\mid 1\leqq i\leqq n\}$ は，次の(1)〜(4)を満たすとき，典型的であると言う．但し，$IN=\{x_i\mid 1\leqq i\leqq n\}$ とする．

(1) IN のformによる像 $\mathrm{form}[IN]$ は，関係 $\leqq$ において全順序集合をなす．

(2) $IN-T_F=\phi$．IN の元はすべて典型的入力である．

(3) $\forall t_1\in\mathrm{form}[IN]$,

$\forall t_2\in\mathrm{form}[T_F]-\mathrm{form}[IN]$,

$t_2\nleqq t_1$.

(4) $\forall i,j(i\neq j)$

$\mathrm{form}[x_i]\neq\mathrm{form}[x_j]$. □

(3)は IN に含まれる入力のあるものよりも小さな典型的入力がない．すなわち，与えられた入出力例の間に，与えるべきもので抜けているものがないことを表している．

但し，(e)の条件は推論アルゴリズムにおける規則性発見の方法に依存するため，ここには含めていない．

3.4 関数の典型性条件

次の条件は，関数に対して要請したい条件である．

[典型性条件] 次の条件(1)〜(3)を，関数 $\boldsymbol{F}$ の典型性条件と呼ぶ．

(1) 関数 $\boldsymbol{F}$ の各型のformによる像は，それぞれ半順序 $\leqq$ に関して最小元をもつ．つまり，各型には典型的入力がある．

(2) 関数 $\boldsymbol{F}$ の典型的入力の集合 T_F のformによる像 $\mathrm{form}[T_F]$ は，系列 $t_1,t_2,t_3,\cdots$ によって表される．つ

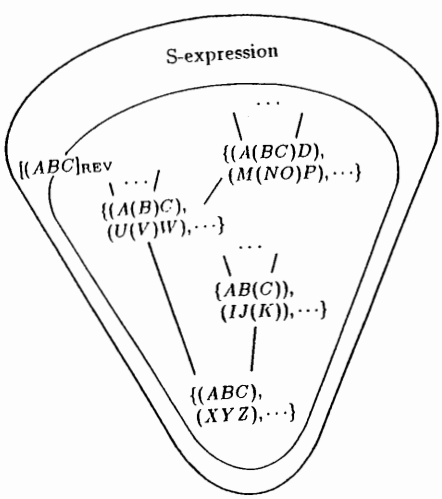

図2 関数REVの型と典型的入力
Fig. 2 Type and representative inputs for function REV.

まり，$\mathrm{form}[T_F]=\lim_{n\to\infty}\{t_i \mid i\leqq n\}$．ここで，$t_i\leqq t_{i+1}$ であり，t_i と t_{i+1} の距離は1である．

（3） 関数 $\boldsymbol{F}$ の任意の型 α，β $(\alpha\neq\beta)$ に対して，

$\forall s_1\in\alpha$，$\forall s_2\in\beta$，$\mathrm{form}[s_1]\nleqq\mathrm{form}[s_2]$，

あるいは

$\forall s_1\in\alpha$，$\forall s_2\in\beta$，$\mathrm{form}[s_2]\nleqq\mathrm{form}[s_1]$． □

この条件を満たす関数の入力S式の型のようすを図3に示す．前述のREVやSNDはこの条件を満たす．

典型性条件のもとでは，典型的なサンプルが存在する．

[定理6] 関数 $\boldsymbol{F}: S_0\to S_0$ が，典型性条件(1)および(2)を満たしていれば，$\boldsymbol{F}$ には任意の濃度 $n(\leqq|\mathrm{form}[T_F]|)$ をもつ典型的なサンプルが存在する．

(証明) $\boldsymbol{F}$ のすべての型 $[s]_F$ から典型的入力を各々一つ選び，それから得られる典型例を合わせることでサンプルを作れば，典型性条件(1)，(2)を仮定したことからこれは定義5の(1)〜(4)を満たしており，典型的なサンプルである．これを EX とする．また，EX の任意の切片 $X[s]=\{x\to y\in EX \mid \mathrm{form}[x]\leqq\mathrm{form}[s]\}$ も典型的なサンプルである．更に，T_F は全順序なので，EX の切片 $X[s]$ は任意の濃度 $n(\leqq|\mathrm{form}[T_F]|)$ をとるように作ることができる． □

上記の定理から，任意の濃度の典型的なサンプルが存在しており，前に非形式的に挙げたサンプルが典型的であるための条件のうち，(e)の十分多くのサンプルを与えるという条件には答える可能性が残される．

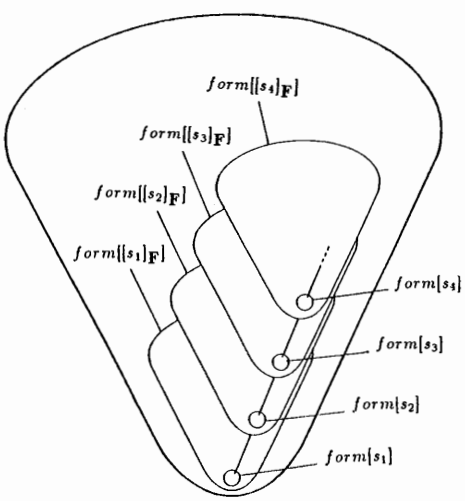

図3 典型性条件を満たす関数の例の順序構造
Fig. 3 Order of examples for function that satisfies the representativity conditions.

更に，典型性条件を満たしている関数を推論アルゴリズムの領域と考えることにより，推論アルゴリズムのある意味での部分正当性が示される．このことを見るために，まず，関数フラグメントおよび述語に関する性質を次に示す．

[性質7] 関数 $\boldsymbol{F}$ に対し，$\boldsymbol{F}$ の任意の典型的入力 $\hat{s}$ は，$\hat{s}$ と独立で，$\hat{s}\to\boldsymbol{F}[\hat{s}]$ に対して正規な関数フラグメント $f\in FF$ に対し，次が成り立つ．

$\forall s=_F\hat{s}$，$f[s]=\boldsymbol{F}[s]$ □

この性質を示すには次の二つの補題が必要である．

[補題8] 任意の関数フラグメント f は，その中に，アトム $a_1,\cdots,a_n$ を含まないならば，置換 $\sigma=\{a_1/s_1,\cdots,a_n/s_n\}$ に対し，

$\forall s\in S_0$，$f[s\sigma]=f[s]\sigma$． □

この補題は，関数フラグメントの構成法に関する数学的帰納法により証明することができる．

[補題9] $\hat{s}$ が関数 $\boldsymbol{F}$ の典型的入力ならば，任意の $s=_F\hat{s}$ に対して，

$\exists\sigma,\ \hat{s}\sigma\Rightarrow s\wedge\boldsymbol{F}[\hat{s}]\sigma=\boldsymbol{F}[s]$．

このとき，$\sigma=\{a_1/s_1,\cdots,a_n/s_n\}$ とすると，$a_1,\cdots,a_n$ は $\hat{s}$ に現れる． □

この補題は，典型的入力は順序 SF_0 において，最小元であること，および，関数 $=_F$ の定義から明らかである．

(性質7の証明) $\hat{s}$ は $\boldsymbol{F}$ の典型的入力なので，補題9より，

$\forall s=_F\hat{s}$，$\exists\sigma,\ \hat{s}\sigma\Rightarrow s\wedge\boldsymbol{F}[\hat{s}]\sigma=\boldsymbol{F}[s]$ (*)

である．また，仮定により，$\boldsymbol{F}[\hat{s}]=f[\hat{s}]$ である．従って，(*)を成り立たせる σ を用いることで，補題8により，

$\boldsymbol{F}[\hat{s}]\sigma=f[\hat{s}]\sigma=f[\hat{s}\sigma]$

である．更に，$\hat{s}\sigma\Rightarrow s$ および f の $\hat{s}\to\boldsymbol{F}[\hat{s}]$ に対する正規性により，$f[\hat{s}\sigma]=f[s]$ が成り立つ．また，同じく(*)より，$\boldsymbol{F}[\hat{s}]\sigma=\boldsymbol{F}[s]$ であるので，$\boldsymbol{F}[s]=f[s]$ が成り立つ． □

[性質10] 典型性条件(1)〜(3)を満たす関数 $\boldsymbol{F}$ に関し，$\mathrm{form}[T_F]$ の系列が典型性条件(2)の条件に従って $t_1, t_2, t_3, \cdots$ と並んでいるとする．このとき，$s_i, s_{i+1}\in T_F(\mathrm{form}[s_i]=t_i,\ \mathrm{form}[s_{i+1}]=t_{i+1})$ が任意の述語 p に対して，$p[s_i]=\mathrm{true}$，$p[s_{i+1}]=\mathrm{false}$ ならば任意の $s_i'\in[s_i]_F$，$s_{i+1}'\in[s_{i+1}]_F$ に対しても $p[s_i']=\mathrm{true}$，$p[s_{i+1}']=\mathrm{false}$ を満たす．

(略証) p は $\lambda x.\ \mathrm{atom}[f[x]]$ と書くことができる．t_i と t_{i+1} との距離が1であることから，述語 p が $p[s_i]=$

true，$p[s_{i+1}]$=falseであるのは，fで示される位置においてs_iはアトムであり，s_{i+1}は$(\omega.\omega)$の形をしている場合に限られる．また，s_i，s_{i+1}が$\boldsymbol{F}$の典型例であること，$\boldsymbol{F}$が典型性条件を満たすことから，それぞれの同じ型の要素s_i，s_{i+1}においてもこのことは成り立つ．

□

以上の性質によって，典型性条件を満たす関数において，次のことが成り立つ．

[定理11] 関数$\boldsymbol{F}$：$S_0 \to S_0$が，典型性条件の(3)を満たしているとき，典型的なサンプルEXを与えることで手続きによって生成された関数Fは，少なくとも$\{s \in S_0 \mid \exists x \to y \in EX,\ \exists s' \in [x]_F,\ \mathrm{form}[s] \leqq \mathrm{form}[s']\}$に属するS式では，意図した関数$\boldsymbol{F}$と入出力関係が等しい．

(証明) 与えられた典型的なサンプルを，$\{x_i \to y_i \mid 1 \leqq i \leqq n\}$ $(\mathrm{form}[x_i] \leqq \mathrm{form}[x_{i+1}])$と表すことができる．このとき生成される関数は次のようなものと等価である．

$$
\begin{aligned}
F = \lambda x.[\ & p_1[x] \to f_1[x]\ ; \\
& p_2[x] \to f_2[x]\ ; \\
& \cdots \\
& p_n[x] \to f_n[x]\ ; \\
& \cdots \qquad]
\end{aligned}
$$

そしてp_iに関しては，$p_i[x_j]$=true$(i=j)$，false$(i<j)$，f_iに関しては$f_i[x_i]=y_i$であった．従って，$F[x_i]=f_i[x_i]=y_i=\boldsymbol{F}[x_i]$である．このことに，性質7，10を合わせることで，各x_iの属する型$[x_i]_F$については正しい振舞いを行うことがわかる．すなわち，$\{s \in S_0 \mid \exists x \to y \in EX,\ s \in [x]_F\}$に属するS式$s$については$F[s]=\boldsymbol{F}[s]$である．更に，典型性条件(3)および典型的サンプルの定義(3)より，$\{s \in S_0 \mid \exists x \to y \in EX,\ \exists s' \in [x]_F,\ \mathrm{form}[s] \leqq \mathrm{form}[s']\} = \{s \in S_0 \mid \exists x \to y \in EX,\ s \in [x]_F\}$がわかる．□

ここで，考察の対象としている推論アルゴリズムが行う一般化について考察しておく．性質7は典型的なサンプルが与えられることで，各例が示す振舞いをそれが属する型全体へ一般化していることを示している．また，型を超えての一般化の性質は，個々のアルゴリズムにおいて関数フラグメントや述語のどのような規則性に注目するかに依存している．

4. 典型的なサンプルへの変換

前章では，例やサンプルの典型性について考察してきた．本章では，任意に与えられたサンプルを典型的なサンプルへ変換する手続きを示す．

推論手続きに与えられるサンプルは，前章で考察した典型的なサンプルであることが望まれる．しかし，必ずしも典型的なサンプルが与えられるとは限らない．そこで，任意に提示されたサンプルの典型性を確認し，典型的なものでなければ典型的になるように変換する手続きを考える．

与えられたサンプルは，SF_0上での半順序に従ってグラフとみなされ，変換手続きによって順次変形される．必要と判断された入出力例は外部に要求される．

[変換手続き] サンプル$EX=\{x_i \to y_i \mid 1 \leqq i \leqq n\}$を変換する．但し$IN=\{x_i \mid 1 \leqq i \leqq n\}$とする．

(1) EXに含まれる各入出力例を節とし，$x_i, x_j \in IN$ $(x_i \neq x_j)$について，$\mathrm{form}[x_i] \leqq \mathrm{form}[x_j]$で，$\mathrm{form}[x_i] \leqq \mathrm{form}[x_k] \leqq \mathrm{form}[x_j]$なる$x_k$ $(\in IN, x_k \neq x_i, x_j)$が存在しないとき，$x_i \to y_i$から$x_j \to y_j$への有向辺をもつ有向グラフ$G$を考える．

(2) nilを入力とする例が与えられていないとき，nilに関する入出力例nil$\to \boldsymbol{F}$[nil]を外部に求め，例として加えることで新たにグラフGを作る．

(3) Gの分岐点(すなわち，出次数が2以上の節)$x_i \to y_i$に対して，その節の子$x_{i1} \to y_{i1}, \cdots, x_{in} \to y_{in}$への辺のうち，各同じ型の子(つまり，$x_{ik} =_F x_i$なる$x_{ik}$)への辺をすべて開放除去する．

また，その分岐点と前に除去した以外の各子$x_{ik} \to y_{ik}$との間に，$\mathrm{form}[x_i] \leqq \mathrm{form}[s] \leqq \mathrm{form}[x_{ik}]$なるS式$s \in S_0$がある場合，$SF_0$上の$\leqq$の意味で最も小さいすなわち$x_i$に近いS式の一つについて，これに関する入出力例$s \to \boldsymbol{F}[s]$を外部に求める．これが$x_i$と異なる型に属するならば例として加え，次に小さなS式についてこの操作を繰り返す．もし，同じ型に属するならば別の最も小さなS式についてこの操作を繰り返す．この操作はx_{ik}に達するまで続けられる．結果のグラフの，入出力例nil$\to \boldsymbol{F}$[nil]の属している連結成分を，新たにGとする．

(4) (3)の操作のために生じた分岐点xについて，(3)と同様，そこから出ている子$x_{i1} \to y_{i1}, \cdots, x_{in} \to y_{in}$への辺のうち，各同じ型の子への辺を開放除去する．結果のグラフの，入出力例nil$\to \boldsymbol{F}$[nil]の属している連結成分を新たにGとする(この時点で，グラフは線形になる)．

(5) グラフGの各節$x \to y$と，その子$x' \to y'$の間に$\mathrm{form}[x] \leqq \mathrm{form}[s] \leqq \mathrm{form}[x']$なる$s \in S_0$があれば，(3)と同じようにして補う．

(6) 例nil$\to \boldsymbol{F}$[nil]から順にGをたどり，その節と

その子が同じ型に含まれる場合，その子への辺を開放除去し，例 nil→$\boldsymbol{F}$[nil] の属している連結成分を新たに G とする． □

この変換手続きにおいて，二つの入出力例が同じ型のものかどうか判断する必要があるが，そのために次の性質を用いる．

[性質 12] 入出力例 $s_1 \to \boldsymbol{F}[s_1]$ に対して正規で s_1 と独立な関数フラグメントを f とする．このとき，form$[s_1] \leqq$ form$[s_2]$ ならば，

$$f[s_2]=\boldsymbol{F}[s_2] \Longleftrightarrow s_1 =_F s_2.$$

(証明) (⇒) $f[s_2]=\boldsymbol{F}[s_2]$ とする．form$[s_1] \leqq$ form$[s_2]$ なので，$s_1\sigma \Rightarrow s_2$ となる置換 σ がある．そこで，補題 8 および f の正規性より，

$$\boldsymbol{F}[s_1]\sigma=f[s_1]\sigma=f[s_1\sigma]$$
$$=f[s_2]=\boldsymbol{F}[s_2]$$

故に，$s_1 =_F s_2$ である．

(⇐) 性質 7 より明らか． □

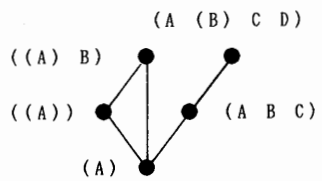

図 4 例の集合 EX のグラフ表現
Fig. 4 Graph representation for sample EX.

つまり，s_1, s_2 (form$[s_1] \leqq$ form$[s_2]$) が同じ型の入力であるかを調べるには，例 $s_1 \to \boldsymbol{F}[s_1]$ から s_1 に独立で正規な関数フラグメントを作り，それを s_2 に適用し，$\boldsymbol{F}[s_2]$ に一致することを確かめればよい．

次の例をもとに典型例に変換するようすを示す．

$\{(A) \to (A)$;
$((A)) \to ((A))$;
$((A)\ B) \to (B\ (A))$;
$(A\ B\ C) \to (C\ B\ A)$;
$(A\ (B)\ C\ D) \to (D\ C\ (B)\ A)\}$

この例をグラフに表すと図 4 のようになる．このグラフの節 '●' はそれぞれ例 (図には入力 s 式のみ記してある) を示すが，図 5 (1) では説明の便宜上節と節の間にあるはずの S 式を入力とした例を，'○' で示す．図 5 の (1)〜(6) はそれぞれ上の手順の (1)〜(6) に対応しており，これらの操作の結果得られた例の集合を次に示す．これは典型的なサンプルである．

$\{() \to ()$;
$(A) \to (A)$;
$(A\ B) \to (B\ A)$;
$(A\ B\ C) \to (C\ B\ A)$;
$(A\ B\ C\ D) \to (D\ C\ B\ A)\}$

5. むすび

本論文では，2. で述べたタイプの推論アルゴリズム

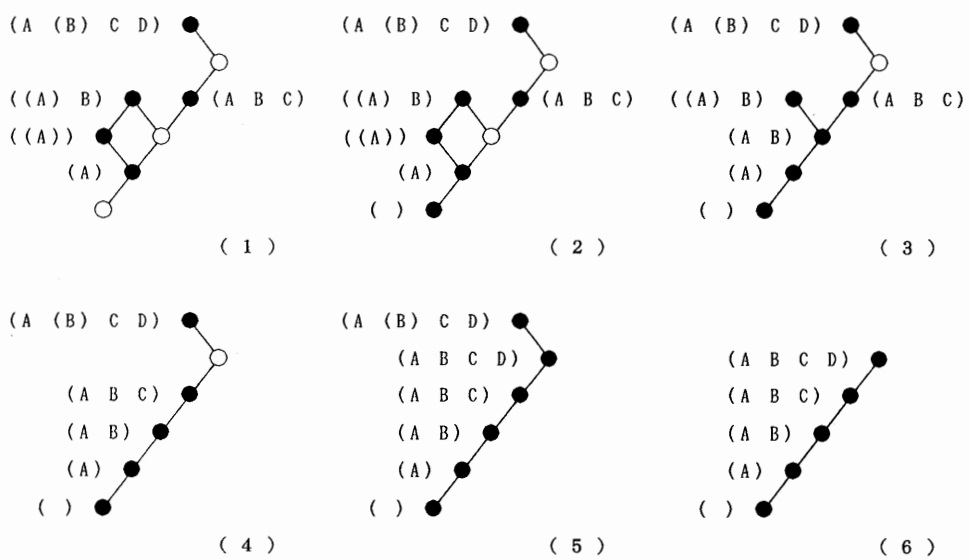

図 5 典型的な例の集合への変換
Fig. 5 Translation into representative sample.

に対して，入出力例として与えるのにふさわしいサンプルを典型的なサンプルとして特徴づけた．そして，このような典型的なサンプルを与えることで推論アルゴリズムが関数を生成したときは，その関数は少なくとも与えられた例と同じ型に属する入力に対しては正しく振る舞うことを示した．例として与えられていない部分については何も言えないが，このことは，帰納的推論の非妥当性からやむを得ないことである．

これらの検討には二つの意味がある．一つは，この種のアルゴリズムが扱うことのできる関数をその振舞いから特徴づけたことであり，もう一つは，与えられたサンプルが正しく関数を推測でき得るものかどうかを確認できるようになったことである．また，半自動的であり会話的であるが，典型例に変換するための手続きを与えることを可能にした．

我々は，関数の典型性条件として典型的サンプルの系列において各典型例間の距離が1であることを挙げた．これは推論手続きにおいて作られる述語の性質から受ける制約である．しかし，この条件は外すべきものであると考えられる．また，**4.** で述べた典型的なサンプルへの変換アルゴリズムについて，問合せの回数は少ないとは言えない．これらの改良は今後の課題である．一般的な帰納的学習に関する基礎的研究は盛んに進められているが，ここで論じたような典型例の特徴付けとその性質の解明は重要であると考える．

謝辞　本研究の一部は平成2年度文部省科学研究費補助金（奨励研究（特別研究員）02952158）により行われた．また，適切な御助言を頂いた査読委員の方に謝意を表します．

文　献

(1) 仁木和久，石崎　俊："概念の帰納的学習"，人工知能学会誌，**3**, 6, pp. 695-703（昭63-11）.
(2) R. S. Michalski, J. G. Carbonell and T. M. Michell: "Machine Learning: An Artificial Intelligence Approach", Morgan Kaufman (1986).
(3) D. Angluin and C. H. Smith: "Inductive inference: theory and methods", ACM Computing Surveys, **15**, 3, pp. 237-269 (Sept. 1983).
(4) 有川節夫，原口　誠："例によるプログラム合成"，知識の獲得と学習，大須賀ほか編，オーム社（昭62）.
(5) P. D. Summers: "A methodology for LISP program construction from examples", J. ACM, **24**, pp. 161-175 (Jan. 1977).
(6) P. D. Summers: "Program construction from examples", Ph. D. Thesis, Dept. Comptr. Sci., Yale Univ., New Haven, Conn. (1975).
(7) 犬塚信博，高橋健一，石井直宏："高階の差分を用いた入出力例からの関数の合成"，信学論(D-II)，**J72-D-II**, 9, pp. 1484-1492（平1-09）.

（平成元年11月20日受付，2年5月14日再受付）

犬塚　信博

昭62名工大・工・情報卒．平1同大大学院博士前期課程電気情報工学専攻了．現在，同大大学院博士後期課程電気情報工学専攻在籍．人工知能，特に帰納的推論に関する研究に従事．

高橋　健一

昭52名工大・工・情報卒．昭54同大大学院修士課程了．同年同大・工・助手．平1同大・電気情報工学科助教授，現在に至る．この間，画像情報処理および画像通信の研究に従事．工博．IEEE，情報処理学会各会員．

石井　直宏

昭38東北大・工・電気卒．昭43同大大学院博士課程電気および通信工学専攻了．同年同大・医・助手．昭50名工大・工・助教授を経て，現在，同大・電気情報工学科教授．この間，しきい値論理，医用情報処理，および非線形処理の研究に従事．工博．