TM-0233

文脈自由言語の帰納的推論

横森　貴
（富士通）

October, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

概要

文脈自由言語を具体例の提示から帰納的に推論する問題が考察される。この問題に対する従来のアプローチの殆どは，対応する文脈自由文法を推論するという方法を論じている。ここでは，正則表現のひとつの拡張と考えられる，文脈自由表現を用いて，文脈自由言語の推論問題に対するアルゴリズムを与える。得られるアルゴリズムは"枚挙による同定"アルゴリズムになっており，従来の文法推論アルゴリズムと比べて非常に簡潔なものになる。

また，文脈自由言語族の部分クラスである準線型文脈自由言語の族が考察され，それに対するより簡潔な推論アルゴリズムが提示される。さらに，推論問題に対する計算量に関する結果が示される。

〔目次〕

1. はじめに

帰納的推論は，部分的な情報から一般化したものを形成する過程として表現することができる。この帰納的推論のメカニズムは，我々が直面する種々の問題解決過程における知識獲得機能を支援する重要な機能の一つであり，機械学習という領域における主要な研究テーマである。

現在までのところ，帰納的推論に関する研究の焦点は主に，有限状態オートマトン，形式文法，一階論理，LISPプログラム等々の領域であり，各領域においてはそれなりに成果が挙げられて来たが，例からのプログラムの自動合成等のより実際的応用問題に対しては，未だ実用的なアルゴリズムは知られていない。

さて，我々は以下の様な帰納的推論のモデルを考える：帰納的推論装置（IID)は与えられた規則のクラスから選ばれたある対象(規則)Rを推論しようとする。この装置には，すべての規則が少なくとも一度は枚挙されるような仕組みが仮定されている。また，対象に関する具体例の情報を提示するためのオラクルが認められている。IID はオラクルに対して具体例を求め，計算し次の推測を出力し，別の具体例に対してまたオラクルに質問し，これを繰り返す。(無限の) 推測の系列 H_1 , H_2 ,...において，あるn があって H_n がR を表しすべてのi>0 に対して H_{n+i}は H_n に等しい時，IID は極限においてR を同定するという。Gold ([Go 67]) によって最初に示唆されたこの極限における同定という概念は，その後の帰納的推論，特に形式言語の推論問題に対する標準的な形式的枠組みを与えている。

極限における同定の非常に単純なアルゴリズムは，枚挙による同定アルゴリズムである。H_1 ,H_2 ,...を可能な推測の実際的な数え上げとする。具体例e_1 , ...,e_k が提示された時,IIDは次の推測として,k個の例すべてと無矛盾な系列のなかの最初のH_i を出力する。具体例の完全な提示を案内にして，この推測の系列は極限において収束する。

さて，Shapiro([Sh 82]) は，一階の述語の領域にこの考えを拡張し，論理的帰結としての例から文を推論するプログラムを作成し，ある程度の成功を収めた。彼は，表現空間を節形式の集合に制限し，かつ枚挙における一階述語言語の記号を固定している。また，推論の目標はエルブランモデルであり，モデルの具体例は基礎原子式の形で与えられた。彼の方法は，多くの領域上の帰納的推論を可能にし，また他のアルゴリズムは適用不可能であるが，一階述語で表現可能な領域へ拡張されている。しかし，Laird （ [La86] ）も指摘しているように，彼の表現空間(一階述語) は必ずしも最も適当というわけではなく，対象を表現する直観的な言語でもない。例えば，節形式を用いて正則集合は定義できるが，正則表現や有限状態図は通常これらの集合の表現としてより直観的である。

この論文の動機となったのは，まさにこの点である。次節以降において我々は文脈自由言語の帰納的推論問題を考察するが，そこでは従来から行われている研究と異なる表現空間を採用する。それにより，この言語族に対する帰納的推論問題とそのアルゴリズムに関するエレガントな議論が可能になる。

2. 帰納的推論の形式的枠組み

帰納的推論という問題を形式化する試みは古くから行われて来たが，言語の帰納的推論の形式的枠組みとしてはGoldによって定式化された“極限による同定”という概念が支配的である。ここでは，最近Laird によって提案された，より抽象化された形式的枠組みを採用する。

抽象化された推論問題とは以下のように定義される。

【定義2.1 】（ [La86] ）

抽象推論問題とは六つ組(D, do, $\mathcal{E}$, h, ASK, EX)であり，

(1) Dは半順序$\geq$で順序付けされた高々可算な集合（意味空間），

(2) doはDの指定された要素（目標），

(3) $\mathcal{E}$は表現の可算集合（表現空間），

(4) h : $\mathcal{E}$→Dは全射，

(5) ASK は$\geq$に関するオラクルでh(e_i) $\geq$h(e_j) ならばASK(e_i, e_j)= 1，さもなければ0 を返す（e_i, e_j $\in\mathcal{E}$），

(6) EXはdoの具体例に関するオラクルで，EX( )= +eならばdo$\geq$h(e)，EX( )=-e ならばdo$\not\geq$ h(e) を意味する（e : doの具体例），

である。

この定式化は，目標の要素に関する具体例が表現の集合の要素と見做せるような問題設定に対して非常に簡潔な形式的枠組みを与えている。実際，これまでに研究されている推論対象の領域の幾つか（論理プログラム，正則表現等）はこの枠組みで議論できる。

さて，推論問題において，具体例の性質及びその提示法は推論能力に大きく影響する事は，研究の初期から良く知られている。([Go67]) 以下の定義は，これまで報告されているこの分野の殆どの仕事において，用いられている。

【定義2.2 】([La86])

do $\geq$h(e)なるすべてのe $\in\mathcal{E}$に対して少なくも一度EX()は究極的に+eを返し，do $\not\geq$ h(e') なるすべてのe' $\in\mathcal{E}$に対して少なくも一度EX()は究極的に-e' を返す時，オラクルEX()はdoの完全提示を与えると云う。

この時，この抽象推論問題を解く一つのアルゴリズムは以下の様になる事が知られている：

《枚挙による同定アルゴリズムＡ０》（［La86］より）

入力:　表現の帰納的可算集合$\mathcal{E}$

$h(e_1) \geq h(e_2)$?のためのオラクルASK($e_1 \geq e_2$?)

doの完全提示のためのオラクルEX()

出力:　H_n が最初のn 個の具体例に対して正しい様な，表現の列 $H_1, H_2, \ldots$

手続き:　$e_1, e_2, \ldots$を$\mathcal{E}$の枚挙とする。

```
i ← 1
examples ← 空集合
do (永久に)
    examples←examples∪EX()    {次の具体例を取れ}
    while  ASK(e_i ≧e?)=1(ある負の例-eに対して) 或いは
           ASK(e_i ≧e?)=0(ある正の例+eに対して)
           i ←i+1
  e_i を次の仮説として出力
```

アルゴリズムＡ０の正当性は以下の様に保証されている。即ち，

【定理2.1 】（［La86］）

アルゴリズムＡ０は極限においてdoを同定する。

さて，言語の推論問題を考える時，形式言語理論に基づく形式化においては，表現空間は形式文法のクラスをとる事が常であり，正則言語の族等に対する推論アルゴリズムの研究は良くなされているが，言語族のクラスが大きくなると，多くの様々な試みにもかかわらず，いまだ有効かつ完全な推論方法は良く分かっていない。（［TA77］，［Wh77］，［榎本他76］，［棚次86］）

本論文において，我々は文脈自由言語の族に対する帰納的推論アルゴリズムを提示する。推論問題の枠組みは，本節で紹介したモデルが用いられる。そのためには，表現空間と具体例の関係がここでの問題設定を満たす様に選ばなければならない。

我々の方法は，以下の点において従来の諸方法とは異なっている：即ち，ここでは表現空間として文脈自由文法のクラスを選択しない。その代わりに，我々は"文脈自由表現"とも言える一種の拡張された正則表現を採用する。これにより，文脈自由言語族の帰納的推論問題は，上記で定義された抽象推論問題のひとつとして定式化され，簡潔な推論アルゴリズムが提示される。

3．文脈自由言語の推論

この節では，先ず表現空間を構成する要素として，文脈自由表現なる概念を導入する。これは，元来Gruska（ [Gr71] ）によって提案された文脈自由言語に対する特徴付けの一表現方法であり，正則言語に対する正則表現の自然な拡張になっている。

3.1 文脈自由表現—拡張正則表現—

まず，形式言語理論における幾つかの基本的な概念と記法について説明する。（ ここでの議論にとって必要最小限の説明にとどめる。この分野のより詳細は，例えば [Sa73] を参照されたい。）

有限アルファベットΣに対して，Σの元からなる有限長の系列全体から成る集合をΣ^*で表す。（ここに長さゼロの列をεと記す。）Σ上の言語Lとは，Σ^*の部分集合を言う。無限アルファベットΓに対して，LがΓ上の言語であるとは，Γのある有限部分アルファベットΣ上の言語である時を言う。

言語L_1とL_2の積$L_1 L_2$とは言語 $\{xy \mid x \in L_1, y \in L_2\}$ を言う。言語Lと整数$i \geq 0$に対して，L^iは以下の様に定義される： $L^0 = \{\varepsilon\}$ （ε：空語），$L^{i+1} = L^i L$。さらに，Lの＊閉包（＋閉包）とは，言語 $L^* = \cup_{i=0}^{\infty} L^i$ （$L^+ = \cup_{i=1}^{\infty} L^i$）を言う。

文脈自由文法Gは四つ組（N，T，P，S）で定義される：ここにNは非終端記号の有限アルファベット，Tは終端記号のアルファベットで$N \cap T = \phi$，S（$\in N$）は初期記号，PはA→wなる形（$A \in N$，$w \in (N \cup T)^*$）の規則の有限集合である。

二つの語x，yに対して，二項関係⇒を以下の様に定義する：$x \Rightarrow y$であるのは，ある$u, v \in (N \cup T)^*$と$A \to w \in P$が存在して，$x = uAv$，$y = uwv$となる時かつその時に限る。$\Rightarrow^*$を⇒の反射的かつ推移的閉包とする。集合$L(G) = \{x \mid S \Rightarrow^* x$かつ$x \in T^*\}$をGによって生成される言語と言う。言語Lが文脈自由言語であるとは，ある文脈自由文法Gが存在して$L = L(G)$となる事をいう。

さて，次に導入される演算はここでの議論において決定的に重要な役割を果たす。

【定義3.1 】（ [Gr71] ）

（i）σをある記号，L_1とL_2を言語とする。L_2のL_1中へのσ代入（ σ-substitution）とは，

$$L_1 \overset{\sigma}{\leftarrow} L_2 = \{x_1 y_1 \cdots x_k y_k x_{k+1} \mid x_1 \sigma x_2 \cdots \sigma x_{k+1} \in L_1, \ \sigma \text{は各} x_i \text{には現れない，かつ} y_i \in L_2 \ (1 \leq i \leq k)\}$$

で定義される言語を言う。

（ii）σをある記号，Lを言語とする時，Lのσ反復（ σ-iteration ）とは，

$$L^{\sigma} = \{x \mid x \in L \cup L \overset{\sigma}{\leftarrow} L \cup L \overset{\sigma}{\leftarrow} L \overset{\sigma}{\leftarrow} L \cup \cdots, \ x \text{はσを含まない}\}$$

なる言語を言う。

［注記］

(1) L_1 がσを含まない時，$L_1 \overset{\sigma}{\uparrow} L_2 = L_1$ とする。

(2) Lを T上の言語とし，Tはσを含まないとする。この時，$L^* = (L\sigma \cup \{\varepsilon\})^{\sigma}$ かつ$L^+ = (L\sigma \cup L)^{\sigma}$である。

さて，文脈自由言語を表現する一方法を導入する。この記法は，正則表現が正則言語に果たすのと同様の性質を，ある意味において文脈自由言語に対して保持しているので，“文脈自由表現”と呼ばれる。

【定義3.2 】(［Gr71］)

Γを（必ずしも有限とは限らない）アルファベット，Γ’ をΓの元のボールドフェイスからなる集合，即ちΓ’ $= \{\boldsymbol{\sigma} \mid \sigma \in \Gamma\}$ ，とする。Γ上の文脈自由表現とはΓ∪Γ’ ∪ $\{\phi\}$ 上の語であり次の条件を満たすものを言う：

（ⅰ）ϕは文脈自由表現である

（ⅱ）もし$a \in \Gamma \cup \{\varepsilon\}$ ならば，aは文脈自由表現である

（ⅲ）もしE_1 ，E_2 が文脈自由表現であり，$\sigma \in \Gamma$ならば，$(E_1 + E_2)$ ，$E_1\ E_2$ そして$E_1\ \boldsymbol{\sigma}$は文脈自由表現である

（ⅳ）（ⅰ）～（ⅲ）を有限回適用して得られるものだけが文脈自由表現である。

各文脈自由表現Eは以下の既約に従ってΓ上の言語を表すと見做す事が出来る。

【定義3.3 】

Γ上の文脈自由表現のクラスからΓ上の言語族への写像｜｜を以下の様に定義する：

（ⅰ）$|\phi| = \Phi$（空集合）

（ⅱ）$|a| = \{a\}$ $(a \in \Gamma \cup \{\varepsilon\})$

（ⅲ）$|E_1 + E_2| = |E_1| \cup |E_2|$，$|E_1\ E_2| = |E_1|\,|E_2|$，そして $|E\boldsymbol{\sigma}| = |E|^{\sigma}$.

【例3.1 】

a，b，$\sigma \in \Gamma$とする時，$E = (a\sigma b + a b)\boldsymbol{\sigma}$ は文脈自由表現である。また，$|E| = |(a\sigma b + a b)|^{\sigma} = \{a\sigma b,\ a b\}^{\sigma} = \{a^i\ b^i \mid i \geq 1\}$ である。

【定義3.4 】(言語族$\mathcal{E}_T$)

Tを（一般に無限の）アルファベットとする時，$\mathcal{E}_T$ を以下の条件を満足する最小の言語族として定義する：

(1) Φ，$\{\varepsilon\} \in \mathcal{E}_T$，

(2) もし$a \in T$ならば，$\{a\} \in \mathcal{E}_T$ である，

(3) もしL_1 ，$L_2 \in \mathcal{E}_T$ かつ$\sigma \in T$ならば，$L_1 \cup L_2$ ，$L_1\ L_2$ ，$L_1{}^{\sigma} \in \mathcal{E}_T$ である。

さて，次の結果は，ここでの議論にとって重要な位置を占める。

【定理3.1 】([Gr71] の系2.8)

Σを有限アルファベットとし，LをΣ上の言語とする。この時，Lが文脈自由言語である必要十分条件は，Σ⊂Tなるある有限アルファベットTに対して$L \in \mathcal{E}_T$ となる事である。

従って，我々は文脈自由言語とその表現に関して，以下の特徴付けを得る。

【定理3.2 】

Σを有限アルファベットとし，LをΣ上の言語とする。この時，Lが文脈自由言語である必要十分条件は，Σ⊂Tなるある有限アルファベットTとその上の文脈自由表現Eが存在して，|E|=Lとなる事である。

（証明）定理3.1 と諸定義とから明らかである。

この様に，文脈自由表現は，文脈自由言語を定義する一つの表現を提供し，我々の目的，即ち文脈自由言語の帰納的推論にとって，文法等の他の表現方法に比較して，より好ましい表現空間を与えてくれる。即ち，次節で示されるように，文脈自由表現による文脈自由言語の推論アルゴリズムは，正則表現による正則言語の推論アルゴリズムを自然な形で拡張することにより得られる。

3.2　帰納的推論アルゴリズム

さて，本論文では，文脈自由言語族の推論問題を以下のように定式化する。

《文脈自由言語の帰納的推論問題》

① 意味空間Dは文脈自由言語の全体からなる族，D上の半順序は包含関係$\supseteq$，

② doは与えられた文脈自由言語，

③ 表現空間$\mathcal{E}$は文脈自由表現のクラス，

④ 全射hは$E \in \mathcal{E}$に対して$|E| \in D$を対応させる，

⑤ オラクルEXは EX( )= +eならば　e $\in$do，EX( )=-eならばe $\notin$ do を意味する。

言語Lを与えられた有限アルファベットΣ上の文脈自由言語とする。我々は，Lを具体例の提示により帰納的に推論する問題を考察する。
文脈自由表現のクラスを構成する無限アルファベットΓを固定して考える，ここに$\Sigma \subset \Gamma$である。（補助的な記号＋，括弧（，）はΓに含まれないとする。）

さて，$\mathcal{E}$上の演算τを以下の様に定義しよう：

E，E_1，E_2をΓ上の文脈自由表現とする。記法として，$E_1 \rightarrow E_2$と書いて関係$E_2 \in \tau(E_1)$を表す。

(1). $\phi \rightarrow \phi\,\phi$

(2). $\phi \rightarrow a \quad (\forall a \in \Gamma \cup \{\varepsilon\})$

(3). $\phi \rightarrow (\phi)\,\boldsymbol{\sigma} \quad (\forall \sigma \in \Gamma)$

(4). $\phi \rightarrow (\phi + \phi)$

(5). もし$E_1 \rightarrow E$ならば$E_1 + E_2 \rightarrow E + E_2$，かつ$E_2 + E_1 \rightarrow E_2 + E$

(6) もし$E_1 \rightarrow E$ならば$E_1\,\boldsymbol{\sigma} \rightarrow E\,\boldsymbol{\sigma} \quad (\forall \sigma \in \Gamma)$

(7). もし$E_1 \rightarrow E$ならば$E_1\,E_2 \rightarrow E\,E_2$，かつ$E_2\,E_1 \rightarrow E_2\,E$

ここに，各σに対して$\boldsymbol{\sigma}$は新しい記号としてのボールドフェイスである。

τを構成する変換規則には二つのタイプがあることに注意しよう。ひとつはϕを書き換える規則（(1)〜(4)），もう一つは表現構造を保存する規則（(5)〜(7)）である。

【補題3.1 】

上記のように定義された演算τは以下の性質をもつ：

（i）τはある特殊な表現ϕに対して完全である，即ち，ϕからτを有限回適用して得られるすべての表現の集合$\tau^*(\phi)$はすべてのΣ上の文脈自由言語に対して少なくとも一つの文脈自由表現を含む。

（ii）任意のE_1，$E_2 \in \mathcal{E}$に対して，もし$E_1 \in \tau(E_2)$ならば$|E_1| \supseteq |E_2|$である。

〔証明〕

（ｉ）定義3.2 の構成法に対応して帰納法による：

先ず(2)により $a \in \tau(\phi)$ $(\forall a \in \Gamma \cup \{\varepsilon\})$ である。次に，いま $E_1 \in \tau^*(\phi)$ かつ $E_2 \in \tau^*(\phi)$ と仮定する。（以下において，記法 $\rightarrow^*$ によって τ のゼロ回以上の適用を表す。）この時，

$\phi \rightarrow \phi + \phi$ ((4)による) $\rightarrow^* E_1 + \phi$ ((5)の繰り返しによる) $\rightarrow^* E_1 + E_2$ ((5)の繰り返しによる)

よって $E_1 + E_2 \in \tau^*(\phi)$ を得る。同様に，

$\phi \rightarrow \phi\,\phi$ ((1)による) $\rightarrow^* E_1\,\phi$ ((7)の繰り返しによる) $\rightarrow^* E_1\,E_2$ ((7)の繰り返しによる)

よって $E_1\,E_2 \in \tau^*(\phi)$ を得る。 さらに(3)により

$\phi \rightarrow \phi\,\sigma$ であり，$\phi \rightarrow^* E_1$ であるから，(6)の繰り返しにより

$\phi\,\sigma \rightarrow^* E_1\,\sigma$ 従って，$\phi \rightarrow^* E_1\,\sigma$

即ち，$E_1\,\sigma \in \tau(\phi)$ である。

（ii）規則(1)から(4)により $\tau(\phi) = \{\phi\,\phi,\ \varepsilon,\ \phi + \phi,\ \sigma,\ \phi\,\sigma\ (\forall \sigma \in \Gamma)\}$ であり，すべての $E \in \tau(\phi)$ に対して $|E| \supseteq |\phi| = \Phi$ が成り立つ。

さて，$E' \in \tau(E)$ とする。この時，Ｅは以下の三つの場合に限られる。（$E \in \Gamma \cup \{\varepsilon\}$ の時は，τ は適用可能でない事に注意。）

①$E = E_1 + E_2$ の時，帰納法の仮定から $E_i' \in \tau(E_i)$ $(i = 1, 2)$ とすると $|E_i| \subseteq |E_i'|$ である。よって τ が(5)の時，

$|E| = |E_1 + E_2| = |E_1| \cup |E_2| \subseteq |E_1'| \cup |E_2| = |E'|$

或いは

$|E| = |E_1 + E_2| = |E_1| \cup |E_2| \subseteq |E_1| \cup |E_2'| = |E'|$

となる。

②$E = E_1\,E_2$ の時，同様の仮定のもとに，τ が(7)の時，

$|E| = |E_1\,E_2| = |E_1|\,|E_2| \subseteq |E_1'|\,|E_2| = |E'|$

或いは

$|E| = |E_1\,E_2| = |E_1|\,|E_2| \subseteq |E_1|\,|E_2'| = |E'|$

となる。

③さらに，$E = E_1\,\sigma$（ある $\sigma \in \Gamma$ に対して）の時，帰納法の仮定により $E_1' \in \tau(E_1)$ とすると，$|E_1| \subseteq |E_1'|$ である。よって，τ が(6)の時，

$|E| = |E_1|^{\sigma} \subseteq |E_1'|^{\sigma} = |E'|$

が成り立つ。以上で証明された。 □

【例３．２】（表現導出過程）

以下のϕからの導出を考える：

	使われた主な規則
$\phi \rightarrow (\phi)\ \tau$	(3)
$\rightarrow (\phi\ \phi)\ \tau$	(1)
$\rightarrow ((\phi + \phi)\ \phi)\ \tau$	(5)
$\rightarrow ((\phi\ \phi + \phi)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ \phi + \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (\phi)\ \sigma + \phi)\ \phi)\ \tau$	(3)
$\rightarrow ((a\ (\phi + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(4)
$\rightarrow ((a\ (\phi\ \phi + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ (\phi\ \phi\ \phi + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ (a\ \phi\ \phi + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \phi + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + \phi)\ \sigma + \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + \phi\ \phi)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ \phi)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (\phi)\ \nu)\ \phi)\ \tau$	(3)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (\phi + \phi)\ \nu)\ \phi)\ \tau$	(4)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (\phi\ \phi + \phi)\ \nu)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (\phi\ \phi\ \phi + \phi)\ \nu)\ \phi)\ \tau$	(1)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \phi\ \phi + \phi)\ \nu)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \phi + \phi)\ \nu)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + \phi)\ \nu)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + a)\ \nu)\ \phi)\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + a)\ \nu)\ (\phi + \phi))\ \tau$	(4)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + a)\ \nu)\ (\tau + \phi))\ \tau$	(2)
$\rightarrow ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + a)\ \nu)\ (\tau + \varepsilon))\ \tau$	(2)

この様に，最終的に導出される表現Ｅは

$E = ((a\ (a\ \sigma\ \sigma + b)\ \sigma + b\ (b\ \nu\ \nu + a)\ \nu)\ (\tau + \varepsilon))\ \tau$

であり，これは言語

$$|E| = \{w \mid \#a\ (w) = \#b\ (w),\ w \in \{a, b\}^*\}$$

ここに$\#x\ (w)$は語wに含まれている文字xの個数，

を表している。（なお，この言語はＳを初期記号として，規則：

$S \rightarrow aBS$，$S \rightarrow aB$，$S \rightarrow bAS$，$S \rightarrow bA$，

$A \rightarrow bAA$，$A \rightarrow a$，$B \rightarrow aBB$，$B \rightarrow b$

を持つ文法によって生成される。）

さて，上記の演算γを利用して，我々は以下で示される様な，非常に単純な文脈自由言語に対する推論アルゴリズムを得る。アルゴリズムは“枚挙による同定”アルゴリズムになっている。

具体例の集合に関して我々は正・負の例を提供するオラクルEXを仮定する。正の具体例は，文脈自由表現Eでその言語｜E｜がただ一つの語からなるようなものと見做す事が出来るから，正の例全体は表現集合$\mathcal{E}$の部分集合を形成していると考える事ができる。

【定義3.5 】

（i）（完全かつ十分なオラクル）

doを目標の文脈自由言語とする。次の条件を満たす集合Kが存在する時，オラクルEXをdoの完全かつ十分なオラクルと呼ぶ：

(1) $\mathcal{E}$の符号付けされた部分集合Kでh(E)=do なるすべてのE($\in\mathcal{E}$) はKの元だけの提示によって決定される(即ち，集合 {E $\in\mathcal{E}$ ｜ Kのすべての正の例$\in$h(E)かつKの全ての負の例$\notin$h(E)} が集合 { E$\in\mathcal{E}$ ｜h(E)=do } と一致する。)

(2) e$\in$doなるすべての e$\in$Kに対して少なくとも一度EX( ) は究極的に+eを返し，そうでないすべてのe' $\in$Kに対しては少なくとも一度EX( ) は究極的に-e' をかえす。

（ii）（容認される提示）

完全かつ十分なオラクルをもつdoの例集合をdoの容認される例提示と呼ばれる。

〔注記〕

Kとして {E$\in\mathcal{E}$｜　｜E｜は要素一つ，かつもし｜E｜$\in$doならば＋E，さもなければ－Eにする} なる集合を考えると，明らかにKはdoの完全かつ十分なオラクルEXを提供する。

アルゴリズムを提示する前に幾つかの準備が必要である。

与えられた目標の文脈自由言語Lに対して，$\Sigma=\{a_1, \ldots, a_n\}$ をそのアルファベットとする。さらに，整数k$\geq$1に対して，$\Gamma_k=\Sigma\cup\Delta_k\cup\Delta_k'$，ここに$\Delta_k=\{\sigma_1, \sigma_2, \cdots, \sigma_k\}$，$\Delta_k'=\{\sigma_1', \sigma_2', \cdots, \sigma_k'\}$，とする。

定理3.2 によれば，『任意の文脈自由言語Lに対して，ある整数k$\geq$1と表現Eが存在して｜E｜＝LかつEはΓ_k の元だけからなる。』と言う事が言える。（一般に，kはLに依存する事に留意。）

与えられた表現Eとi$\geq$0，k$\geq$1なる整数i，kに対して，

$\gamma(E,i,k)=\{E' \mid E \rightarrow^{i} E'$かつ E' はΓ_k の元だけから成る$\}$ とする。この時，

$\gamma^*(E)=\cup_{i=0}^{\infty}\cup_{k=1}^{\infty}\gamma(E,i,k)$　である。

〔アルゴリズム概要〕

アルゴリズムの詳細をあたえる前に，その概略を述べる。先ず，$\gamma(\phi, i, k)$を$\gamma(i,k)$と略記する。アルゴリズムはϕから始めてすべての表現を枚挙して行く手続きが必要である。それは以下の様に行われる：

$$\phi \rightarrow \tau(1,1) \Rightarrow \tau(2,1) \Rightarrow \tau(3,1) \Rightarrow \tau(4,1) \Rightarrow \cdots$$

$$\phi \rightarrow \tau(1,2) \Rightarrow \tau(2,2) \Rightarrow \tau(3,2) \Rightarrow \cdots$$

$$\phi \rightarrow \tau(1,3) \Rightarrow \tau(2,3) \Rightarrow \cdots$$

$$\phi \rightarrow \tau(1,4) \Rightarrow \cdots$$

$$\vdots$$

記法として，$\tau(i,k)$に属する表現E を［E,(i,k)］と表す。任意のi,k に対して $\tau(i,k)$ は有限集合である事に注意。　一般に$\tau(i,k) \Rightarrow \tau(i+1,k)$ は，

$$\tau(i+1,k) = \{E' \mid E \rightarrow E' \text{ かつ} E \in \tau(i,k) \text{ かつ} E' \in \Gamma_k^*\}$$

を意味する。即ち，τ（i＋1，k）はτ（i，k）にτ_k を一回適用する事によって得られる。ここに，τ_k は規則(2)，(3)，(6)を$\Gamma = \Gamma_k$ に制限した演算である。また，τ（1，k）はϕにτ_k を一回適用して得られる。

さて，アルゴリズムは，ϕからτを適用して得られる推測の表現を［E，（i，k）］の形でキューQを用いて枚挙していく。そして，次々とそれらの推測を具体例により，一般化して行く。その順序は，上述した図において，

$\tau(1,1)$，$\tau(2,1)$，$\tau(1,2)$，$\tau(3,1)$，$\tau(2,2)$，$\tau(1,3)$，．．．

である。アルゴリズムは推測の表現Eに対して，正の例を含まなければ（"特殊化過ぎる"），Eを一般化するが，負の例を含んでしまう（"一般化過ぎる"）時は，単にこれを捨てる。この繰り返しは，推論すべき言語を正しく表わしている表現が見つかるまで続けられる。

アルゴリズムＡ１《　文脈自由言語の推論アルゴリズム　》

〔入力〕　帰納的可算な文脈自由表現の集合
　　精密化演算γ
　　容認される目標Ｌの例提示
〔出力〕　最初のｎ個の具体例に対して正しい様な仮説E_nの系列
　　E_1，E_2，…
〔手続き〕（初期化）
　　Q←γ（1，1）；
　　EXAMPLES←Φ（空集合）；｛EXAMPLESは具体例をストアする｝
　　X ← next(Q)；｛Qの先頭の要素をXに送る｝
　　do （永久に）:
　　次の具体例を呼びEXAMPLESへ追加;
　　（γ（i，k）＝｛[E_j,(i,k)] | j=1,...,p｝とする）
　　while X = [E_j,(i,k)] とする時，　∃e ∈EXAMPLES s.t.
　　E_j はe に対して正しくない
　　if E_j が"特殊化"過ぎる
　　then do
　　if i≧2
　　then if k=1 かつj=1
　　then γ(1,i) γ(E_1,1,1) をQの末尾へ追加;
　　X← next(Q)；
　　else γ(E_j,1,k) をQの末尾へ追加;
　　X←next(Q);
　　else γ(E_j,1,k) をQの末尾へ追加；
　　X←next(Q);
　　else （E_jが"一般化"過ぎる）
　　E_jを破棄；
　　X←next(Q);
　　E_jを現在の推測として出力;

ここに
　　Eが"特殊化"過ぎる：＝
　　if ∃+e∈EXAMPLES s.t. e ∉ | E |, then trueを返す
　　else falseを返す
　　Eが"一般化"過ぎる：＝
　　if ∃-e∈EXAMPLES s.t. e ∈ | E |, then trueを返す
　　else falseを返す

【注記】

① 与えられた任意の（i，k）に対して，τ（i，k）は明らかに計算可能である。

② 任意のEと任意の具体例wに対して，$w \in |E|$か否かを決定するアルゴリズムは存在する。（文脈自由言語の所属問題は決定可能である。）

アルゴリズムA1は演算τによるϕからの数え挙げを基本にする枚挙による同定アルゴリズムになっている。

【定理3.3 】

任意の文脈自由言語Lと任意の容認されるLの提示に対して，アルゴリズムA1は極限においてLを同定する。

（証明） 我々は，二つの事を示す必要がある：即ち，アルゴリズムA1はある推測Eに収束する事，そしてその表現EはLの正しい推測である事，である。

さて，定理3.2 とτの完全性の性質（補題3.1 の(i)）によりϕから導出可能なEで$|E| = L$なるものが存在する。即ち，$\phi = E_0 \rightarrow \cdots \rightarrow E_n = E$かつ$|E| = L$である。ここでnを可能な最小の数とする。さらに，もう一つの性質（補題3.2 の(ii)）とnの最小性により，$|E_i| \subset |E_n|$であるとして良い。この時，あるw_iで$w_i \in L$かつ$w_i \notin |E_i|$がある。また，$E_n \in \tau(n, k)$とする。

アルゴリズムA1がある表現Eに収束すると仮定しよう：即ち，あるi,k ≥ 0 とある$E \in \tau$（i，k）が存在して，EXAMPLESの全ての元wに対してEは正しく，かつそれ以後のすべての具体例に対してEは正しい。（それが収束の意味である。）従って，明らかにEはLを正しく表現している。

アルゴリズムA1が発散すると仮定する。上記の一般化チェーンの長さjに関する帰納法によって『一般化のチェーンにおける各E_j（$0 \leq j \leq n$）はQに現れ，そして一般化される』という事を示す。j＝0の時は明かである。今，j＝iの時，E_iはτ（i，p）の元としてQに現れ，一般化されると仮定する。$E_{i+1} \in \tau(E_i)$であるから，あるqがあって$E_{i+1} \in \tau(i+1, q)$となっている。 アルゴリズムは発散するから，E_iからE_{i+1}までの間の表現はQにおいて調べられ，ある反例によって一般化されるか，或いは捨てられる。結局，E_{i+1}は何時かはQの先頭に現れ，w_{i+1}によって一般化される。このようにE_n（j＝nの時）はいずれQの先頭に現れるが，アルゴリズムは発散するので，あるw_0に対してE_nは正しくない。これは，$|E_n| = L$に矛盾する。よってアルゴリズムは収束する。

この様に，A1は正しい推測の表現E_nを必ず枚挙し，かつそれに収束する。 □

4．　部分クラスの推論

前節では，推論対象の意味空間として文脈自由言語族全体を考えたが，推論アルゴリズムの効率は明らかに良くない。有限オートマトンに対する最小状態推論問題がNP- 完全な問題である事（［Go78］）から推測して，効率的なアルゴリズムを発見するのは非常に難しいと思われる。

ここでは，意味空間を文脈自由言語族のあるサブクラスに制限し，そのクラスに対する推論問題を考察する。

4．1 準線型文脈自由言語と推論アルゴリズム

【定義4.1 】（ 準線型言語 semilinear language［Gr71］）

（ｉ）文脈自由文法Ｇ＝（Ｎ，Ｔ，Ｐ，Ｓ）が準線型であるとは，「全てのＡ∈Ｎと全てのｘ∈（Ｎ∪Ｔ）＊に対して，Ａ⇒＊ ｘならばｘには高々一つのＡしか現れない」という性質を満たす時を言う。

（ii）言語Ｌが準線型（文脈自由）言語であるとは，ある準線型の文脈自由文法ＧがあってＬ＝Ｌ(Ｇ）となる時を言う。

〔注記〕

① 準線型文法（ 言語）と等価な用語としてfinite-index grammar and language ［Sa73］non-expansive grammar，derivation-bounded language ［GS68］，superlinear grammar ［Br68］等々が知られている。

② "semilinear"は通常"半線型"とも訳せるが，この用語は別の意味ですでに定着しているので，ここでは"準線型"とした。

Ｔを有限終端アルファベットとし，$\Gamma_2 = T \cup \{\sigma_1, \sigma_2, \bar{\sigma}_1, \bar{\sigma}_2\}$ とする。

【補題4.1 】（［Gr71］）

任意の準線型言語は$\mathcal{E}_{\Gamma_2}$に含まれる。即ち，任意の準線型言語Ｌに対して，Γ_2 の元だけから成るある文脈自由表現ＥでＬ＝｜Ｅ｜なるものが存在する。

この性質は，準線型言語の特徴付けに関して，他の（ 文法或いは抽象機械等の）表現方法と文脈自由表現による方法との顕著な相違を表している。即ち，準線型言語の文法による表現においては，必要な非終端記号の個数に関する上限は存在しない。それは，「任意の自然数ｎ対してある線型言語L_nでｎ個より少ない非終端記号しか持たない文脈自由文法によっては生成できないものが存在する」という結果で知られている。

補題4.1 は以下で示すように，準線型言語の帰納的推論問題に対して好都合な性質を提供している。

【補題4.2 】

EXP $_2$ をΓ_2 の元からなる文脈自由表現の集合とする。即ち，EXP $_2$ ＝ {E | $\phi \rightarrow^*$ E かつEはΓ_2 の元からなる} とする。 この時，ある文脈自由文法Gがあって EXP $_2$ ＝ L（G）となる。

証明.

次の様な文脈自由文法Gを考えよう：G＝（{A}，Σ ，P，A），ここに Σ＝Γ_2 ∪ {（，），+}，Pは以下の(1)～(4)からなる：

(1) A→AA

(2) A→a （∀ a ∈T∪ {σ_1 ， σ_2 ， ε， ϕ} ）

(3) A→（A）σ_i （i＝1，2）

(4) A→（A＋A）

Eを任意のEXP $_2$ の元とすると，補題3.1 で示したのと同様に表現の長さに関する帰納法を用いて，A⇒* EかつE ∈Σ* である事が言える。逆に，A⇒^{n} xかつx ∈Σ* とする。導出過程の長さnの帰納法によって，『xはEXP $_2$ に属する』事を言う。n＝1の時，は明らかである。n＝k以下の時成り立つと仮定して，

$$A \Rightarrow^* w_1\ A w_2 \Rightarrow w_1\ w w_2 = x$$

とする，ここにw，w_1 ，w_2 ∈Σ* である。これを

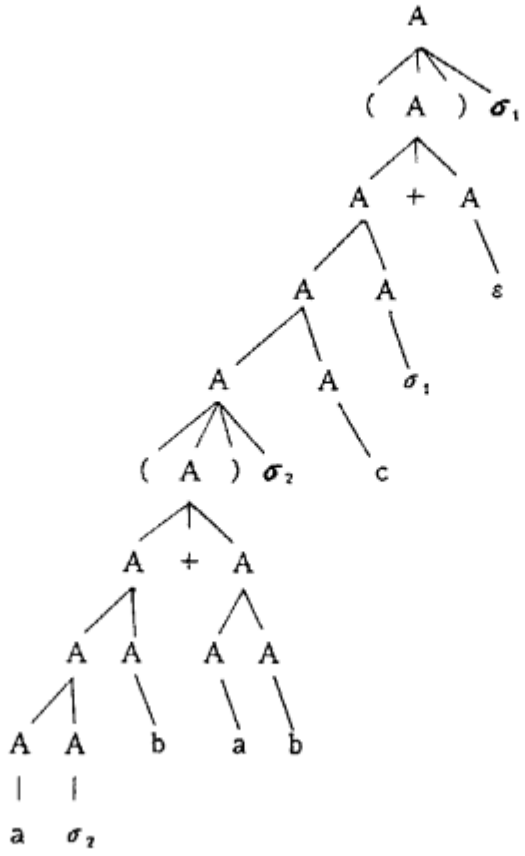
$$A \Rightarrow y \Rightarrow^*\ x$$

と考え直すと，第一ステップA→yは(1)，(3)，(4)のいずれかである。y＝AAとすると，あるi，jが存在して$A \Rightarrow^i\ x_1$ かつ$A \Rightarrow^j\ x_2$ かつx＝x_1 x_2 かつi＋j＝kである。帰納法の仮定により，x_1 ，x_2 ∈EXP $_2$ であるから x ∈Exp $_2$ となる。y＝（A）σ_i 及びy＝（A＋A）の時も同様にしてx ∈EXP $_2$ が言える。 □

この様に，準線型言語の帰納的推論における表現の枚挙手続きは，ある特定の文脈自由文法を生成装置として実現できる。

【例4．1】（表現の導出木）

言語L＝（{a^i b^i | i ≧1} c）* を考えよう。Lは準線型言語である事が知られている。L＝| E |なる表現Eは，上記の文脈自由文法Gによって以下の様に導出される：



結果として導出される表現Eは：

$$E = ((a\,\sigma_2\ b + a\ b)\ \sigma_2\ c\,\sigma_1\ + \varepsilon)\ \sigma_1$$

となる。□

さて，EXP_2 上の演算 τ_2 を以下の様に定義する：

(1) $\phi \rightarrow \phi\ \phi \mid (\phi + \phi) \mid (\phi)\ \sigma_1 \mid (\phi)\ \sigma_2 \mid a\ (a \in T \cup \{\sigma_1, \sigma_2, \varepsilon\})$

(2) もし $E_1 \rightarrow E$ ならば $E_1 + E_2 \rightarrow E + E_2$，かつ $E_2 + E_1 \rightarrow E_2 + E$

(3) もし $E_1 \rightarrow E$ ならば $E_1\ \sigma_1 \rightarrow E\ \sigma_1$ かつ $E_1\ \sigma_2 \rightarrow E\ \sigma_2$

(4) もし $E_1 \rightarrow E$ ならば $E_1\ E_2 \rightarrow E\ E_2$，かつ $E_2\ E_1 \rightarrow E_2\ E$

【補題4.3】

上記のように定義された演算 τ_2 は以下の性質をもつ：

（i）τ_2 は表現 ϕ に対して完全である，即ち，ϕ から τ_2 を有限回適用して得られるすべての表現の集合 $\tau_2{}^*(\phi)$ はすべての準線型文脈自由言語に対して少なくとも一つの文脈自由表現を含む。

（ii）任意の E_1，$E_2 \in EXP_2$ に対して，もし $E_1 \in \tau_2(E_2)$ ならば $|E_1| \supseteq |E_2|$ である。

（証明）補題4.1，4.2及び定義とから明らか。

準線型文脈自由言語の推論アルゴリズムは，一般の文脈自由言語の特殊な場合として，以下に示す様により簡潔なものとなる。

アルゴリズムＡ２《 準線型文脈自由言語の推論アルゴリズム 》

〔入力〕　帰納的可算な文脈自由表現のクラスEXP，
精密化演算子r_2，
容認される目標Ｌの例示
〔出力〕　最初のｎ個の具体例に対して正しい様な仮説E_nの系列E_1，E_2，…
〔手続き〕　(初期化)
Q←空のキュー　{Qはこれから可能な仮説を保持するキュー}
E←φ；　{Eは現時点での仮説を保持；φは最も特殊な表現}
EXAMPLES←Φ（空集合）；　{EXAMPLESは具体例をストアする}
do （永久に):
次の具体例を呼びそれをEXAMPLESへ追加する；
repeat
if　Eが"特殊化"過ぎる
then　Qの末尾にr_2（Ｅ）を付け加える
E← next(Q)　{Qの先頭を取る；Qが空の時失敗する}
else　if　Eが"一般化"過ぎる
E← next(Q)
until　EがEXAMPLESにおけるすべての語に対して正しい
Eを現在の推測として出力

ここに
「Eが"特殊化"過ぎる」及び「Eが"一般化"過ぎる」の定義は
アルゴリズムＡ１におけるものと同じ

【定理4.1 】
任意の準線型文脈自由言語Ｌと任意の容認されるＬの提示に対して，アルゴリズムＡ２は極限においてＬを同定する。
(証明) 定理3.3 と同様。(略)

4．2　推論問題とその複雑さ

本節では，以下のような型の推論問題を考察する：「与えられたサンプル集合Ｓに対して，Ｓと一致するような最小インデックスを持つ文脈自由表現Ｅを見つけよ」。さらに，この問題の変形として，「与えられたサンプル集合Ｓと正数ｔに対して，高々サイズｔの文脈自由表現ＥでＳと一致するようなものが存在するか否か？」という問題が考察される。

【定義4．2】

（1）文脈自由表現Ｅのサイズとは，Ｅに現れるアルファベット文字の個数を言う。

（2）Σ上のサンプル集合Ｓとは対（u_i，a_i）（$u_i \in \Sigma^+$，$a_i \in \{+, -\}$）から成る有限集合であり，もし（u_i，a_i），（u_j，a_j）∈Ｓかつ$u_i = u_j$ならば$a_i = a_j$を満たすものを言う。

（3）文脈自由表現Ｅがサンプル集合Ｓに一致するとは，任意の（ｕ，ａ）∈Ｓに対して$u \in |E|$なる事とａ＝＋である事が同値である時を言う。

【定義4．3】

ＳをΣ上のサンプル集合，ｔを正の整数とする。また，$\mathcal{L}$を文脈自由言語の部分族とし，$\mathcal{E}$を$\mathcal{L}$を生成するのに必要な文脈自由表現の最小の部分集合とする。さらに，ある固定された$\mathcal{E}$の枚挙手続きが与えられているとする。この時，

（1）$\mathcal{L}$に対する最小推論問題とは，その枚挙手続きにおける最小インデックスを持つＥでＳと一致するものを見つける問題を言う。また，

（2）$\mathcal{L}$に対する推論決定問題とは，高々サイズｔの$\mathcal{E}$の元ＥでＳと一致する様なものが存在するか否か？を決定する事を言う。

【定理4．2】

準線型文脈自由言語族に対する最小推論問題はＮＰ－困難である。

証明．　[An78] によると，「正則表現による正則言語族に対する最小推論問題はＮＰ－困難である」事が知られている。また，任意の正則表現Ｅに対して，$|E| = |E'|$なる文脈自由表現Ｅ’で EXP_2 に属するものは容易に構成可能である。実際，$E = (E_1)^*$（或いは$= (E_1)^+$）なる場合は$E' = (E_1 \sigma_1 + \varepsilon)\sigma_1$（或いは$= (E_1 \sigma_1 + E_1)\sigma_1$）とし，$E = E_1 + E_2$ 或いは$= E_1 E_2$ なる場合はＥ’＝Ｅでよい。これを再帰的にE_i に対して繰り返し適用して行く。その際，σ_1 とσ_2 とを交互に使用する。結果として得られる表現Ｅ’はEXP_2 に属する事及び$|E| = |E'|$なる事は殆ど明らかである。これに要する手間は表現Ｅのサイズをｎとすると０（ｎ）で十分である。従って，正則表現に対する問題はこの問題に多項式時間還元可能であり，この問題はＮＰ－困難である。□

【系4.1】
文脈自由言語族に対する最小推論問題はＮＰ－困難である。

この様に，例えば，前述のアルゴリズムＡ２における枚挙手続きを仮定した時の，サンプル集合Ｓに関する最小推論問題はＮＰ－困難である事が分かる。

【定理4.3】
準線型文脈自由言語族に対する推論決定問題はＮＰ－完全である。

証明．以下の二つの事を示せば良い。即ち，集合Ｒを
Ｒ＝｛（Ｓ，ｔ）｜高々サイズｔのEXP$_2$に属する文脈自由表現でＳと一致するものが存在する｝
と定義する時，（ｉ）ＲはＮＰに属する，（ii）ＲはＮＰ－困難である，の二つである。
（ｉ）の証明：補題4.2 より，EXP$_2$ の元は文脈自由文法Ｇによって生成される。Ｇの生成規則の右辺は少なくとも二個　　のアルファベット文字を生成するから，サイズｔの表現Ｅに対するＧによる導出木の深さは高々０（ｔ）である。よって，我々はＧによる導出を非決定的に推測しながら高々０（ｔ）の手間で表現Ｅを求めることが出来る。さらに，｜Ｓ｜＝ｍかつｎ＝ｍａｘ｛lg（ｕ）｜（ｕ，ａ）∈Ｓ｝　とすると，Ｓの各元（ｕ，ａ）に対してｕ∈｜Ｅ｜の時かつその時のみａ＝＋である事を調べるのに高々０（ｍｎ^3）必要とする。（文脈自由文法Ｇに対する所属問題は０（ｎ^3）で解ける事に留意せよ。［HU69］）よってＲはＮＰに属する。
（ii）の証明：［An78］によると，正則表現による正則言語族に対する推論決定問題はＮＰ－困難である事が知られている。また，前定理の証明で示された様に，任意の正則表現Ｅに対して，｜Ｅ｜＝｜Ｅ'｜なる文脈自由表現Ｅ'で EXP$_2$ に属するものが多項式時間で容易に構成可能である。従って，ＲはＮＰ－困難である。□

5. 結論

文脈自由言語族に対する帰納的推論問題を，文脈自由表現を用いて考察した。ここでのアプローチは，従来の文法推論による方法と比べて，非常に簡潔な推論アルゴリズムを提供しうる，と言う点で特色がある。この方法の主な利点は，正則言語の正則表現によるアプローチを自然な形で拡張できる可能性を秘めていると言う事を持って代表される。

一般に，帰納的推論アルゴリズムには二つの型が存在する：枚挙的手法と構成的手法である。構成的手法に比べて，枚挙的手法は全域探索を基本にしているための完全性及びある意味での最適解を保証するという利点がある。反面，後者は前者に比して効率上実用的なアルゴリズムには成り難い，という欠点を持ち合わせている。

ところで，文脈自由言語族の帰納的推論を扱った文献としては以下の様なものが知られている。[KK76] においては，プログラミング言語の帰納的推論という視点から，文脈自由文法（より正確には BNF表現）のインターラクティブな構成的推論方法が論じられている。また，[Wh77] は，文脈自由文法を数え上げる方法を論じ，幾つかの改良アルゴリズムを提示している。前者は構成的手法に基づいているため完全性等の理論的背景が明確でなく，また後者は枚挙手法によるアドホックな効率改良に止まっており，実用的なアルゴリズムには程遠い。[Ta86] では，線型文法に対するアルゴリズムを拡張する方法で文脈自由文法を構成する手法を提案している。この方法は，探索空間を有限に制約するため，幾つかのパラメタに上限を与える事を前提としていて，構成的でかつ完全性を保証しているが，前提の上限を与える基準については論じていない。

我々の文脈自由表現によるアプローチは，枚挙的手法に基づいているため，上記の利点は保持している。しかしながら，効率的なアルゴリズムの提示という所には未だ到っていない。完全性等の理論的結果を維持しかつ如何に効率の問題を解決していくかは，今後の課題である。

6. 謝辞

日頃御指導，御鞭撻を頂く北川敏男会長に感謝致します。また，帰納的推論に関する文献の提供をはじめ，議論して頂いた榎本肇所長に深謝致します。

また，グループメンバーである高田裕志氏，榊原康文氏，石坂裕毅氏との討論は非常に有益であった事は付記するまでもありません。

なお本研究の一部は，第５世代コンピュータ・プロジェクトの一環としてICOTの委託で行ったものである。

7. 参考文献

[An78] Angluin,D., On the Complexity of Minimum Inference of Regular Sets, Inf. and Contr. 39, 337-350 (1978).

[Br68] Brzozowski,Y.A., Regular-like Expressions for Some Irregular Languages, IEEE Conf. record of 9th Ann. Sym. on Switching and Automata Theory 278-280, 1968.

[Go67] Gold,E.M., Language Identification in the Limit, Inf. and Contr. 10,447-474 (1967).

[Go78] Gold,E.M., Comlexity of Automaton Identification from Given Data, Inf. and Contr. 37, 302-320 (1978).

[Gr71] Gruska,J., A Characterization of Context-free Languages, J. of Comput. and Sys. Sci. 5, 353-364 (1971).

[GS68] Ginsburg,S. and Spanier,E.H., Derivation-Bounded Languages, J.of Comput. and Sys. Sci. 2, 228-250 (1968).

[HU69] Hopcroft,J.E. and Ullman, J.D., Formal Languages and Their Relation to Automata, Addison-Wesley, 1969.

[Is86] Ishizaka,H., Model Inference Incorporating Generalization, Proc. of Sym. on Software Sci. and Engineering, Kyoto, Sept. 1986.

[KK77] Knobe,B. and Knobe,K., A Method for Inferring Context-free Grammars,Inf. and Contr. 31, 129-146 (1976).

[La86] Laird,P.D., Inductive Inference by Refinement, Tech.Rep., TR-376, Dept. of Comp. Sci., Yale University, May 1986.

[Sa73] Salomaa,A., Formal Languages, Academic Press, 1973.

[Sh81] Shapiro,E., Inductive Inference of Theories from Facts, Tech.Rep. 192, Dept. of Comput. Sci., Yale University, 1981.

[Sh82] Shapiro,E., Algorithmic Program Debugging, Ph.D dissertation, Dept. of Comput. Sci., Yale University, 1982, also published by MIT Press, 1983.

[TA77] Tanatsugu,K. and Arikawa,S.,On Characteristic Sets and Degrees of Finite Automata, Int. J. of Comput. and Inf. Sci. 6, no.1, 83-93 (1977).

[Wh77] Wharton,R.M., Grammar Enumeration and Inference, Inf.and Contr. 33, 253-272 (1977).

[榎本他76] 榎本, 富田: 決定性有限オートマトンの代表記号列集合, 電子通信学会論文誌, vol.59-D, No.9, 660-667 (1976).

[棚次86] 棚次: Self-embeddingを利用した文脈自由言語に対する文法推論, LAシンポジウム予稿, 京大数解研, 1986年, 2月.