ICOT Technical Memorandum: TM-0712

TM-0712

機械学習
—言語の帰納的推論—

高田裕志，榊原康文，
横森　貴（富士通　国際研）

March, 1989

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

「機械学習 — 言語の帰納的推論」

高田裕志　榊原康文　横森貴

国際情報社会科学研究所
富士通株式会社

要旨

本報告書は，“機械学習”における最も基本的かつ重要な研究課題のひとつである“帰納的推論”という知的推論メカニズムに注目し，言語の帰納的な学習，特に“形式言語の帰納的推論”に関して行なった筆者らの最近の研究成果をまとめ上げたものである．

一般的に，“帰納的推論”とは，「有限個の具体例からそれらを規定している規則(法則)を見つける」ことであり，機械による学習機能の実現を考えるうえで，不可欠な研究課題である．

本稿では，“言語”として“形式言語”を考察する．形式言語の理論は，過去三十有余年の研究成果の歴史的蓄積があり，またその数学的な性質もかなり解明されているため，開発された推論アルゴリズムを計算機上で実装するのにも適している等の利点がある．

形式言語の帰納的推論問題は，従来“文法推論の問題”として論じられてきたが，正則言語等に関する限られた成果を除いては，その計算量的な壁がその後の展開を阻んでいた．

本稿では，与えられる具体例に加えて，対象の言語に関する種々の情報量を仮定する問題設定において，文脈自由言語の種々のクラスにたいする効率的な推論アルゴリズムを報告する．

ABSTRACT

This report summarizes the recent results concerning the authors' work on the inductive inference of formal languages, which is one of the most significant subjects in the research on Machine Learning.

Generally, "inductive inference" is taken as a task of finding rules underlying from given finite examples, and the research on this subject is indispensable for realizing a machine learning mechanism.

In this report, "formal languages" are employed as a class of inference objects. Formal language theory has its long (say, more than 30 years) history of research activity, and its mathematical properties enable us to easily implement various algorithms developed on a practical computer system.

Although a problem of inductive inference for formal languages has been longly discussed in the context of "grammatical inference problem", its computational complexity has prohibited itself from further development.

In this report, we employ a problem setup in which besides given examples, various types of information on the target language are available, and present efficient inference algorithms for various subclasses of context-free languages.

目次

第 1 章

はじめに

帰納的推論とは，平たくいって「一を教えられて，十を推測する」ことであり，また少し改まっていうと「有限個の具体的な例からそれらを含む無限の例を説明する規則を見つける」ことである．また，言語の学習とは，「その言語を定義している文法体系を学ぶ」ことであろう．したがって，帰納的推論による言語学習とは，「その言語に関する有限個の具体例から，その言語を規定している文法を見つけること」である，と規定できる．

一般に，学習には，「新しい知識を獲得する」こと，および「以前より上手に行なえるようになる」ことの，ふたつのタイプがあるが，ここでの主要テーマである“言語の学習”では，前者に関する話題が中心になる．(後者に関する研究は，前者に比べて形式化がより難しく，したがって理論的解明が遅れていると思われる．)

本章では，学習に関する研究を，特に“機械学習”の観点から概観し，帰納的推論による言語学習の研究目的 / 意義等について，私見もまじえて，簡単にふれてみる．

1.1 機械学習と帰納的推論

“機械学習”の歴史は，Rosenblatt のパーセプトロンに象徴されるように，神経ネットワークの研究に始まる 1950 年代にさかのぼることができる．初期には，決定理論に基づく線形判別関数の学習が研究の中心であり，パターン認識に盛んに応用された．また，統計的決定理論に基づく方法も試みられたが，Samuel のチェッカー・プログラム等の成功例は少なく，実用的な成果への期待は遠のいた．

60 年代には，これらの経験を踏まえて“概念の学習”研究が盛んに行なわれた．これは，それまでのパラメタ (数値) の学習と異なり，概念の記号表現を学習の対象にしている．Winston の例からの構造記述の学習システム，Buchanan の META-DENDRAL システム等が報告され，また Mitchell の version space の提案もなされた．

その後，70 年代の後半には，これらの個別の概念学習の研究から，多くの予備 (基礎) 知識を前提としてより広範な，有益な概念の学習研究へと移行していった．Winston の類推による学習，あるいは Lenat の初等数学の概念学習システム AM 等が知られている．

ごく近年になって，パーセプトロンの再来ともいえる“コネクションマシン”が注目されているが，Minsky と Papert によって指摘された理論的限界をどう乗り越えるかが，これからの研究の鍵となろう．

さて，このような機械学習の研究潮流にあって，我々の研究アプローチの基礎になっている帰納的推論（帰納的学習）の理論の歴史は，Moore の順序機械の同定にその端を発しているが，1960年代後半の Gold による極限における学習に関する仕事が最も重要であろう．帰納的推論の“理論的研究”は文法推論という問題意識から始まった．（帰納的推論の理論的研究の対象は，関数・プログラム・論理式等々あるが，紙面の都合上，ここでは文法についてのみふれる．）Gold は，人間の言語習得の数学モデルとして生成文法の各クラスの推論問題を考察し，この分野の理論的な基礎を与えた．その後，Biermann らによる正則言語の推論，あるいは Knob らの文脈自由言語の推論等，具体的領域に対するアルゴリズムが提案されて行った．また，Fu らによるパターン認識への応用も考えられた．

1980 年代にはいって，計算量の理論の発展にともない，アルゴリズムそのものだけでなくその時間的効率が重要視されてきた．その中にあって，Angluin による正則言語の推論問題における効率と質問数に関する解析，あるいは，質問の種類と効率との関係に関する最近の研究は，注目すべき成果である．また，Valiant による確率的な学習可能性概念の提案もなされ，より柔軟な学習可能性概念が指向されるようになってきた．

このように，“機械学習一般”の研究に対する“帰納的推論”の研究の特徴は，前者が学習という広範な話題を多くの側面から，特に認識モデルに関する研究を中心に「これができる」を主なテーマに行なわれているのに対して，後者は，その一側面を 特に理論的な考察を基礎に「これは，これぐらいでできるが，これはできない」を主張しているように思われる．

文法推論とは，ある言語に属する有限個の具体的な記号列（語ともいう）が与えられたとき，その言語を規定している文法を見つけ出すことをいう．すなわち，有限個の具体例から，それを定義している文法，したがって対応する言語を帰納的に学習することである．（本稿では，“文法推論”と“言語の帰納的学習”とを同義で用いることにする．）

1.2 なぜ形式言語の学習か

我々は，帰納的推論による学習研究の対象領域として，形式言語を選択したが，これには幾つかの理由がある．形式言語の理論は 1950 年代の半ばにその起源をもち，初期には自然言語理解の形式モデルとして，また 1960 年代後半にはむしろ種々のプログラミング言語との密接な関係からその研究が急速に進展した．したがって，すでに理論的研究成果の蓄積がなされていてそれらを利用することにより，形式的（厳密）な議論が可能になる．と同時に，計算機上でのインプリメンテーション / 実験が容易である．形式言語の理論はまた，帰納的関数の理論，プログラムの理論，述語論理等々の他の計算機に関連する研究分野との関係が明確であり，形式言語の学習に関して得られた

成果をこれらの領域へ応用することも，またこれらの領域での結果を形式言語の学習研究に生かすこともできる．

このように，形式言語の理論は，単に言語の数学的モデルというのにとどまらず，人間に関わるあらゆる“概念の抽象モデル”として考えられる，わけである．

1.3 計算的学習研究の意義

前述したように，“機械学習”における研究潮流にはふたつの流れがある．我々の選択しているアプローチは，ごく最近になって提唱された**計算的学習理論**に基づいている，といえる．そこでの基本的な考え方は，「あるタスクを計算機に行なわせる前にできる限りその理論的な評価を行なっておこう」というものである．そうすれば，「なぜうまく行く(あるいは行かない)」のかが分かり，それを基にその後の研究方針の決定が容易になる．また，不可能なことを追い求め続けるような危険も回避できることになる．(ここでは，“言語の学習”にのみ焦点をあてたが，こうした研究姿勢(アプローチ)は，ロボティックス，神経ネット，あるいはパターン認識等の他の学習の研究領域に対しても適用できることは言うまでもない．)

1.4 学習モデル

このような基本理念に基づいて，我々は以下のような**学習モデル**を設定し，言語の帰納的学習の研究を行なった．すなわち，学習者は“システム”(あるいは“アルゴリズム”)であって，学習対象は“形式文法”(あるいはそれと等価な表現形式)である．モデルの設定によっては，教師として“ユーザ”を仮定することもある．学習者は，与えられた有限個の具体例から，必要ならば予め許された幾種かの質問を教師に対して行いながら，未知の文法を見つけ出すことが目的であり，以下の基準によって学習がなされたとみなす．すなわち，**極限における学習**と**有限時間学習**というふたつの考え方である．前者は，学習が完了したことを極限において認める考え方であり，学習者にはいつ学習が完了したかは判断できない．後者は，ある有限の時間内に正しい文法を見い出して学習が完了することを要求する．

本報告書では，このような学習の問題設定に基づいて行なわれた幾つかの研究成果が，以下に述べられている．

第3章では，記号列の具体例からの文脈自由言語の帰納的学習問題が扱われ，“極限における学習”と“有限時間学習”のふたつの問題設定において，その学習アルゴリズムが考察される．

第4章では，記号列の具体例のみならず構造情報をも用いた，文脈自由言語の学習問題が考察され，効率的なアルゴリズムが与えられる．

第5章では，記号列の具体例からの線形文脈自由言語の学習問題を通して，“記号列の変換機能の帰納的学習”に焦点をあて，変換機能のクラスとその学習に必要な教師の能力との関係が考察される．

最後に，今後の課題について簡単にふれられる．

本報告書は，(原則として) 上記の各話題に関する要点のみをまとめ上げたものである．したがって，より詳細は巻末にあげた参考文献を参照されることを希望する．

第 2 章

準備

ここでは, 本報告書で用いる形式言語と帰納的推論の概念と用語を説明する. ここで与えられないものに関しては, [Har78], [HU79], [Sal73], [AS83] などを参照されたい.

Σ をアルファベット, Σ^* を Σ 上のすべての記号列 (または語) の集合とする. ただし, Σ^* は空記号列 λ を含む. $lg(u)$ を記号列 u の長さとする.

アルファベット Σ 上の言語とは, Σ^* の部分集合である.

決定性有限オートマトン M は 5 項組 $(K, \Sigma, \delta, q_0, F)$ である. ここで, K は状態の空でない有限集合, Σ は有限集合である入力アルファベット, δ は $K \times \Sigma$ から K への遷移関数, q_0 は初期状態で K の要素, F は最終状態の集合で K の部分集合である. 次のように, 遷移関数 δ を $K \times \Sigma^*$ から K への関数に拡張する: 任意の状態 q にたいして,

$$\delta(q, \lambda) = q$$

任意の記号列 $u, v \in \Sigma^*$ にたいして, $\delta(q, uv) = \delta(\delta(q, u), v)$.

M によって受理されるすべての記号列の集合は $T(M)$ で表され,

$$T(M) = \{u \in \Sigma^* \mid \delta(q, u) \in F\}$$

である.

非決定性有限オートマトン M' は 5 項組 $(K, \Sigma, \delta', q_0, F)$ である. K, Σ, q_0, F は決定性有限オートマトンの場合と同じであるが, δ' は $K \times \Sigma$ から 2^K への関数である. 遷移関数 δ' も $2^K \times \Sigma^*$ から 2^K への関数に拡張される: 任意の状態 $q \in K$ にたいして,

$$\delta'(q, \lambda) = \{q\}$$

任意の記号列 $u, v \in \Sigma^*$ にたいして,

$$\delta'(q, uv) = \{p \mid \text{ある状態 } r \in \delta'(q, u) \text{ にたいして, } p \in \delta'(r, v)\},$$

$Q \subseteq K$ にたいして,

$$\delta'(Q, u) = \bigcup_{q \in Q} \delta'(q, u).$$

M' によって受理される記号列の集合 $T(M')$ は

$$T(M') = \{u \in \Sigma^* \mid \delta'(q_0, u) \cap F \neq \emptyset\}$$

である. 定義により決定性有限オートマトンは非決定性有限オートマトンである.

集合 R が非決定性有限オートマトンによって受理されるならば, R を受理する決定性有限オートマトンが存在する. 決定性有限オートマトンによって受理される Σ^* の部分集合 R を**正則**という.

文脈自由文法 G は 4 項組 (N, Σ, Π, S) である. ここで, N は非終端記号の空でない有限集合である. N と Σ は共通要素を持たないと仮定する. $N \cup \Sigma$ を V で表す. Π は生成規則の空でない有限集合である; 各生成規則は $A \to x$ の形で表される. ここで, A は非終端記号, x は V 上の記号列である. ここでは, ラベル π_i をつけて生成規則を区別する. S は特別な非終端記号で, 開始記号と呼ばれる.

$G = (N, \Sigma, \Pi, S)$ を文脈自由文法とする. V^* の記号列の間に次の関係 $\overset{\pi_i}{\underset{G}{\Longrightarrow}}$ を定義する. 任意の V^* の記号列 x と y にたいして, $x = uAv$, $y = uzv$, $\pi_i : A \to z$ が G の生成規則ならば, $x \overset{\pi_i}{\underset{G}{\Longrightarrow}} y$ である. このとき, 生成規則 $\pi_i : A \to z$ が記号列 x に適用され, y が得られた, と言われる.

$x_0, x_1, \ldots, x_k$ を V^* の記号列とする.

$$x_0 \overset{\pi_1}{\underset{G}{\Longrightarrow}} x_1,\ x_1 \overset{\pi_2}{\underset{G}{\Longrightarrow}} x_2,\ \ldots,\ x_{k-1} \overset{\pi_k}{\underset{G}{\Longrightarrow}} x_k$$

であるならば, $x_0 \overset{\alpha}{\underset{G}{\Longrightarrow}} x_k$ と表し, **付随語** α をもつ G における x_0 から x_k への**導出**と言う. ここで, $\alpha = \pi_1\pi_2\cdots\pi_k$ である. 付随語に関心がない場合には, α を省略して, $x_0 \overset{\alpha}{\underset{G}{\Longrightarrow}} x_k$ を単に $x_0 \underset{G}{\Longrightarrow} x_k$ と書く.

文脈自由文法 $G = (N, \Sigma, \Pi, S)$ によって生成される言語 $L(G)$ は,

$$L(G) = \{w \in \Sigma^* \mid S \overset{\alpha}{\underset{G}{\Longrightarrow}} w,\ \alpha \in \Pi^*\}$$

であり, **文脈自由言語**という. 文脈自由文法 G_1, G_2 は, $L(G_1) = L(G_2)$ のとき, **等価**であるといわれる.

次の条件を満たす文脈自由文法 $G = (N, \Sigma, \Pi, S)$ を**簡約された文脈自由文法**という :

- Π は $A \to A$ の形をした生成規則を含まない,
- $A \to \lambda$ の形をした生成規則が Π に含まれるならば, A は開始記号 S である,
- 任意の非終端記号 A にたいして, $S \overset{\alpha}{\underset{G}{\Longrightarrow}} uAv \overset{\alpha'}{\underset{G}{\Longrightarrow}} w$ である $w \in L(G)$ が存在する.

任意の文脈自由文法 G に対して, $L(G) = L(G')$ である簡約された文脈自由文法 G' が存在する.

$G = (N, \Sigma, \Pi, S)$ を文脈自由文法とする. **導出**

$$u_1A_1v_1 \overset{\pi_1}{\underset{G}{\Longrightarrow}} u_2A_2v_2 \overset{\pi_2}{\underset{G}{\Longrightarrow}} \cdots \overset{\pi_{n-1}}{\underset{G}{\Longrightarrow}} u_nA_nv_n$$

において, $u_i \in \Sigma^*$ $(1 \leq i \leq n)$ のとき, この導出を最左導出という. $L(G)$ の中にふたつ以上の相異なった最左導出を有する記号列が存在しないとき, G を**無あいまい**であるという.

$\mathcal{M}$ がある未知の言語 L を推論しようとする文法推論アルゴリズムとする. $\mathcal{M}$ が L のしだいに増加する事例の集まりにたいして繰り返し適用され, $\mathcal{M}$ の仮説である文法の無限列が $G_1, G_2, \ldots$ のように生成されるものとする. もしある数 m が存在して, G_m が L を生成する文法であり,

$$G_m = G_{m+1} = G_{m+2} = \cdots$$

とすると, $\mathcal{M}$ はこの事例の系列の上で**極限において L を正しく同定する**といわれる. もし $\mathcal{M}$ が G_m を出力したあと停止するならば, $\mathcal{M}$ は**有限の計算 (時間) で L を正しく同定する**といわれる.

第 3 章

記号列からの文脈自由言語の学習

3.1 背景

一般に, 言語の学習とは, 「その言語を規定している法則を学ぶ」ことであり, 法則は通常“文法”とよばれる. 自然言語の学習には自然言語の文法が, プログラミング言語の学習にはそのプログラム (文) を規定している文法が各々対応する. これらの状況を形式的に記述し, その数学的な性質を議論し, 何が本質的であるかを解明するための一手法として, “形式言語の理論”がある. 形式言語は, あらゆる自然言語, あるいは人工言語を記号表現によって抽象化したものであり, 形式文法によって規定 (生成) される. この意味で, 形式言語の学習のことを一般に, “文法推論”とよぶ.

ここでは, 文脈自由言語にたいする文法推論問題を考察する. (この研究の動機, その意義等については, 前述されているので省略する.) 本報告書の第 4章「例題からのパーザの自動合成」では, 構造記述の例からそれらを規定する文法 (パーザ) を自動合成する問題を考察するが, ここでは, あくまで「(構造を持たない) 記号列レベルの具体例からの文法学習」の問題を扱う. 本章では,

[課題 1] 記号列の具体例からの文脈自由言語の文法推論問題にたいするひとつの解 (学習アルゴリズム), および

[課題 2] 記号列の具体例とある種の補助的情報とを用いる単純 (決定性文脈自由) 言語の効率的な学習アルゴリズム,

のふたつを論ずる.

3.2 学習モデルの設定 — 学習可能性の定義

前述されたように, 学習可能性を形式化する方法には, (1) 極限における学習と (2) 有限時間学習というふたつの考え方がある. 前者は, 学習が完了したことを極限において認める考え方であり, これまでの言語学習の研究において支配的な地位を占めている. 実際, 人間の言語獲得の形式的モデルとしての Gold [Gol67] による説明は, 説得力がある. しかし, 言語学習以外の, 帰納的推論のメカニズム一般を論じ, また応用可能性を考えるとき, 後者のように, ある有限の時間内にしかも効率的な時間において学習が完了してくれることが望ましい.

これとは別の観点から，学習モデルを外部情報（あるいは追加情報）の有無で区別することも可能である．外部情報は，しばしば**オラクル（神託）**とよばれ，教師の能力として仮定される．このような教師付き学習は，あらゆる現実の学習現象をモデル化したものとして，広く受け入れられている．前出の［課題 1］は，極限における学習の原理に基づく，文脈自由言語の**教師なし学習**を，また［課題 2］は，有限時間学習に基づく，単純言語の教師付き学習を，各々扱っていることになる．

一般に，“アルゴリズム”とは，機械的に実行可能な命令系列の有限列であって，必ず有限時間内に出力して停止するものをいう．が，教師付き学習の問題設定を考える時は，機械的な命令系列のなかに，オラクルへの問い合わせという過程が含まれてくるために，得られる手続きは純粋な本来のアルゴリズムとは異なるものとなる．（しかし，ここでは，このようなオラクルを含んだものも“アルゴリズム”と呼ぶことにする．）

3.3 表現による学習 — 文脈自由表現

“文法推論”という言葉がしめすとおり，この課題における従来の研究の多くは，未知の言語 L を生成するある文法 G を見い出すことが目的であった．が，この“帰納的な学習”という問題の本質からして，文法を求めることが唯一の解決法ではない．（言語 L を規定できる表現形態であれば，文法にこだわる必然性はないわけである．）この考え方を，“文脈自由言語の学習”に適用することによって，「**簡潔で，理解し易い，完全な（正しい）アルゴリズム**」を得ることができる．

“文法”に代わる表現形態を，**文脈自由表現**とよぶ．文脈自由表現は，いわゆる正則言語にたいする“正則表現”の拡張概念である．文脈自由表現を定義するまえに準備が必要である．

定義　σ を記号，L, L_1, L_2 を言語とする．このとき，

1. L_2 **の** L_1 **への** σ**-代入**（これを $L_1 \uparrow^{\sigma} L_2$ と表わす）とは，

$$L_1 \uparrow^{\sigma} L_2 = \{x_1 y_1 \cdots x_k y_k x_{k+1} | x_1 \sigma \cdots x_k \sigma x_{k+1} \in L_1, \sigma \text{ は } x_1 \cdots x_{k+1} \text{ には現われない，また } y_i \in L_2 \ (1 \leq i \leq k)\}.$$

2. また，L **の** σ**-反復**（これを L^{σ} と表わす）とは，

$$L^{\sigma} = \{x | x \in L \cup L \uparrow^{\sigma} L \cup \cdots, \text{かつ } x \text{ は } \sigma \text{ を含まない}\}$$

をいう．

定義　Γ を（一般には，無限集合の）アルファベット，$\Gamma' = \{\sigma | \sigma \in \Gamma\}$ とする．Γ 上の**文脈自由表現**とは，以下で定義される $\Gamma \cup \Gamma' \cup \{\Phi, +, (,)\}$ 上の記号列である：

1. Φ は文脈自由表現である

2. $\Gamma \cup \{\epsilon\}$ の要素 a は文脈自由表現である

3. E_1, E_2 が文脈自由表現であり, σ が Γ の要素であるならば, $(E_1 + E_2), E_1E_2, (E_1)\boldsymbol{\sigma}$ は文脈自由表現である

4. 以上の手続きで得られるものだけが文脈自由表現である.

Γ 上の文脈自由表現に対して, 以下のように言語を対応させることができる:

定義　文脈自由表現のクラスから言語のクラスへの写像 $||$ を以下のように定義する:

1. $|\Phi| = \emptyset$

2. $|a| = \{a\}$ ($\Gamma \cup \{\epsilon\}$ の任意の要素 a にたいして)

3. $|E_1 + E_2| = |E_1| \cup |E_2|$, $|E_1E_2| = |E_1||E_2|$, かつ $|E_1\boldsymbol{\sigma}| = |E_1|^{\sigma}$

例 3.1 a, b, σ を Γ の要素とするとき,

$$E = (a\sigma b + ab)\boldsymbol{\sigma}$$

は文脈自由表現であり, これに対応する言語は

$$|E| = |(a\sigma b + ab)|^{\sigma} = \{a^n b^n | n \geq 1\}$$

となる. これは正則でない文脈自由言語の典型的な例であり, 文法

$$G = (\{S\}, \{a, b\}, \{S \to ab, S \to aSb\}, S)$$

によって生成される. (この例では, σ が非終端記号 S に対応しているが, 一般には必ずしもそうならないことに留意されたい.)

さて, 文脈自由表現による“文脈自由言語の学習 (推論)” は以下の結果に基づいている.

定理 3.1 L を Σ 上の言語とする. このとき, L が文脈自由言語であるための必要十分条件は, ある有限アルファベット T と T 上の文脈自由表現 E があって, $\Sigma \subseteq T$ かつ $|E| = L$ となることである.

このように, 文脈自由表現は, 文脈自由言語を規定する文法に代わる役割を果たす.
例 3.1からも分かるように, 文脈自由表現によって言語を規定する方法の利点は,

1. “表現” が一次元の記号列で容易に表わされる,

2. “表現” と言語の要素 (記号列) とが同一のレベルで扱える,

3. (学習対象である) 表現を, 与えられた記号列からの一般化として得られる,

等がある.

3.4 学習アルゴリズム

Ω を, ある固定されたアルファベット Γ 上の文脈自由表現全体の集合とする. このとき, 以下のような Ω 上の変換演算 δ を考える: ($E_2 \in \delta(E_1)$ であることを, $E_1 \to E_2$ で表わす.)

1. $\Phi \to \Phi\Phi$
2. $\Phi \to a(\forall a \in \Gamma \cup \{\epsilon\})$
3. $\Phi \to (\Phi)\boldsymbol{\sigma}(\forall \sigma \in \Gamma)$
4. $\Phi \to (\Phi + \Phi)$
5. $E_1 \to E$ ならば $E_1 + E_2 \to E + E_2$ かつ $E_2 + E_1 \to E_2 + E$
6. $E_1 \to E$ ならば $E_1\boldsymbol{\sigma} \to E\boldsymbol{\sigma}(\forall \sigma \in \Gamma)$
7. $E_1 \to E$ ならば $E_1E_2 \to EE_2$ かつ $E_2E_1 \to E_2E$

このとき, 演算 δ は以下の性質を持つ:

補題 3.1 Φ から δ によって導かれるすべての表現の集合を $\delta^*(\Phi)$ とする. このとき,

1. $\delta^*(\Phi)$ は, すべての文脈自由言語 L にたいして $L = |E|$ となる文脈自由表現 E を少なくともひとつ含んでいる.
2. Ω の任意の表現 E_1, E_2 にたいして, $E_1 \in \delta(E_2)$ ならば $|E_2| \subseteq |E_1|$ である.

補題 3.1の 1は, Ω 上の演算 δ は, 文脈自由言語全体に対する文脈自由表現の集合を枚挙するために十分な能力を保持していることを示している. また, 補題 3.1の 2は, δ によって枚挙される表現の言語表現能力は, δ の適用回数にしたがって単調に増加する, ことを示している.

例 3.2 文脈自由言語 $L = \{w \in \{a,b\}^* | \sharp_a(w) = \sharp_b(w)\}$ (ここに $\sharp_x(w)$ は w に現われている記号 x の個数を表わす) を考えよう. このとき, $|E| = L$ なる文脈自由表現 E は以下のように, Φ から δ によって導かれる:

$$
\begin{aligned}
\Phi &\to (\Phi)\boldsymbol{\tau} \\
&\to (\Phi\Phi)\boldsymbol{\tau} \\
&\to ((\Phi + \Phi)\Phi)\boldsymbol{\tau} \\
&\to^* ((a(\Phi)\sigma + \Phi)\Phi)\boldsymbol{\tau} \\
&\to^* ((a(\Phi\Phi\Phi + \Phi)\boldsymbol{\sigma} + \Phi)\Phi)\boldsymbol{\tau}
\end{aligned}
$$

$$
\begin{aligned}
&\to^* \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + \Phi)\Phi)\boldsymbol{\tau} \\
&\to \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + \Phi\Phi)\Phi)\boldsymbol{\tau} \\
&\to^* \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + b(\Phi)\boldsymbol{\nu})\Phi)\boldsymbol{\tau} \\
&\to^* \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + b(\Phi\Phi\Phi + \Phi)\boldsymbol{\nu})\Phi)\boldsymbol{\tau} \\
&\to^* \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + b(b\nu\nu + a)\boldsymbol{\nu})(\Phi + \Phi))\boldsymbol{\tau} \\
&\to^* \quad ((a(a\sigma\sigma + b)\boldsymbol{\sigma} + b(b\nu\nu + a)\boldsymbol{\nu})(\tau + \epsilon))\boldsymbol{\tau}
\end{aligned}
$$

すなわち, $L = |((a(a\sigma\sigma + b)\boldsymbol{\sigma} + b(b\nu\nu + a)\boldsymbol{\nu})(\tau + \epsilon))\boldsymbol{\tau}|$, である.

アルゴリズムの概要

d_0 を学習の対象となる Σ 上の文脈自由言語とする. 我々は, d_0 の正の例 (d_0 に属する語) と負の例 ($\Sigma^* - d_0$ に属する語) のすべてがいつかは必ず現われるような, 具体例を枚挙するオラクルを仮定する. これを**d_0 の完全提示オラクル**と呼ぶことにする.

学習アルゴリズムは, "極限における学習" の原理に基づいている. アルゴリズムは, ある固定された順序にしたがって, 最も簡単な出発点の表現である Φ から, すべての可能な "表現" を枚挙していく. すなわち, $\delta^*(\Phi)$ の要素を次々に枚挙していく.

δ の性質 (補題 3.1の 2) から, この枚挙は, より簡単 (特殊) なものから複雑なものへと次々に作り出される. d_0 の完全提示オラクルからそれまでに与えられた, 正負の具体例の集合に対して, 一旦 「複雑すぎる (一般的すぎる)」と判明した表現は, 単に捨てさられる. また, 「簡単すぎる (特殊すぎる)」ときは, それをより複雑な (より一般的な) 表現にしていく. また, (その時点の具体例の集合に対して) 一旦正しいと判定された表現は, それが新しい具体例によって反駁されるまでは, 不変に保たれる.

$\delta^*(\Phi)$ の枚挙は, 実際, 表現に現われる補助記号 ("反復" 演算に対応するギリシャ文字) の個数 k と, 出発点 Φ にたいする δ の適用回数 i との間の, いわゆる "ダブテイリング" 技法に基づく.

このように, アルゴリズムは, 「i より簡単 (特殊) な表現から複雑なものへと次々に枚挙していき, いつかは必ず正しい (目標の) 表現のうちの最も簡単なものが得られることが保証される」ように設計される.

アルゴリズムの収束性 / 正当性

定理 3.1および補題 3.1の 1より, $d_0 = |E|$ なる表現 E を導く Φ からの δ による導出が存在する. アルゴリズムの枚挙技法から, そのような E はある時点で必ず現われ, したがって, それ以後は, 変化せずに保たれる (収束性). また, 一旦, ある時点である表現に収束したならば, その "仮説 (表現)" は, 正しい. さもなければ, d_0 の完全提示オラクルを仮定しているから, ある時点においてその仮説は反駁され, 更新されることになるからである (正当性).

学習アルゴリズムの詳細が図 3.1に与えられる.

結果として, 以下の定理を得る:

定理 3.2 与えられた文脈自由言語 d_0 に対して, アルゴリズム **IA1** は極限において d_0 を学習する.

表現による文脈自由言語の帰納的学習アルゴリズムの特徴は以下のようにまとめられる:

1. (正則表現による) 正則言語学習アルゴリズムの自然な拡張として, 簡潔かつ完全なアルゴリズムである.

2. 特に, (準線形言語と呼ばれる) ある部分クラスに限った場合, 非終端記号の同定という作業から解放されることにより, 表現の枚挙がさらに簡潔になる.

3. 計算量の明確な把握が可能であり, その限界も分かっている.

3.5 協調問題解決としての学習

3.5.1 教師と生徒

前述したように, 学習アルゴリズムの応用を考えるとき, 効率の問題は最重要課題であるが, 与えられた記号列のみを用いての学習 (いわゆる教師なし学習) の場合は, その効率の悪さは本質的である. したがって, 学習という課題を, 生徒であるアルゴリズムと教師であるユーザとの「協調による問題解決の作業」としてとらえることが必要になってくる.

教師は, 生徒からの質問に正しく答えることが要求され, 生徒はその質問の答えを使って, 有限時間内 (できれば多項式時間) で正しい "解 (文法)" を出すことが期待される.

ここでは, 教師に課せられる能力 (オラクル) として, 二種類を考える: 任意の記号列 (語) w に関して, 「w?」なる形の質問にたいして, **接頭語所属性オラクル**は, もし w を接頭語とする語が未知の言語 L のなかにあるならば, そのうちの最短のもの (のひとつ) を答え, さもなければ "No" と答える. **導言語同値オラクル**は, 任意のふたつの語対 (u_1, v_1), (u_2, v_2) に関して, 「$u_1 \backslash L / v_1 = u_2 \backslash L / v_2$?」なる質問に "Yes-No" で正しく答えなければならない. (ここに, $u\backslash L(L/v)$ は, L の要素で u を接頭語 (v を接尾語) として持つもの全体の集合を表わす.)

3.5.2 単純言語の効率的な学習

ここでは, 文脈自由言語の部分クラスである**単純言語**を学習するアルゴリズムを論ずる. 単純言語は以下で定義される単純文法によって生成される:

定義 $G = (N, \Sigma, \Pi, S)$を文脈自由文法とする.

学習アルゴリズム IA1

入力: 枚挙可能な (文脈自由) 表現の集合 Ω,
枚挙のための演算 δ,
d_0 の完全提示オラクル
出力: 表現の系列 $E_1, E_2, \ldots, E_n, \ldots,$ であって,
E_n は最初の n 個の具体例にたいして正しい
手続き:

$Q \leftarrow \Phi$;(Q はキュー)
$EXAM \leftarrow \emptyset$
$X \leftarrow next(Q)$;($next(Q)$ は Q の先頭の要素)
do(forever)
  $EXAM \leftarrow EXAM \cup \{$新しい例$\}$
  while ($X = E$ を現在の仮説 (表現) とする)
    ある $e \in EXAM$ で E にたいして正しくない
    if E が「特殊化過ぎる」
    then $\delta(E)$ を Q の後尾に付ける;
      $X \leftarrow next(Q)$;
    else E が「一般化過ぎる」
      E を捨てさる ;
      $X \leftarrow next(Q)$;
  E を出力する;

ここに,
E が「特殊化過ぎる」とは, $EXAM$ の正の例 e で $e \notin |E|$ なるとき,
E が「一般化過ぎる」とは, $EXAM$ の負の例 e で $e \in |E|$ なるとき,
をいう.

図 3.1: 文脈自由言語の学習アルゴリズム

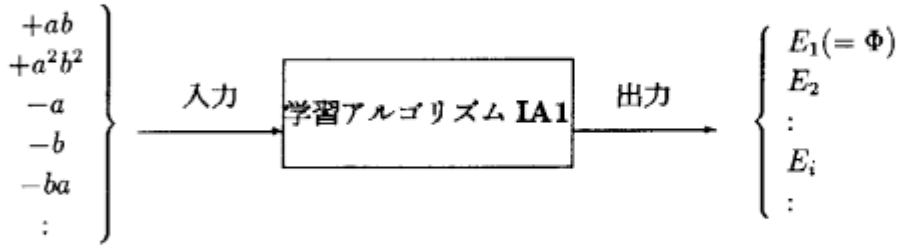

図 3.2: アルゴリズム IA1 の学習過程

1. Gが**単純文法**であるとは,

$$\text{もし } A \to a\alpha, A \to a\beta \in \Pi \text{ ならば } \alpha = \beta$$

が成り立つときをいう. ここに, $a \in \Sigma, A \in N, \alpha, \beta \in N^*$ である.

2. 言語 L が**単純言語**であるとは, ある単純文法 G があって $L = L(G)$ となることである.

例 3.3 文脈自由文法 $G = (\{S, A, B, C\}, \{a, b\}, \Pi, S)$ を考えよう: ここに,

$$\Pi = \{S \to aAC,\ A \to a,\ A \to bAB,\ B \to b,\ C \to a\}$$

である. この G は単純文法であり, $L(G) = \{ab^n ab^n a | n \geq 0\}$ である.

単純文法には, いわゆる "2−正規形定理" が存在する.

補題 3.2 任意の単純文法に対して, 等価な単純文法 $G = (N, \Sigma, \Pi, S)$ で,

1. $\forall A \in N, \exists S \Rightarrow^* \alpha A\beta \& A \Rightarrow^* w$ (ここに, $\alpha, \beta \in (N \cup \Sigma)^*, w \in \Sigma^*$),
2. $\forall A, B \in N (A \neq B),\ L(A) \neq L(B)$ (ここに, $L(X)$は, X から導かれる Σ 上の語全体の集合),
3. $\Pi \subseteq N \times (\Sigma \cup \Sigma N \cup \Sigma N^2)$,

を満たすものが存在する.

したがって, 以下では, この2−正規形を仮定して議論する.

単純言語に関する性質で基本的なものとして, まず, 以下の結果に注意されたい.

補題 3.3 L を Σ 上の言語とする. このとき,

1. もし, L が正則言語であるならば, $L\sharp$ (ここに, $\sharp$ は Σ に属さない新しい記号) は, 単純言語である.
2. もし L が単純言語であるならば, L を受理する決定性プッシュダウンオートマトンが存在する.

このように, 単純文法によって生成される言語のクラスは, 正則言語のクラスを (ある意味で) 真に包含し, 文脈自由言語のクラスに真に含まれる.

特徴カバーグラフ

与えられた単純文法 $G = (N, \Sigma, \Pi, S)$ にたいして, 以下のような方法によって, ある有向グラフ CC_G を考える:

1. S は CC_G におけるすべてのパスの出発点のノードであり, 各パスはノード F (新しい記号) で終わる.

2. F からでる辺はない.

3. 各 $a \in \Sigma$ に対して, $S \Rightarrow a\alpha_1 \Rightarrow \cdots w_k\alpha_k \Rightarrow w_{k+1}$ を, S から Σ 上の語を導ける最短の導出とするとき, 各 α_i をノードとし, $\alpha_i \rightarrow \alpha_{i+1}$ なるラベル b_i の各辺は CC_G に含まれる. (ここに, $\alpha_i \Rightarrow \alpha_{i+1}$ において規則 $A \rightarrow bB$ が用いられたとする.)

4. 今 $A\beta$ なるノードがあり, A がいまだに, ある $b \in \Sigma$ に対して用いられない規則 $A \rightarrow bB$ が存在するならば, $A\beta \Rightarrow bB\beta$ で始まる最短の導出に対応するパスを CC_G に含める.

以上の手続きは, 有限で終了し, 得られるグラフ CC_G は有限な有向グラフになる. これを, G の**特徴カバーグラフ**という.

例 3.4 例 3.3で考察された単純文法 G に対する特徴カバーグラフ CC_G は, 図 3.3の (a) のようになる.

さて, 次の結果は, 単純言語の効率的な学習アルゴリズムが存在するための重要な性質を与えている.

補題 3.4 与えられた単純文法 $G = (N, \Sigma, \Pi, S)$ に対して, その特徴カバーグラフ CC_G のサイズ (ノードの総数) は $(2t+1)mn$ 以下である. ここに, n は N の要素の数, m は Σ の要素の数, $t = max_{A \in N}\{lg(w_A) | w_A : A$ から導かれる最短の語 $\}$ である.

グラフから文法への復元

与えられた単純文法 $G = (N, \Sigma, \Pi, S)$ の特徴カバーグラフ CC_G を以下のようにして修正してできるグラフ g_G を考える:

1. もし CC_G のノードが $A\alpha$ $(A \in N \cup \{F\}, \alpha \in N^*)$ なるラベルを持つならば, g_G は A なるラベルのノードを持つ.

2. g_G の辺の集合は, CC_G のそれと同じである.

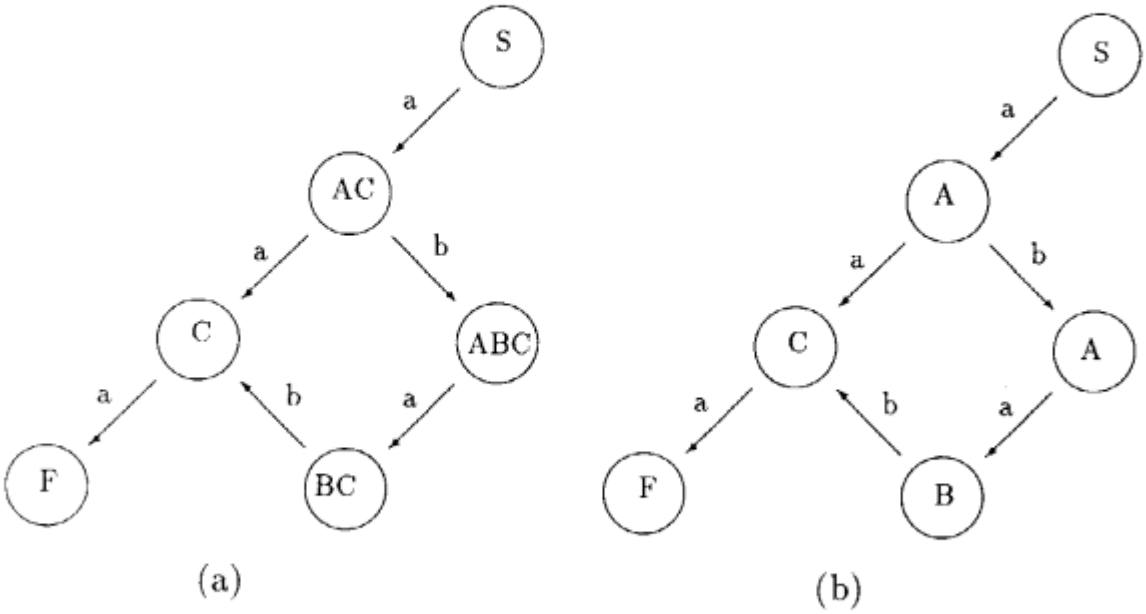

図 3.3: (a) 特徴カバーグラフCC_Gと (b) 基本グラフg_G

このグラフ g_G を**基本グラフ**という. 前例の文法 G にたいする基本グラフ g_G が図 3.2の (b) に与えられる.

基本グラフ g_G が与えられると, $L(G) = L(G')$ なる文法 G' が構成できる. (このとき, G' は, 一般に, 2−正規形でも, 単純でもないが, 容易にそのような性質を持つ等価な文法が構成できる. 次の例で見てみよう.)

例 3.5 図 3.3の (b) に与えられた基本グラフ g_G から, 再帰的な非終端記号 A に注目し, さらに初期記号以外の非再帰的な非終端記号は, それが導く終端記号列で置き換えることにより, G と等価な文法 $G' = (\{S, A\}, \{a, b\}, \{S \to aAa, A \to a, A \to bAb\}, S)$ を得る.

アルゴリズムの概要

未知の単純言語 L が与えられたとすると, アルゴリズムは単純言語の決定性的性質を用いて, その構造を表わすグラフ (正則集合の有限状態グラフの拡張概念に対応) を構成していく. (このグラフは, L を生成するある単純文法 G の特徴カバーグラフ CC_G に対応している.) その際, 接頭語所属性オラクルからの答えの最短性と導言語同値性オラクルによって, 冗長な非終端記号の導入が避けられ, 結果として得られるグラフは最簡なものになる. 目標の単純文法は, このグラフから容易に構成される.

学習アルゴリズム **IA2** の概要が図 3.4に与えられる.

学習アルゴリズム IA2 (概要)

入力: 単純言語L_0
L_0にたいする接頭語所属性オラクル PMO
L_0にたいする導言語同値性オラクル DEO
出力: $L_0 = L(G)$ なる単純文法 G
手続き:
用いるすべてのパラメタを初期化する;
すべての $a \in \Sigma$ にたいして,「a?」を PMO に質問し,
最初のラベル無しのグラフ g_0 を構成する;
repeat (現在のグラフを g_i とする)
while g_i のあるノード $[w]$ で, DEO によって推定できないものが存在する **do**
begin
すべての $a \in \Sigma$ に関して, そのノードに関する質問「wa?」を
オラクル PMO にたいして行ない, 拡張されたグラフ g_{i+1} を構成する;
g_{i+1} を新たに g_i とする;
end
until g_i におけるすべてのノードの推定に成功する
g_i から得られる文法 G_i を出力し, 停止する.

図 3.4: 単純言語の学習アルゴリズム

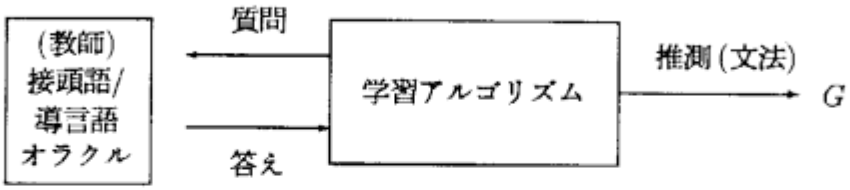

図 3.5: 対話的な学習過程

結果として, 以下の定理を得る:

定理 3.3 与えられた単純言語 L_0 に対して, アルゴリズム **IA2** は, m, n, t に関する多項式時間で L_0 を学習する. ここに, n は N の要素の数, m は Σ の要素の数, $t = max_{A \in N}\{lg(w_A)|w_A : A$ から導かれる最短の語 $\}$ である.

アルゴリズム **IA2** の特徴を列挙すると

1. アルゴリズムは, 教師 (ユーザ) との対話によって未知の言語 L に関する情報を得ていく.
2. アルゴリズムは, 目標のグラフのサイズの多項式時間で, 正しい文法を出して停止する.
3. (極限における学習アルゴリズムと異なり) アルゴリズムは推測の文法を最後に一度だけ出力して停止する.

第 4 章

例題からのパーザの自動合成

4.1 問題設定

例題を与えることによって文法を帰納的に推論し，その文法に基づくパーザを合成する問題を考える．有限個の文の例からその言語を規定する文法を推論する問題は，文法推論として知られている．ここで提案されるシステムは，この文法推論のメカニズムを基本に用いる．しかし今までの文法推論と異なり，文の構造記述が例としてシステムに与えられる．それはパーザの推論という問題においては，推論されるパーザの文法はコンパイラにおける文の変換とか解釈というような作業を伴う状況において使用されることが意図されているので，推論される文法は未知の言語を正しく生成するばかりでなく，その言語の文に正しい構造を割り当てなければならないからであり，そのためには文法の構造についての情報が推論システムによって利用可能でなければならないからである．この問題設定のもとで，我々は文脈自由文法全体のクラスを正，負の例から効率良く学習するアルゴリズム (システム名: Full-LESSON) と，正だけの例から文脈自由文法の部分クラスを効率良く学習するアルゴリズム (システム名: LESSON) を提案する．ここで効率の良いアルゴリズムとは，あるパラメータに関して，多項式時間の計算量で推論するアルゴリズムを言う．今までの文法推論，さらには帰納的推論が実用化されなかった理由に，この計算量の困難さがあり，ここではそれのひとつの解決を与えている．

4.2 構造記述と木オートマトン

文法推論の問題は，ある未知の言語の有限個の例からその言語にたいする“正しい”文法を同定する問題と定義される．ここで，“正しい”文法とは単にその言語を生成する文法を意味する．これらの文法は一般に，正則表現，有限オートマトンや文脈自由文法，変形文法などによって表わされる．ここでは文脈自由文法を推論する問題を考える．その理由はプログラミング言語などのほとんどの人工言語や多くの自然言語処理において，文脈自由文法がそれらの構文を規定する文法として用いられ，そしてそれらの構文解析に文脈自由文法のパーザが利用されているからである．そこでいくつかの基本的定義について述べる．

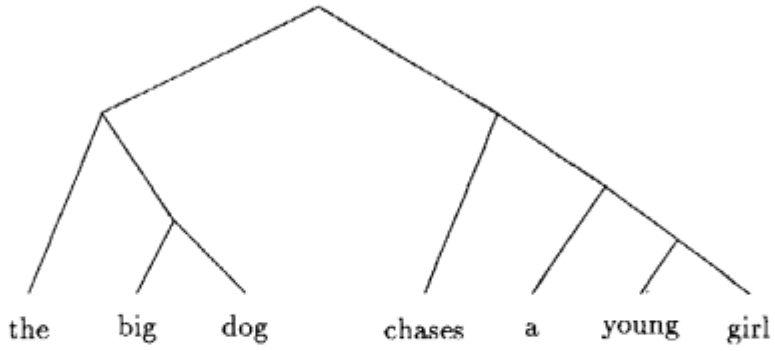

図 4.1: 文 "the big dog chases a young girl" にたいする構造記述の例

定義　$G = (N, \Sigma, \Pi, S)$ を文脈自由文法とする. 各 $A \in N \cup \Sigma$ にたいして, 集合 $D_A(G)$ を次のように再帰的に定義する:

$$D_A(G) = \begin{cases} \{a\} & (A = a \in \Sigma \text{ のとき}), \\ \{A(t_1, \ldots, t_k) \mid A \to B_1 \cdots B_k,\ t_i \in D_{B_i}(G)\ (1 \le i \le k)\} & (A \in N \text{ のとき}). \end{cases}$$

このようにして定められた $D_A(G)$ の各要素を, A からの G の**導出木**とよぶ. 開始記号 S にたいしては, 単に $D(G)$ と書く.

定義　t をある導出木とする. このとき t の**構造記述** $s(t)$ は, 次のように再帰的に定義される:

$$s(t) = \begin{cases} \sigma(s(t_1), \ldots, s(t_n)) & (t = A(t_1, \ldots, t_n) \text{ のとき}), \\ t & (t \in \Sigma \text{ のとき}). \end{cases}$$

導出木の集合 T にたいして, その構造記述の集合を $K(T) = \{s(t) \mid t \in T\}$ と定義する. このように, 構造記述は内部ノードがラベルを持たない (内部ノードがただ一種類のラベル σ を持つ) 一種の木である. 文脈自由文法 G の**構造記述の集合**とは, $K(D(G))$ である. ふたつの文脈自由文法 G_1 と G_2 が**構造的等価**であるとは, $K(D(G_1)) = K(D(G_2))$ が成り立つことを言う.

定義　アルファベットまたは空集合の有限列 $V = (V_0, V_1, V_2, \ldots, V_m)$ を**階層付きアルファベット**と呼び, V 上の**木**を次のように再帰的に定義する:

1. V_0 の元は高さ 0 の木である.
2. $t_1, t_2, \ldots, t_k$ がそれぞれ高さ $h_1, h_2, \ldots, h_k$ の木で $f \in V_k$ のとき, $f(t_1, t_2, \ldots, t_k)$ は高さ $\max\{h_1, h_2, \ldots, h_k\} + 1$ の木である.

このようにして定義される有限の高さの V 上の木の全体を V^T で表す.

階層付きアルファベット $V=(V_0,V_1,V_2,\ldots,V_m)$ の成分の和 $V_0\cup V_1\cup V_2\cup\cdots V_m$ を $\cup V$ と書き, V 上の木をアルファベット $\cup V$ の木とも呼ぶ. 階層付きアルファベットの成分は互いに共通部分を持ってもよい.

V 上の木を $\{1,2,\ldots,m\}^*$ のある有限部分集合から $\cup V$ への関数の形で表すこともできる. まず V 上の木にたいして $Dom(t)\subseteq\{1,2,\ldots,m\}^*$ を再帰的に次のように定義する:

$$Dom(f(t_1,t_2,\ldots,t_k))=\{\epsilon\}\cup 1\cdot Dom(t_1)\cup 2\cdot Dom(t_2)\cup\cdots\cup k\cdot Dom(t_k)$$

ここで, $t_1,t_2,\ldots,t_k\in V^T$, $f\in V_k$. この $Dom(t)$ から $\cup V$ への関数 t を次のように再帰的に定める:

$$\begin{cases} t(\epsilon)=f, \\ t(i\cdot u)=t_i(u) \quad \text{ここに } u\in Dom(t_i),\ i=1,2,\ldots k. \end{cases}$$

このように定義された $t:Dom(t)\mapsto\cup V$ を t の関数表現と呼ぶ.

定義　**木オートマトン** A は, 次の 4 項組 $A=(Q,V,\delta,F)$ よりなる: 状態の有限集合 Q, 階層付きアルファベット V, 受理状態の集合 F, と状態遷移関数 $\delta=(\delta_0,\delta_1,\ldots,\delta_m)$:

$$\delta_k:V_k\times(Q\cup V_0)^k\mapsto Q \qquad (k=1,2,\ldots,m),$$

$$\delta_0(a)=a \qquad \text{ここに } a\in V_0.$$

この遷移関数 δ を V^T 上の応答関数まで拡張する :

$$\delta(f(t_1,\ldots,t_k))=\begin{cases} \delta_k(f,\delta(t_1),\ldots,\delta(t_k)) & (\text{k¿0 のとき}), \\ \delta_0(f) & (\text{k=0 のとき}). \end{cases}$$

木オートマトン A によって**受理される**木の集合 $T(A)$ は, $T(A)=\{t\in V^T\mid\delta(t)\in F\}$ と定義される.

遷移関数 δ が $V_k\times(Q\cup V_0)^k$ から 2^Q $(k=1,2,\ldots,m)$ への関数であるときに, その木オートマトン A は**非決定性**と呼ばれる.

帰納的推論にたいする Gold による定式化 [Gol67] においては, 例の提示のしかた (学習プロトコル) と推論を正しいと見る基準が重要となる. 文法推論における学習プロトコルは大きくふたつに分けられる. 未知の言語に属する文だけを提示する正提示と, 未知の言語に属する文と属さない文のすべてを提示する完全提示がある. 推論を正しいと見る基準にたいして Gold は, 極限における同定という概念を導入した. そして彼はこの極限における同定という概念において, 正提示からの文法推論は, 完全提示からの文法推論よりも能力が劣ることを示した.

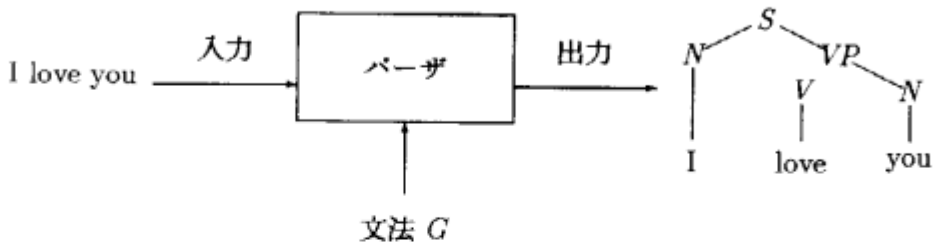

図 4.2: パーザの機能

4.3 パーザの推論

パーザを推論するという問題において, 例題とは何であろうか？まずパーザの機能について考えて見よう. パーザはある文を入力として受取り, 自分の持つ文法に従ってその文の構文解析を行ない, その結果として構文解析木 (parse tree) を出力する. この構文解析木の構造が重要である. それがユーザが意図したとおりの正しい構造となっていることが必要である. すると例題から文法を推論し, それからパーザを合成するという問題を考えるときは, 推論されたパーザによって解析, 出力される構文解析木が正しい (ユーザが意図したとおりの) 構造になっていることが要求されるであろう.

さて, 文法推論の問題は未知の言語 L の例から $L = L(G)$ なるある文法 G を同定する問題と定義された. しかし一般には, ひとつの言語にたいしてそれを生成する無限個の文法が存在する. そしてこれらの文法には, 異なる構造を持つものがいくつも存在する. したがって従来の文法推論においては, ユーザが意図しなかった構造を持つ文法が推論されるかもしれない. 例えば, 次の例を考えよ. 以下に定義される文法 G_1 は変数 v と乗算 $\times$, 加算 $+$ の演算を含むすべての算術式の集合を生成する.

$$S \to v \mid Av$$

$$A \to v+ \mid v\times \mid v + A \mid v \times A$$

(文法 G_1)

しかし文法 G_1 によって文にたいして割り当てられる構造は無意味である. 同じ言語が次の文法 G_2 によって意味ある構造を持って生成される.

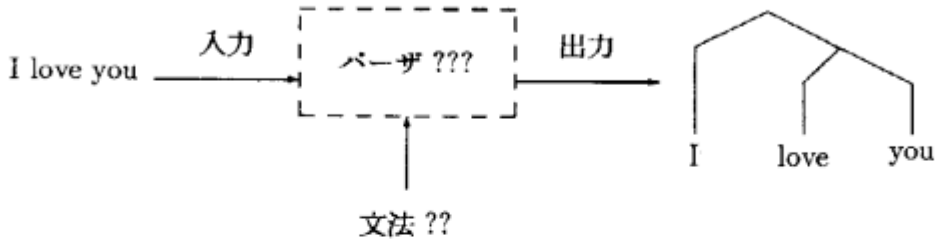

図 4.3: パーザの合成問題における例題とは ?

$$S \to E$$

$$E \to F \mid F + E$$

$$F \to v \mid v \times F$$

(文法 G_2)

ここに句構造はすべて算術の規則に関して意味のあるものとなっている. 文法 G_1 と文法 G_2 は等価である ($L(G_1) = L(G_2)$) が, この事実は実際的にはあまり意味が無い. 文法 G_1 のような文にたいして無意味な構造を割り当てる文法を考えることは, パーザを合成するという立場においては意味が無いからである. このようにパーザの推論という問題を考える時は, 推論されるパーザの文法が持つ構造がより重要となる. なぜならパーザの推論という問題においては, 推論されるパーザの文法はコンパイラにおける文の変換とか解釈というような作業を伴う状況において使用されることが意図されているので, 推論される文法は未知の言語を正しく生成するばかりでなく, その言語の文に意味のある構造を割り当てなければならない. しかし今までの文法推論という枠組みにおいては, 常に正しい (または意図したとおりの) 構造を持つ文法 (例えば文法 G_1 ではなくて, 文法 G_2) を推論することは明らかに不可能である. そこでここでは, 文法の構造についての情報が推論システムによって利用可能であることを仮定する. 文脈自由文法の場合には, 文法 G の構造は一般に構造記述の集合 $K(D(G))$ によって表される. したがって推論システムには, 未知の文法 G_U の構造記述の集合 $K(D(G_U))$ の要素が例題として与えられることになる.

一方, 文脈自由文法 G の導出木の集合 $D(G)$ は木オートマトンによって受理されることが知られている. さらに, 文脈自由文法の構造記述の集合 $K(D(G))$ もやはり木オートマトンによって受理される.

そこで次のような木オートマトンと文脈自由文法の間の変換によって, 互いに翻訳することができる.

定義　$G=(N,\Sigma,\Pi,S)$ を任意の文脈自由文法とすると，それに対応する (非決定性) 木オートマトン $NA(G)=(Q,Sk\cup\Sigma,\delta,F)$ は次のように定義される:

$$Q=N,$$

$$F=\{S\},$$

$$\delta_k(\sigma,B_1,\ldots,B_k)\ni A \quad (A\to B_1\cdots B_k\in P \text{ のとき}),$$

$$\delta_0(a)=a \qquad\qquad \text{ここに } a\in\Sigma.$$

補題 4.1 G を任意の文脈自由文法とする. この時, $T(NA(G))=K(D(G))$ が成り立つ. すなわち, $NA(G)$ によって受理される木の集合は G の構造記述の集合と一致する.

定義　$A=(Q,Sk\cup\Sigma,\delta,F)$ を任意の木オートマトンとすると，それに対応する文脈自由文法 $G(A)=(N,\Sigma,\Pi,S)$ が次のように定義される:

$$\begin{aligned} N &= Q\cup\{S\},\\ \Pi &= \{\delta_k(\sigma,x_1,\ldots,x_k)\to x_1\cdots x_k \mid \sigma\in Sk_k,\ x_1,\ldots,x_k\in Q\cup\Sigma\}\\ &\quad \cup\{S\to x_1\cdots x_k \mid \delta_k(\sigma,x_1,\ldots,x_k)\in F\}. \end{aligned}$$

補題 4.2 A を構造記述の集合を受理する任意の木オートマトンとする. この時, $K(D(G(A)))=T(A)$ が成り立つ. すなわち, $G(A)$ の構造記述の集合は A によって受理される木の集合と一致する.

これらの事実によって，文とその構造からの文脈自由文法のパーザの帰納的推論問題が木オートマトンの帰納的推論問題に還元される. すると，有限オートマトンにたいする効率的な帰納的推論アルゴリズムを木オートマトンにまで拡張することによって，文脈自由文法のパーザにたいする効率的な帰納的推論アルゴリズムが得られる. すなわち，構造例からのパーザの推論問題を木オートマトンの推論問題に還元し，推論された木オートマトンをあるコード化に従って文脈自由文法，そしてそのパーザに翻訳することによって文脈自由文法のパーザが推論される. この方法によって得られるアルゴリズムは，未知の文法 G_U と構造的等価な文法 G' (すなわち，$K(D(G'))=K(D(G_U))$) を推論する. したがってそれから合成されるパーザは，未知の言語を正しく認識するだけでなく，その言語の各文にたいして正しく構造を割り当てる.

4.4　文法学習システム 1 - Full-LESSON

この理論に基づき我々は，未知の文法の構造記述に関する二種類の質問から文脈自由文法全体のクラスを効率良く学習するアルゴリズムを開発した [Sak88b]. その学習アルゴリズムを簡単に紹介する. ここで紹介されるアルゴリズムは基本的には [Ang87] の木オートマトンへの拡張である.

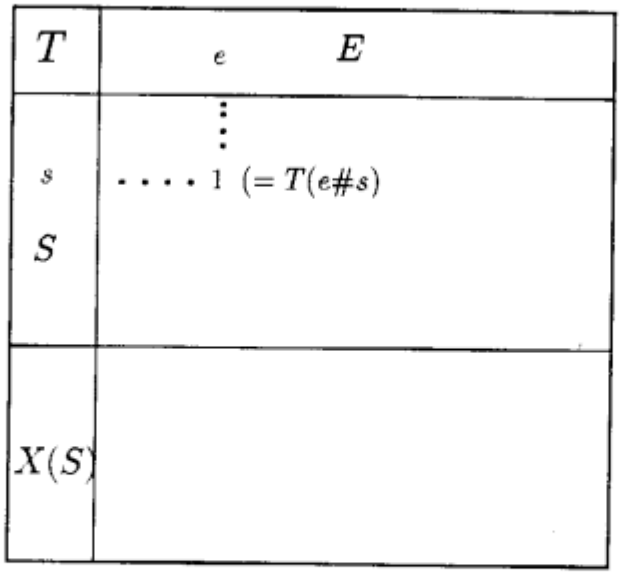

図 4.4: 観測行列 (S, E, T)

アルゴリズムは, みっつの要素からなる観測行列 (S, E, T) を構築する. ここで, S は深さ 1 以上の構造記述の集合, $X(S) = \{\sigma(u_1, \ldots, u_k) \mid \sigma \in Sk_k,\ u_1, \ldots, u_k \in S \cup \Sigma,\ \sigma(u_1, \ldots, u_k) \notin S\}$, そして$E$ は \$ をちょうどひとつ含む構造記述の集合, である. 観測行列は行に $S \cup X(S)$ の要素, 列に E の要素がラベル付けされており, 観測行列 T の要素は 0 または 1 であり, T に含まれるデータは, 行の値 s, 列の値 e にたいして $T(e\#s) \in \{0,1\}$ と定義される. ここで, 演算 # は, e 中の \$ とラベル付けされたノードに s を代入する演算であり, 例えば $e = \sigma(\$, a), s = \sigma(b)$ のとき $e\#s = \sigma(\sigma(b), a)$ となる. そして T の解釈は, $e\#s$ が未知の文法の構造記述であるとき $T(e\#s) = 1$, すなわち行のラベル s, 列のラベル e の要素が 1, 構造記述でないときは $T(e\#s) = 0$, と与えられる.

$s \in S \cup X(S)$ にたいして, $row(s)$ は $f(e) = T(e\#s)$ で定義される E から $\{0,1\}$ への有限関数 f を表す.

さてアルゴリズムは木オートマトンを構築するのに最終的にこの観測行列を用いる. S の要素がラベル付けされた行は構築される木オートマトンの状態の候補であり, E の要素がラベル付けされた列はこれらの状態を区別するために用いられる. $X(S)$ の要素がラベル付けされた行は遷移関数を構築するために用いられる.

定義　$X(S)$ の任意の要素 x にたいして $row(x) = row(s)$ なる S の要素 s が存在するとき, その観測行列は**閉じている** (closed) と言う. また観測行列が**無矛盾** (consistent) であるとは, s_1 と s_2 が $row(s_1) = row(s_2)$ であるような S の任意の要素であるときは必ず, 任意の $\sigma \in Sk$, $u_1, \ldots, u_{k-1} \in S \cup \Sigma$ と $1 \leq i \leq k$ にたいして,

$$row(\sigma(u_1, \ldots, u_{i-1}, s_1, u_i, \ldots, u_{k-1})) = row(\sigma(u_1, \ldots, u_{i-1}, s_2, u_i, \ldots, u_{k-1}))$$

が成り立つときを言う.

定義　(S, E, T) が閉じた, 無矛盾な観測行列であるならば, それから対応する木オートマトン $A(S, E, T) = (Q, Sk \cup \Sigma, \delta, F)$ を次のように構成できる:

$$Q = \{row(s) \mid s \in S\},$$

$$F = \{row(s) \mid s \in S \text{ and } T(s) = 1\},$$

$$\delta_k(\sigma, row(s_1), \ldots, row(s_k)) = row(\sigma(s_1, \ldots, s_k)) \quad \text{ここに } s_1, \ldots, s_k \in S \cup \Sigma,$$

$$\delta_0(a) = a \qquad \text{ここに } a \in \Sigma.$$

このとき, 関数 row は $a \in \Sigma$ にたいして $row(a) = a$ と拡張される.

学習アルゴリズム **LA** が, 以下に与えられる. **LA** への入力は, 構造記述 s を与えると, s が未知の文法 G_U の構造記述か否かにより yes または no の答が得られる**構造的メンバーシップ質問** (structural membership queries) と, 文脈自由文法 G を与えると, その文法が未知の文法 G_U と構造的に等価 (すなわち, $K(D(G)) = K(D(G_U))$) であるか否かにより yes または no (この時には, 反例 (counter-example) も一緒に返す) の答が得られる**構造的等価質問** (structural equivalence queries), に答える教師である. 出力は, 未知の文法 G_U と構造的に等価な文脈自由文法 G である.

アルゴリズム **LA** の重要な性質が次の定理によって与えられる.

定理 4.1 未知の文法を G_U とすると, アルゴリズム **LA** は 必ず停止し, G_U と構造的に等価な文脈自由文法を出力する. さらに, n を G_U の構造記述の集合 $K(D(G_U))$ を受理する最小状態木オートマトンの状態数, m をアルゴリズムの実行中に与えられる反例の最大のサイズとするならば, アルゴリズム **LA** の実行時間は m と n の多項式で押さえられる.

4.5　文法学習システム 2 - LESSON

次に我々は, 正だけの例から文脈自由文法の部分クラス (リバーシブル文脈自由文法) を効率良く学習するアルゴリズムを開発した [Sak88a]. それらのアルゴリズムは入力される例のサイズに関して多項式時間アルゴリズムであることが示されている. またそれらのアルゴリズムを用いたシステムは, 第五世代プロジェクトにおける逐次型推論マシン PSI 上に LESSON という名前で実現されている.

さてこのシステム LESSON の目的は, 帰納的推論メカニズムに基づいた構文的知識 (文法) の獲得を支援することにある.

この文法学習支援システムは次の特徴を持つ:

1. 構造的データからの学習

文法学習アルゴリズム LA

S:=ϕ; E:=$\{\$\}$;

G:= 文脈自由文法 $G = (\{S\}, \Sigma, \phi, S)$;

G を推測とし G にたいする構造的等価質問を行う;

If 答が *yes* **then** G を出力し, 停止する;

反例 t とその部分木で深さが 1 以上のものすべてを S に加える;

構造的メンバーシップ質問を使って, 最初の観測行列 (S, E, T) を構築する;

Repeat

While (S, E, T) が閉じていないまたは無矛盾でない;

If (S, E, T) が無矛盾でない **then**

次のような S 中の s_1 と s_2, $e \in E$, $u_1, \ldots, u_{k-1} \in S \cup \Sigma$, そして $i \in N$ を見つける :

$row(s_1) = row(s_2)$ かつ

$T(e\#\sigma(u_1, \ldots, u_{i-1}, s_1, u_i, \ldots, u_{k-1})) \neq T(e\#\sigma(u_1, \ldots, u_{i-1}, s_2, u_i, \ldots, u_{k-1}))$;

$e\#\sigma(u_1, \ldots, u_{i-1}, \$, u_i, \ldots, u_{k-1}))$ を E に加える;

構造的メンバーシップ質問を使って, 観測行列 T を $E\#(S \cup X(S))$ まで拡張する;

If (S, E, T) が閉じていない **then**

$row(s_1)$ はすべての $s \in S$ にたいする $row(s)$ と異なる, という $s_1 \in X(S)$ を見つける ;

s_1 を S に加える;

構造的メンバーシップ質問を使って, 観測行列 T を $E\#(S \cup X(S))$ まで拡張する;

観測行列 (S, E, T) が閉じて無矛盾となったならば, G:=$G(A(S, E, T))$ とする;

G を推測とし G にたいする構造的等価質問を行う;

If 答が反例 t で *no* である **then**

反例 t とその部分木で深さが 1 以上のものすべてを S に加える;

構造的メンバーシップ質問を使って, 観測行列 T を $E\#(S \cup X(S))$ まで拡張する;

Until 答が推測 G にたいして *yes*;

G を出力し, 停止する.

図 4.5: Full-LESSON の学習アルゴリズム

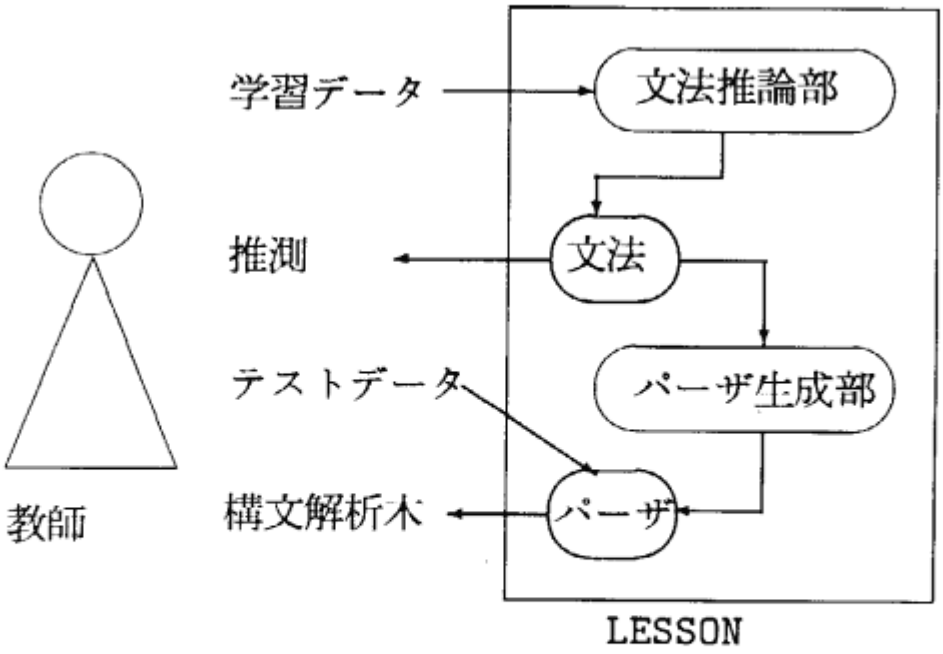

図 4.6: LESSON のシステム構成図

2. 正の例だけからの学習

3. 多項式時間学習

さらにこのシステムは, 学習された文法からパーザを自動合成する機能を持つ.

LESSON のシステム構成図を 図 4.6 に示す. (これらについてのより詳細な説明が以下で与えられる.)

我々の目標は, ボトムアップパーザに適した文法を構築するための文法推論のメカニズムを用いたシステムを提供することである. 我々は構造記述の正の例だけから文脈自由文法を学習するシステムを提案する. 文脈自由文法の構造記述とは前述したように, その文法のラベルなしの構文解析木, すなわち構文解析木の形, である. このように学習システムへの入力は, 有限個の構文解析木の形となる. 我々のシステムは, 出力される文法は意図した構造を持ち, そして正の例だけから効率良く文法を学習するという好ましい特徴を持つ.

学習された文法は教師であるユーザが意図したとおりの構造を持つ

未知の文法と構造的に等価な文法を学習するためには, その文法の構造に関する情報が学習システムに利用可能でなければならない. 我々のシステムは, 構造記述の例から未知の文法と構造的に等価な文法を学習する.

文法は正の例だけから学習される

文法に関する正の例だけを与える教師を仮定することは, 実際的な使用において妥当であるが,

完全な (正と負の) 例を与える教師を仮定することは, 教師であるユーザにとって大きな負担となる. 我々は, 構造的な正の例だけから学習可能な, リバーシブル文脈自由文法と呼ばれる文脈自由文法の部分クラスを定義する.

文脈自由文法 $G = (N, \Sigma, \Pi, S)$ が**インバーティブル**であるとは, Π 中に $A \to \alpha$ と $B \to \alpha$ なる形のふたつの生成規則があるならば, $A = B$ であるときを言う. 文脈自由文法 G が**リセットフリー**であるとは, 任意のふたつの非終端記号 B と C, それから $\alpha, \beta \in (N \cup \Sigma)^*$ にたいして, Π 中に $A \to \alpha B \beta$ と $A \to \alpha C \beta$ なる形のふたつの生成規則があるならば, $B = C$ であるときを言う. 文脈自由文法 G が**リバーシブル**であるとは, G がインバーティブルでかつリセットフリーであるときを言う.

文法は効率的に学習される

文法推論の実際的使用において最も重要な問題点は, 学習システムの実行時間の効率性である. 学習システムの効率性を評価するひとつの基準は, 多項式時間学習である. 我々の学習システムは, 多項式時間学習を達成している.

最後にこのシステム LESSON が基本に用いる学習アルゴリズムを示す.

定理 4.2 学習アルゴリズム **RC** は, 未知の文法の構造記述の正の例が与えられた時に, 未知の文法と構造的に等価な文法を極限において同定し, また新しい例が入力されてから次の推測を出力するまでに入力例のサイズに関して多項式時間で実行する.

文法学習アルゴリズム RC

入力: 構造記述の有限集合 Sa;

出力: リバーシブル文脈自由文法 G;

手続き:

RC は初めにSa から, 次に定義される基本となる文脈自由文法 $G_0 = (N_0, \Sigma, \Pi_0, S_0)$ を作る:

$$N_0 = (Sub(Sa) - \Sigma) \cup \{S_0\},$$
$$\Pi_0 = \{\sigma(A_1, \ldots, A_k) \rightarrow A_1 \cdots A_k\}$$
$$\cup \{S_0 \rightarrow A_1 \cdots A_k \mid \sigma(A_1, \ldots, A_k) \in Sa\}$$

次に RC は次のいずれかの条件が成り立つ時に,

任意の異なるふたつの非終端記号 A と B を繰り返しマージする :

1) $A \rightarrow \alpha$ と $B \rightarrow \alpha$ $\quad (\alpha \in (N \cup \Sigma)^*)$

なる形のふたつの生成規則が存在する

または

2) $C \rightarrow \alpha A \beta$ と $C \rightarrow \alpha B \beta$ $\quad (\alpha, \beta \in (N \cup \Sigma)^*)$

なる形のふたつの生成規則が存在する

このような非終端記号の組がなくなったら,

結果として生じた文法を G として出力し,

停止する.

図 4.7: LESSON の学習アルゴリズム

第 5 章

学習の形式モデルとしての文法推論

学習の研究は認知科学の中心的かつ重要な研究課題のひとつである. 従来の研究の関心は主に認識モデルにあり, 内観法で構築されたモデルにたいして, 観察・実験や計算機によるシミュレーションによって検証が行われる. その際, モデルは形式的なものではなく, モデル自体にあいまい性があることが多く, モデルから得られる帰結が無意味である場合がある.

一方, 計算機科学では帰納的推論に関する興味深い結果が数多く得られている [AS83]. モデルは形式的に定義され, 推論問題に関するさまざまな帰結が論理的に得られている. しかし, アルゴリズムの停止性や計算の効率といった理論的側面に関心が多く, 認識モデルにたいする関心はあまりない. また, 心理学や教育学などの経験科学への貢献にたいする関心もほとんどない.

Pylyshyn は [Pyl86] で, "認知" を "記号計算" と定義し, 認知科学がこれらふたつの科学のどちらの側面をも持つことを主張した. 彼の主張に従えば, 計算機科学での帰納的推論の結果は, 経験科学による人間の学習研究にとって, 重要な意義をもつ.

本章では, 文法推論を学習の形式モデルとしたときに, モデルから得られる論理的帰結の意義を認知科学の側面から考察する. 一般に, 学習と帰納的推論は区別される. それゆえ, 計算機科学における帰納的推論の理論を学習全体の形式モデルとすることはできない. しかし, 学習において帰納的推論が用いられていることは明らかであろう. したがって, 計算機科学の理論を形式モデルとすることで, 帰納的推論の機能に関してさまざまな帰結を論理的に得ることができる.

特にここでは, 線型言語の文法推論をモデルとする. 線型文法は記号列変換機能の形式モデルとなることが可能である. 記号列変換機能は人間の認知過程で重要な役割を演ずる.

まず, 制御集合にもとづく線型言語の文法推論の方法とその方法の性質を述べる. この結果にもとづき, 記号列変換機能の帰納的学習において

- 学習者が学習の達成, つまり, 変換機能を学習したことを認知できるか,
- 学習者が課題の難易度をどのように認知するか,

を, 変換機能のクラスと教師の能力に関する思考実験によって明らかにする. そして, これより, 記号列変換機能の学習における理想的な問題設定を示し, 教師に必要な能力を示す.

5.1 制御集合にもとづく線型言語の文法推論

線型言語のクラスは正則集合のクラスを真に含み，文脈自由言語のクラスに真に含まれる．線型言語のクラスにたいする推論アルゴリズムの研究として，Biermann [Bic71], Radhakrishnan and Nagaraja [RN88], Tanatsugu [Tan87] などの研究がある．これらのアルゴリズムは文法の自己埋め込み性にもとづいて言語を同定するが，記号列の中の自己埋め込み性を発見する手続きは効率が悪い．

そこで，我々は制御集合にもとづく文法推論の方法を提案する．まず，線型言語に関する表現定理を示す．この表現定理により，ある固定された線型文法の生成規則をアルファベットとする正則集合を同定することが，線型言語を同定することに対応する．

5.1.1 表現定理

始めに，アルファベット Σ 上の任意の線型言語 L はある固定された線型文法 G と正則な制御集合 C によって生成されることを示す．

定義 **線型文法** $G = (N, \Sigma, \Pi, S)$ とは

$$A \to uBv,\ A \to u$$

の形をした生成規則だけからなる文脈自由文法である．ここで，$A, B \in N$, $u, v \in \Sigma^*$.

線型文法 G によって生成される言語 $L = L(G)$ を**線型言語**という．

定義 Σ をアルファベットとする．このとき，生成規則の集合 Π^0 が

$$\begin{aligned} \Pi^0 \quad = \quad & \{S^0 \to aS^0 \mid a \in \Sigma\} \\ \cup \ & \{S^0 \to S^0a \mid a \in \Sigma\} \\ \cup \ & \{S^0 \to a \mid a \in \Sigma\} \\ \cup \ & \{S^0 \to \lambda\} \end{aligned}$$

からなる線型文法 $G^0 = (\{S^0\}, \Sigma, \Pi^0, S^0)$ を**万能** (universal) という．

アルファベット Σ にたいして，万能線型文法 G^0 はただひとつ存在し，$\Sigma^* = L(G^0)$ である．

定義 $G = (N, \Sigma, \Pi, S)$ を線型文法とする．このとき，Π^* の部分集合 C を G にたいする**制御集合** (control set) という．また，言語

$$L_C(G) = \{w \in \Sigma^* \mid S \overset{\alpha}{\underset{G}{\Rightarrow}} w,\ \alpha \in C\}$$

を制御集合 C とともに文法 G によって生成された言語という．

$G=(N,\Sigma,\Pi,S)$ を線型文法, $G^0=(\{S^0\},\Sigma,\Pi^0,S^0)$ を万能線型文法とする. Π^*からΠ^{0*} への準同型写像 h を次のように定義する:

$$h(\pi)=\begin{cases}\pi^0 & \text{ただし } \pi^0:S^0\to\lambda,\ \pi:S\to\lambda \text{ のとき},\\ \pi_i^0 & \text{ただし } \pi_i^0:S^0\to a,\ \pi:A\to a \text{ のとき},\\ \pi_j^0 & \text{ただし } \pi_j^0:S^0\to aS^0,\ \pi:A\to aB \text{ のとき},\\ \pi_k^0 & \text{ただし } \pi_k^0:S^0\to S^0a,\ \pi:A\to Ba \text{ のとき}.\end{cases}$$

G に対応する非決定性有限オートマトン $M=(K,\Pi,\delta,S,F)$ を次のように定義する:

- $K=N\cup\{q_F\}$ ただし $q_F\notin N$,

$$\delta(S,\pi^0)=\begin{cases}\{q_F\} & \pi^0:S^0\to\lambda,\ \pi\in h^{-1}(\pi^0) \text{ かつ } \pi:S\to\lambda \text{ のとき}\\ \emptyset & \pi^0:S^0\to\lambda \text{ かつ } h^{-1}(\pi^0)=\emptyset \text{ のとき}\end{cases}$$

- $$\delta(A,\pi_i^0)=\begin{cases}\{q_F\} & \pi_i^0:S^0\to a,\ \pi\in h^{-1}(\pi_i^0) \text{ かつ } A\to a \text{ のとき}\\ \emptyset & \pi_i^0:S^0\to a \text{ かつ } h^{-1}(\pi_i^0)=\emptyset \text{ のとき}\end{cases}$$

$$\delta(A,\pi_j^0)=\begin{cases}\{B\,|\,\pi:A\to aB\in\Pi,\ \pi\in h^{-1}(\pi_j^0)\} & \pi_j^0:S^0\to aS^0 \text{ のとき}\\ \{B\,|\,\pi:A\to Ba\in\Pi,\ \pi\in h^{-1}(\pi_j^0)\} & \pi_j^0:S^0\to S^0a \text{ のとき},\end{cases}$$

- $F=\{q_F\}$。

線型文法 G, 万能線型文法 G^0 と, G に対応する非決定性有限オートマトン M に関して次の補題が成立する.

補題 5.1 任意の $w\in\Sigma^*$, $A\in N$, $\alpha\in\Pi^*$ にたいして, $A\overset{\alpha}{\underset{G}{\Longrightarrow}}w$ である必要十分条件は $S^0\overset{h(\alpha)}{\underset{G^0}{\Longrightarrow}}w$ かつ $\delta(A,h(\alpha))\ni q_F$ である.

補題 5.1 から線型言語に関する次のふたつの表現定理を得る.

定理 5.1 $G^0=(\{S^0\},\Sigma,\Pi^0,S^0)$ を万能線型文法とする. このとき, アルファベット Σ 上の任意の線型言語 L にたいして $L=L_C(G^0)$ となる正則制御集合 C が存在する.

定理 5.2 $G^0=(\{S^0\},\Sigma,\Pi^0,S^0)$ を万能線型文法, C を G^0 にたいする正則制御集合とする. このとき, $L=L_C(G^0)$ は線型言語である.

表現定理 5.1, 5.2 から, 万能線型文法にたいする正則制御集合を推論することによって, 任意の線型言語を推論することができる.

5.1.2 線型言語の文法推論

一般に, 線型言語の文法推論問題を正則集合の推論問題に還元することはできない. なぜならば, 任意の線型言語にたいして, ただひとつの正則制御集合を構成的に求めることはできないからである. 線型言語 L を生成する線型文法は無限にあるので, 制御集合

$$C=\{\alpha\,|\,S^0\overset{\alpha^0}{\underset{G^0}{\Longrightarrow}}w,\ w\in L\}=\bigcup_{L(G)=L}h(\{\alpha\,|\,S\overset{\alpha}{\underset{G}{\Longrightarrow}}w,\ w\in L\})$$

は正則制御集合の無限和であり，C が正則であるかどうかは一般には判らない．しかし，線型文法に関する何らかの補助情報を用いることで、任意の線型言語にたいしてただひとつの正則制御集合を定義することができる．この補助情報には，文法の非終端記号の個数の上限や文法の代表標本 (representative sample) などが考えられる．

文法の非終端記号の個数の上限を設定すると，任意の線型言語 L にたいして，L を生成する簡約された線型文法，つまり，無用な非終端記号と生成規則のない線型文法の個数は有限になる．したがって，上記の制御集合 C は正則制御集合の有限和となり，任意の線型言語にたいしてただひとつ存在する．

線型文法 G の代表標本は次のように定義される．

定義　$G=(N,\Sigma,\Pi,S)$ を簡約された線型文法とする．G の**代表標本**とは，G の任意の生成規則 π にたいして，$S \overset{\alpha\pi\beta}{\underset{G}{\Longrightarrow}} w$ である記号列 w を少なくともひとつ含む $L(G)$ の有限な部分集合である．

RS を L のある有限部分集合とする．L を生成しかつ，RS を代表標本とする非終端記号の数が最小の線型文法が存在するならば，その個数は有限である．したがって，任意の線型言語 L にたいして，L を生成する線型文法の代表標本を補助情報としても，ただひとつの正則制御集合を定義することができる．

このように，文法の非終端記号の個数の上限や文法の代表標本を補助情報として利用可能な場合，線型言語の文法推論問題を正則集合の推論問題に還元することが可能になる．ただし，推論アルゴリズムは与えられた補助情報が示す条件を満たすような正則制御集合を推論しなければならない．また，その正則制御集合は構成的には定義されない．したがって，自然な問題設定においては，線型言語の推論アルゴリズムは極限においてのみ言語を同定するにすぎないことが予想される．

さらに，推論に要する時間計算量は指数関数のオーダになることが予想される．任意の記号列 w にたいして，万能線型文法 G^0 において開始記号 S^0 から w を得る導出は $2^{lg(w)}$個ある．したがって，素朴なアルゴリズムでは，推論に要する時間計算量は $lg(w)$ の指数関数のオーダになる．

推論の対象を線型言語のサブクラスに限ると，そのクラスの言語の文法推論問題を正則集合の推論問題に還元することが可能になる．次に，還元可能なふたつのサブクラスを示す．

そのひとつは線型文法の**かっこ文法**のクラスである．

定義　$G=(N,\Sigma,\Pi,S)$ を線型文法とする．G の**かっこ文法** $[G]$は$[G]=(N,\Sigma\cup\{[,]\},\Pi',S)$ で表され，Π' は Π の各要素 $A\to x$ を $A\to[x]$ で置き換えることによって得られる．ここで，'[' と ']' は Σ の要素でない特別の記号である．

線型文法 G のかっこ文法 $[G]$ によって生成される言語 $L_p=L([G])$ をかっこ付き線型言語 (linear language with parenthesis) という．

かっこ付き線型言語に関しても，線型言語の場合と同様の表現定理が成立する．

定理 5.3 $[G]^0 = (\{S^0\}, \Sigma, \Pi^0, S^0)$ を非終端記号をひとつだけしかもたない線型文法のかっこ文法, C を Π^0 上の正則制御集合とする. このとき, 言語 $L_C([G]^0)$ はかっこ付き線型言語である.

第2のサブクラスは **even 線型言語**のクラス [AP64] である.

定義　$A \to uBv$ の形をした任意の生成規則が $lg(u) = lg(v)$ の条件を満たす線型文法 G_e を **even 線型文法**という.

even 線型文法によって生成される言語 $L_e = L(G_e)$ を **even 線型言語**という.

even 線型言語に関しても, 線型言語の場合と同様の表現定理が成立する. その際, 万能 even 線型文法 $G_e^0 = (\{S^0\}, \Sigma, \Pi^0, S^0)$ は次のような生成規則を持つ:

$$\begin{aligned} \Pi^0 \;=\; & \{S^0 \to aS^0b \,|\, a, b \in \Sigma\} \\ \cup\; & \{S^0 \to ab \,|\, a, b \in \Sigma\} \\ \cup\; & \{S^0 \to a \,|\, a \in \Sigma\} \\ \cup\; & \{S^0 \to \lambda\} \end{aligned}$$

定理 5.4 $G_e^0 = (\{S^0\}, \Sigma, \Pi^0, S^0)$ を万能 even 線型文法とする. このとき, アルファベット Σ 上の任意の言語 L_e が even 線型言語である必要十分条件は $L_e = L_C(G_e^0)$ となる正則制御集合 C が存在することである.

さらに, かっこ付き線型言語, even 線型言語にたいして, ただひとつの正則制御集合を構成的に定義することができる.

定義　L_p をかっこ付き線型言語, $[G]^0 = (\{S^0\}, \Sigma, \Pi^0, S^0)$ を非終端記号をひとつだけしかもたない線型文法のかっこ文法とし, $L_p = L_C([G]^0)$ である正則制御集合が存在するとする. このとき, C が次のふたつの条件を満たとき, C を**正準** (canonical) であるという:

1. $\pi^0 : S^0 \to \lambda$ にたいして, $\alpha\pi^0\alpha' \in C$ ならば, $\alpha\alpha' = \lambda$,
2. 任意の $\alpha^0 \in C$ にたいして, $S^0 \underset{[G]^0}{\overset{\alpha^0}{\Longrightarrow}} w$ かつ $w \in L$.

even 線型言語の場合も, 同様に万能 even 線型文法にたいして正準な制御集合を定義する.

この正準な制御集合は言語にたいして unique でありかつ, 正則である. したがって, かっこ付き線型言語, even 線型言語の文法推論問題は, 正準な正則制御集合の推論問題に還元可能である [Tak89]. それゆえ, かっこ付き線型言語 L_p と even 線型言語 L_e の文法推論問題に関して次の定理が成立する.

定理 5.5 ([Tak89]、[Tak88])　かっこ付き線型言語 L_p と even 線型言語 L_e の文法推論問題は正則集合の推論問題に還元される。

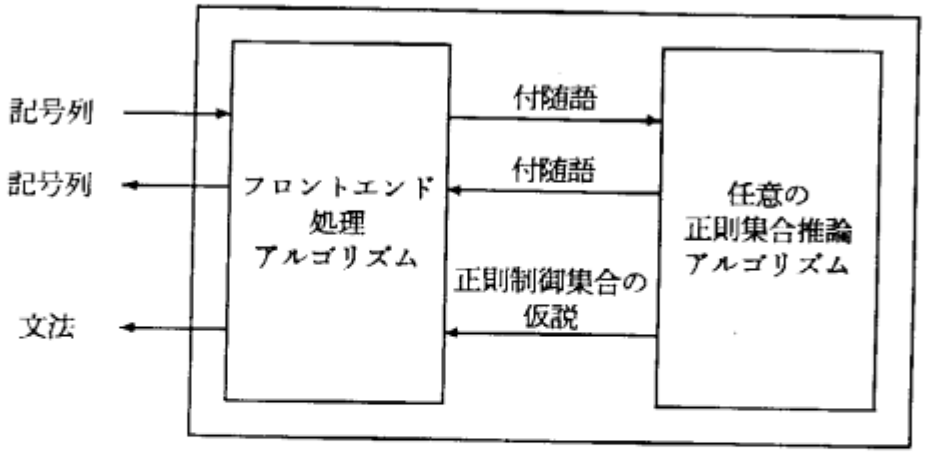

図 5.1: かっこ付き線型言語 L_p や even 線型言語 L_e を推論するアルゴリズム

かっこ付き線型言語 L_p と even 線型言語 L_e を推論するアルゴリズムを構築するには, フロントエンド処理アルゴリズムを用意して, それを正則集合推論アルゴリズムに付け加えればよい. 一般に, 言語を推論するアルゴリズムは記号列を入力, 出力とし, 仮説である文法を出力とする. このとき, 正則集合を推論するアルゴリズムは万能線型文法の付随語を入力, 出力とし, 正則制御集合の仮説を出力とする. そこで, フロントエンド処理アルゴリズムは次の処理をすればよい:

- 入力記号列を万能線型文法 G^0 において構文解析し, G^0 の付随語を求め, それを正則集合推論アルゴリズムに渡す.
- 正則集合推論アルゴリズムの出力である付随語から記号列を生成し, 出力する.
- 正則制御集合の仮説を文法に変換し, 出力する.

この模様を図 5.1 に示す.

正則集合の文法推論問題はよく研究されており, 多くの実用的なアルゴリズムが提案されている [Ang87,Ang81,Bie72,Ish87]. なかでも, Angluin [Ang87] のアルゴリズムは時間計算量の点で現在最も優れたアルゴリズムである. このアルゴリズムが推論に要する時間計算量は, 推論すべき正則集合を受理する最小状態決定性有限オートマトンの状態数を m, 反例の最大長を n とすると, m と n の多項式のオーダである. さらに, フロントエンド処理アルゴリズムの時間計算量も m の多項式のオーダである. したがって, このアルゴリズムを用いて L_p や L_e を推論するアルゴリズムを構築した場合, 推論に要する時間計算量は推論すべき言語を生成する文法が必要とする非終端記号の最小数を s, 反例の最大長を t とすると, s と t の多項式のオーダになる.

かっこ付き線型言語 L_p, even 線型言語 L_e はどちらも次のふたつの性質をもつ:

表 5.1: 推論アルゴリズムと言語の性質

	線型言語 L	かっこ付き線型言語 L_p	even 線型言語 L_e
正則集合推論問題への還元可能性	補助情報が必要	還元可能	還元可能
時間計算量	(指数関数)	多項式	多項式
同定の基準	(極限同定のみ)	有限同定が可	有限同定が可
無あいまいな文法	求められない	求められる	求められる
言語の同値性判定	決定不能	決定可能	決定可能

1. 各言語にたいする万能線型文法は無あいまいである. さらに, 任意の言語にたいして, それを生成する文法から無あいまいかつ非終端記号の数が最小の文法を構成的に求めることが可能である.

2. 言語の同値性判定問題が可解である.

正則制御集合を受理する決定性有限オートマトンが対応する文法 $[G]$ や G_e は無あいまいである. また, 任意の言語 L_p や L_e にたいして, ただひとつの正則制御集合が構成的に定義されるので, 言語の同値性判定問題が正則制御集合の同値性判定問題に還元される. したがって, 表現定理から L_p と L_e の性質 1, 2 が導かれる.

線型言語, かっこ付き線型言語, even 線型言語のクラスに関する性質と制御集合にもとづく文法推論に関する性質を, 表 5.1 にまとめる. 有限同定が可能というのは, そのクラスの言語を有限の計算で同定する推論アルゴリズムが存在することを意味する. ただし, 有限の計算で同定するには, 補助情報を与えるか, 言語の同値性に関する教師が必要である. 表 5.1 が示すように, 言語の性質と推論アルゴリズムの性質には密接な関係がある.

5.2 記号列変換機能の学習に必要な教師の能力

あるアルファベットの記号列を別のアルファベットの記号列に変換する機能は, 計算機や人間などの情報処理システムの基本的な機能のひとつである. 我々人間は、馴染みのない物の数を認知するのに、具体的な馴染みやすい物に対応させると認知しやすくなる。この対応づけは, 別の見方をすれば, 記号列変換機能である.

たとえば, 我々日本人にとってはローマ数字の表す数は理解しにくいが, ローマ数字を算術式に変換すれば理解しやすい. 変換した記号列を算術式と見て, 演算を実行すれば, 容易にローマ数字を理解できる. また, 計算機ではローマ数字を算術式に変換し, その数字の表す数を計算する. 具体的には, ローマ数字 XVI の表す数は算術式 $10+5+1$ に変換すると, 容易に計算できるし理解しやすい. この場合には, アルファベット $\{I, V, X, L, C, D, M\}$ 上の記号列 (ローマ数字) をアル

ファベット$\{1,5,10,50,100,500,1000,+,-\}$上の記号列(算術式)に変換する機能が重要な役割を演じている.

では,情報処理システムがこのような記号列変換機能を帰納的に学習するには,どのような情報が必要となるのか.つまり,どのような情報が与えられたならば,システムはどのような機能をどのように学習できるのか.これらの質問にたいする答えは,学習者にとって理想的な学習の設定を規定し,さらには,学習するシステムに情報を与えるもの,つまり,教師に必要な能力を規定する.また,記号列変換機能のCAI (Computer Aided Instruction) システムに必要な能力も規定される.

ここでは,前節で述べた線型言語の文法推論の結果にもとづいて,記号列変換機能の帰納的学習において

- 学習者が学習の達成,つまり,変換機能を学習したことを認知できるか,
- 学習者が課題の難易度をどのように認知するか,

を,変換機能のクラスと教師の能力に関する思考実験によって明らかにする.そして,この実験結果にもとづき,教師に必要な能力を示す.

5.2.1 記号列変換機能

形式的には,学習者の変換機能を6項組$T=(K,\Sigma,\Delta,\mu,q_0,F)$で定義する.$K$は状態の空でない有限集合,$\Sigma$は入力アルファベット,$\Delta$は出力アルファベット,$\mu$は$K\times\Sigma^*\times\Delta^*\times K$の有限な部分集合,$q_0$は初期状態,$F$は最終状態の集合である.また,$K\times\Sigma^*\times\Delta^*$上の関係$\longmapsto$を,$\Sigma^*$の各要素$w$にたいして,$(p,x,y,q)\in\mu$のとき,$(p,xw,z_1)\longmapsto(q,w,z_1y)$と定義する.関係$\overset{*}{\longmapsto}$は,$\longmapsto$の反射推移閉包である.変換機能$T$が変換可能な入力記号列と,それに対応する出力記号列のペアの集合$M(T)$は,

$$M(T)=\{(w_i,w_o)\mid(q_0,w_i,\lambda)\overset{*}{\longmapsto}(q,\lambda,w_o),\ q\in F\}$$

である.ここでは,学習の対象である変換機能を次のみっつのクラスT_e, T_s, Tに分ける:

- 任意の$T_e\in\mathcal{T}_e$において,μは$K\times\Sigma\times\Delta\times K$の有限な部分集合である.
- 任意の$T_s\in\mathcal{T}_s$において,μは$K\times\Sigma^*\#\times\Delta^*\#\times K$の有限な部分集合である.ただし,記号$\#$は$\Sigma$にも$\Delta$にも含まれないものとする.
- 任意の$T\in\mathcal{T}$において,μは$K\times\Sigma^*\times\Delta^*\times K$の有限な部分集合である.

クラス$\mathcal{T}_e$の変換機能はひとつの記号を必ずひとつの記号に変換する.クラス$\mathcal{T}_s$の変換機能では,区切り記号$\#$によって変換記号列中に変換の単位が明示されている.$\mathcal{T}$は変換機能全体からなるクラスである.

記号列変換機能は線型文法と密接な関係がある. ここでは, 前節で述べた線型言語の文法推論を学習の形式モデルとして用いる. 変換機能 T_e, T_s, T はそれぞれ, even 線型文法 G_e, かっこ線型文法 G_s, 線型文法 G に対応する.

$T = (K, \Sigma, \Delta, \mu, q_0, F)$ を変換機能とする. T より線型文法 $G = (K, \Sigma \cup \Delta \cup \{\#\}, \Pi, q_0)$ を構成する. Π は, 次のように定義される:

- $(p, x, y, q) \in \mu$ のとき, $p \to xqy^R \in \Pi$,
- $q \in F$ ならば, $q \to \# \in \Pi$.

ここで, w^R は記号列 w の逆語を表す.

命題 5.1 任意の記号列のペア $(x, y) \in \Sigma^* \times \Delta^*$ にたいして, $(x, y) \in M(T)$ である必要十分条件は, $x\#y^R \in L(G)$ である.

命題 5.1 の逆も成り立つ. 同様に, 変換機能 T_e, T_s から, それぞれ, even 線型文法, かっこ文法を構成することができる.

5.2.2 学習と思考実験の設定

学習者はある記号列変換機能を学習しなければならない. その際, 教師がいて, 教師はその変換機能を知っている. 学習者は教師から変換の具体例を入出力記号列のペア (w_i, w_o) で与えられ, 具体例の集合から変換機能を帰納的に学習する. また, 学習の過程で, 学習者は教師に“質問”することも許されている. 教師は質問に常に正しく“応答”する.

学習者が許される質問は次のみっつである：

- ある入出力ペア (w_i, w_o) が正しい変換, つまり, $(w_i, w_o) \in M(T)$ かどうかに関する質問.
- 変換機能の補助情報に関する質問（補助情報の要求）.
- 変換機能の同値性に関する質問.

学習者からある入出力ペアが正しい変換であるかどうかを質問された場合, 正しければ教師は“はい”と答え, そうでないならば“いいえ”と答える. 補助情報が求められた場合には, 教師は自分自身が想定する変換機能の代表標本を学習者に与える. ここで, 変換機能の代表標本は, 線型文法のものと同様に定義される. 変換機能の同値性に関する質問は, 学習者の仮説である変換機能 T_H が学習の対象である変換機能 T と同じ能力, つまり $M(T_H) = M(T)$ かどうかという形で行なわれる. もし $M(T_H) = M(T)$ であるならば, 教師は“はい”と答え, そうでないならば反例として $M(T)$ と $M(T_H)$ の対称差の要素をひとつ学習者に与える.

答えうる質問の種類によって教師の能力は異なる. ここでは, 次のみっつの能力をもつ教師を仮定する：

$\mathcal{A}_M$: 正しい変換であるかどうかに関する質問のみに答える能力.

$\mathcal{A}_A$: 正しい変換であるかどうかに関する質問と補助情報の要求に答える能力.

$\mathcal{A}_E$: 正しい変換であるかどうかに関する質問と変換機能の同値性に関する質問に答える能力.

したがって, 変換機能のクラスと教師の能力の組み合わせから, 9 種の実験を行う.

5.2.3 実験結果

学習の達成の認知

学習者の学習手続きを P とする. 学習者は P を学習の対象である変換機能 T のしだいに増加する具体例の集まりにたいして繰り返し適用し, 仮説の列 $T_{H_1}, T_{H_2}, \ldots$ を生成するとする. ある数 m が存在して, $M(T_{H_m}) = M(T)$ であり, P が T_{H_m} を出力したあと停止するならば, その学習者は有限の計算で T を正しく同定する, または単に T を**有限同定する**と言う. 一方, ある数 m が存在して $M(T_{H_m}) = M(T)$ であるが, P は T_{H_m} を出力しても停止せず, ただし $T_{H_m} = T_{H_{m+1}} = T_{H_{m+2}} = \cdots$ であるとする. このとき, 学習者は極限において T を正しく同定する, または単に T を**極限同定する**と言う.

有限同定と極限同定の違いは学習の達成の認知を決定する. 有限同定可能な学習手続きが存在する実験課題の場合, その学習手続きをもつ学習者は学習の達成を認知でき, 認知したあとは変換機能を「知っている」と確信できる. 一方, 極限同定可能な学習手続きしか存在し得ない課題の場合, どのような学習手続きをもってしても, 学習者は学習したことを決して認知できず, 永遠に変換機能を「知っている」とは確信できない.

変換機能のクラスと教師の能力の組み合わせにたいして, 存在し得る学習手続きの同定能力を表5.2 に示す. 表5.2 が示すように, 能力 $\mathcal{A}_M$ をもつ教師を仮定すると, どのクラスの変換機能でも学習者は学習の達成を認知することはない. 逆に, 能力 $\mathcal{A}_E$ をもつ教師を仮定すると, すべてのクラスの変換機能にたいして, 学習者は学習の達成を認知することができる. ただし, クラス $\mathcal{T}$ の変換機能の同値性を判定するアルゴリズムは存在しない. したがって, クラス $\mathcal{T}$ の変換機能の学習において, 能力 $\mathcal{A}_E$ を教師に仮定するのは不自然である. 能力 $\mathcal{A}_A$ をもつ教師を仮定すると, クラス $\mathcal{T}_e$, $\mathcal{T}_s$ の変換機能にたいしては学習者が認知することがあるが, クラス $\mathcal{T}$ の変換機能の学習では, 認知することはない.

課題の難易度の認知

変換記号列のペア (w_i, w_o) の長さを $lg(w_i) + lg(w_o)$ で定義する. ある仮説を出力した時点から次の仮説を出力する時点までに実行する計算ステップの上限が, 入力された変換記号列のペアの長さを引数とする多項式によって常に束縛されている学習手続き P_p を, **多項式時間計算量の学習**

表 5.2: 同定に関する変換機能と教師の能力の関係

	$\mathcal{T}_c$	$\mathcal{T}_s$	$\mathcal{T}$
$\mathcal{A}_M$	極限同定	極限同定	極限同定
$\mathcal{A}_A$	有限同定	有限同定	極限同定
$\mathcal{A}_E$	有限同定	有限同定	有限同定

表 5.3: 変換機能と時間計算量の関係

$\mathcal{T}_c$	$\mathcal{T}_s$	$\mathcal{T}$
多項式時間	多項式時間	指数関数時間

手続きと言う. また, 指数関数によって束縛されている学習手続き P_e を, **指数関数時間計算量の学習手続き**と言う.

学習手続き P_p は, 現在の計算機で効率良く計算可能であるが, 学習手続き P_e は組み合わせ爆発などを起こし, 一般にはうまく計算できない. したがって, P_p によって学習する学習者は課題を「やさしい」と認知し, P_e によって学習する学習者は「難しい」と認知する.

学習手続きの時間計算量は変換機能によって決まる. その際, 少なくとも教師は $\mathcal{A}_M$ の能力をもつと仮定する. 変換機能のクラスと学習手続きの時間計算量の関係を表5.3 に示す. 表5.3 が示すように, クラス $\mathcal{T}_c$、$\mathcal{T}_s$ の変換機能を学習する課題を与えられた場合, 学習者は学習手続き P_p を用いることが可能であり, その課題を「やさしい」と認知する. しかし, クラス $\mathcal{T}$ の変換機能を学習する課題を与えられた場合は, P_e を用いることになり, 「難しい」と認知する.

5.2.4 教師に必要な能力

学習者にとって, 学習の達成を認知でき, 課題を「やさしい」と認知するのが, 理想的な学習である. したがって, 思考実験の結果より, 学習の対象がクラス $\mathcal{T}_c$ または $\mathcal{T}_s$ の変換機能, 教師の能力が $\mathcal{A}_A$ または $\mathcal{A}_E$ である設定が, 学習者が記号列変換機能を学習する理想的な問題設定であることがわかる.

このように, 教師には, 正しい変換であるかどうかに関する質問と, 補助情報の要求または変換機能の同値性に関する質問に答える能力が必要である. この能力をもつ教師は学習者に学習の達成を認知させることが可能である. さらに, 学習の対象が一般の変換機能のクラス $\mathcal{T}$ である場合には, 変換の単位が明示されているように具体例を与え, 学習の対象がクラス $\mathcal{T}_s$ の変換機能になるようにしてやらなければならない. あるいは, アルファベットをうまく選ぶことによって, 学習の対象がクラス $\mathcal{T}_c$ の変換機能になるように工夫してもよい. そうすることによって, 学習者に変換機能を効率良く学習させることができる.

例 5.1 各ローマ数字の文字をその文字が表わす算用数字と算術記号 '+' と '−' に変換する機能を考える.

ここでは, 記号 I, V, X だけからなるローマ数字を対象とする. ローマ数字では, I が 1 に, V が 5 に, X が 10 にそれぞれ対応する. 数を構成する記号は, 対応する算用数字の値が大きい順に左から右へ並べられ, 数の値は各記号の値の加算によって得られる. もし値の小さい記号が大きい記号の左に現われたならば, 負の値をもつと見なされる. したがって, XVI は 16 で, IXX は 19 である. そこで, ローマ数字から算術式への変換機能 $T_R = (K, \Sigma, \Delta, \mu, q_p, F)$ を次のように定義する:

- $K = \{ q_p, q_{n_1}, q_{n_2} \}$,
- $\Sigma = \{ \mathrm{I}, \mathrm{V}, \mathrm{X} \}$,
- $\Delta = \{ 0, 1, 5, +, - \}$,
- $\mu = \left\{ \begin{array}{lll} (q_p, \mathrm{I}, +1, q_p), & (q_p, \mathrm{V}, +5, q_{n_1}), & (q_p, \mathrm{X}, +10, q_{n_2}), \\ (q_{n_1}, \mathrm{I}, -1, q_{n_1}), & (q_{n_1}, \mathrm{V}, +5, q_{n_1}), & (q_{n_1}, \mathrm{X}, +10, q_{n_2}), \\ (q_{n_2}, \mathrm{I}, -1, q_{n_2}), & (q_{n_2}, \mathrm{V}, -5, q_{n_2}), & (q_{n_2}, \mathrm{X}, +10, q_{n_2}) \end{array} \right\}$,
- $F = K$.

ただし, 右から左へと走査するとしたほうが定義しやすいので, ローマ数字の逆語が入力記号列として与えられるものとする. 変換機能 T_R によって, ローマ数字 XVI は算術式 $+1+5+10$ に, IXX は $+10+10-1$ に変換される. これらの算術式の値は簡単に求められる.

変換機能 T_R は, クラス $\mathcal{T}$ に属する. したがって, このままでは, 学習者は学習の達成を認知できないし, 課題を難しいと認知する. そこで, 変換単位ごとに区切り記号 # を挿入して具体例を与えてやることで, 課題をクラス $\mathcal{T}_s$ の変換機能にする. または, 出力記号列のアルファベットを $\{+1, +5, +10, -1, -5\}$ とすることで, 課題をクラス $\mathcal{T}_e$ の変換機能にする. このようにすることによって, 学習の設定を理想的なものにすることができる.

第 6 章

おわりに

機械学習の研究概況を，“言語の学習”しかも“帰納的推論に基づく言語学習”にしぼって，筆者らによる最近の結果を報告した．今後の残された課題について，簡単にふれる．

第3章では，課題1に関して論じたアルゴリズムは，(枚挙的手法によるため) 効率がよくない，という大きな短所がある．これは，(記号列からの) 教師なし学習の問題設定においては，“本質的に難しい”(NP-困難な問題である) ことが証明されている点であり，したがって，これを解決するためには，構成的手法を部分的に取り入れること，教師付きの問題設定で考察しなおすこと，等が考えられる．また，課題2に関しては，より大きな言語クラスへの学習アルゴリズムの拡張と同時に，応用領域の開発が重要な課題となる．

第4章では，文脈自由文法を効率的に学習するアルゴリズムとそれを実現したシステムについて考察されたが，構造記述からの文脈自由文法の学習という問題は属性文法の学習問題に自然に拡張できる．するとふたつの効率的文法学習アルゴリズム LA と RC は，属性文法の学習アルゴリズムに容易に拡張される可能性がある．このアルゴリズムを基本に用いることによって，例からコンパイラーを作成する (生成する) システム等を実現することができると考えている．

第5章では，そこで述べた結果は直感的には妥当な結果であるが，実際の人間の学習においてどれほど妥当なものかは，実験などで実証する必要がある．また，CAI システムの基礎理論を目指すには，そこで述べた以外の質問や，例の提示法と学習の効果などを理論的に解明する必要もある．

また，ここでふれなかった“技能 (スキル) の学習”あるいは“概念の学習”といった重要な研究テーマは，今後益々注目される研究分野となるであろう．しかし，ここで報告した“言語の学習”にたいする研究成果および研究指針 (アプローチ) は，多くの未開の領域を残したこれらの重要課題をアタックするための，貴重な提言，経験を与えてくれるものと確信する．

謝辞

本研究を行なう機会を与え，またご指導いただいた当研究所の北川敏男会長と榎本肇所長に厚く謝意を表する．

なお，本研究は第五世代コンピュータプロジェクトの一環として行なったものである．

参考文献

[Ang80] Dana Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[Ang81] Dana Angluin. A note on the number of queries needed to identify regular languages. *Information and Control*, 51:76–87, 1981.

[Ang82] Dana Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

[Ang86] Dana Angluin. *Types of queries for concept learning.* TR 479, YALEU/DCS, 1986.

[Ang87] Dana Angluin. Learning regular sets from queries and counter-examples. *Information and Computation*, 75:87–106, 1987.

[Ang88] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[AP64] V. Amar and G. Putzolu. On a family of linear grammars. *Information and Control*, 7:283–291, 1964.

[AS83] Dana Angluin and Carl H. Smith. Inductive inference : theory and methods. *ACM Computing Surveys*, 15(3):237–269, 1983.

[Bie71] Alan W. Biermann. A grammatical inference program for linear languages. In *Proceedings of Forth Hawaii International Conference on System Sciences*, pages 121–123, 1971.

[Bie72] Alan W. Biermann. An interactive finite-state language learner. In *Proceedings of First USA-JAPAN Computer Conference*, pages 13–20, 1972.

[Gol67] E Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Gol78] E Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[GS68] Seymour Ginsburg and Edwin H. Spanier. Control sets on grammars. *Mathematical Systems Theory*, 2(2):159–177, 1968.

[Har78] Michael A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, Massachusetts, 1978.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[Ish87] Hiroki Ishizaka. Inductive inference of regular languages based on model inference. In Koichi Furukawa, Hozumi Tanaka, and Tetsunosuke Fujisaki, editors, *Logic Programming '87, Lecture Notes in Computer Science, No.315*, pages 178–194, Springer-Verlag, 1987.

[McN67] Robert McNaughton. Parenthesis grammars. *Journal of the ACM*, 14(3):490–500, 1967.

[MCM86] R. Michalski, J. Carbonell, and T. Mitchell. *Machine Learning, Vol.1,2.* Morgan Kaufmann, Los Altos, CA, 1983,86.

[Moo56] Edward F. Moore. Gedanken-experiments on sequential machines. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages pp.129–153, Princeton Univ. Press, Princeton, N.J., 1956.

[Pyl86] Zenon W. Pylyshyn. *Computation and Cognition: Toward a Foundation for Cognitive Science.* MIT Press, 1986.

[RN88] V. Radhakrishnan and G. Nagaraja. Inference of even linear grammars and its application to picture description languages. *Pattern Recognition*, 21(1):55–62, 1988.

[Sak88a] Yasubumi Sakakibara. An efficient learning of context-free grammars for bottom-up parsers. In *Proceedings of FGCS '88*, pages 447–454, 1988.

[Sak88b] Yasubumi Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proceedings of 1st Workshop on Computational Learning Theory*, pages 296–310, 1988. To appear in *Theoretical Computer Science.*

[Sal73] Arto Salomaa. *Formal Languages.* Academic Press, Inc., New York, 1973.

[Tak87] Yuji Takada. *A constructive method for grammatical inference of linear languages based on control sets.* Research Report 78, IIAS-SIS, FUJITSU LIMITED, 1987.

[Tak88] Yuji Takada. Grammatical inference for even linear languages based on control sets. *Information Processing Letters*, 28(4):193–199, 1988.

[Tak89] Yuji Takada. Inferring parenthesis linear grammars based on control sets. 1989. To appear in Journal of Information Processing.

[Tan87] Keisuke Tanatsugu. A grammatical inference for context-free languages based on self-embedding. *Bulletin of Informatics and Cybernetics*, 22:149–163, 1987.

[Yok87] Takashi Yokomori. Inductive inference of context-free languages–context-free expression method. In *Proceedings of IJCAI'87, August, Milano*, pages 283–286, 1987.

[Yok88a] Takashi Yokomori. Inductive inference of contex-free languages based on context-free expressions. *International Journal of Computer Mathematics*, 24:115–140, 1988.

[Yok88b] Takashi Yokomori. *Learning simple languages in polynomial time.* Tech. Rep., SIGFAI, JSAI, 1988.