帰納的に可算な言語の補集合演算を用いた特徴付けについて

大 川　　知*

On the Characterrization of Recursiveley Enumerabele Languages by Using Comprementation

Satoshi OKAWA

Abstract

There have been many homomorphic characterizations of language classes by a homomorphic image of intersection of two languages.

In this note we attempt to find new homomorphic characterizations without intersection and we find that it is possible to characterize the class of recursiveley enumerable languages if complementation is available, that is, we have following characterizations.

1) For every recursively enumerable language L, there exist a context-free language L' and a homomorphism h such that

$L = h(\overline{L'})$.

2) For every recursively enumerable language L, there exist a deterministic context-free language L' and two homomorphisms h_1, h_2 such that $L = h_1(\overline{h_2(L')})$.

1. はじめに

言語理論において、言語のクラスの特徴付けをするという問題は重要な研究課題となっており、文法によって定まる言語のクラスと、言語の受理装置であるオートマトンによって定まる言語のクラスとの間の対応付けは、その一つの重要な結果である。また、これとは別に、これらのクラスを代数的に特徴付ける問題がある。例えば、文脈自由言語ＬをDyck言語Ｄと正規言語Ｒと準同形写像ｈとによって、$L = h(D \cap R)$ という形で表現する[(1)]、帰納的に可算な言語Ｌを２つの線形文脈自由言語L_1，L_2と準同型写像ｈと

平成２年12月15日受理
＊　電気工学科助教授

によって、$L=h(L_1 \cap L_2)$ という形で表現する[2]などの古典的な結果はよく知られている。

最近、任意の帰納的に可算な言語Lが、右長、左長、等長の３種の線形言語の部分クラスのうち、等長と等長を除く任意の２つのクラスの組に対して、そのおのおのから１つずつ選んだ言語 L_1, L_2 と準同形写像hとによって、$L=h(L_1 \cap L_2)$ と表わされる[4]、任意の帰納的に可算な言語Lが、ある同長線形言語 L_E と３つの準同型写像h, h_1, h_2 とによって $L=h(h_1(L_E) \cap h_2(L_E))$ と表わされる[5]などの結果が得られている。

代表的な特徴付けではいずれの結果においても、共通部分をとる演算と準同型写像とを用いているが、他の演算ではできないのであろうかという問題が生じる。特に、共通部分をとる演算を他の演算に代えて、帰納的に可算な言語の特徴付けができるかどうかという問題は興味深い。本論文では、補集合をとる演算を選んだとき、帰納的に可算な言語が特徴付けられることを示す。

2. 準　　備

本節において、本論文に必要な文法、導出、言語、オートマトン等に関する定義といくつかの事実を述べるが、詳細については、文献(6)などを参照されたい。

文法（grammar）とは $G=(N, \Sigma, P, S)$ である。ここで、Nは非終端記号の集合、Σは終端記号の集合（$V=\Sigma \cup N$ とする）、Pは書きかえ規則の集合、SはNの元で初期記号である。Pの元pは $p: \alpha \to \beta$（$\alpha, \beta \in V^*$, αはNの元を含む）という形をしている。文法Gの書きかえ規則 $p: \alpha \to \beta$ が文字列ωに適用可能であるとは、$\omega=\omega_1 \alpha \omega_2$ となっているときであり、ωにpを適用した結果は、$\omega'=\omega_1 \beta \omega_2$ なる文字列である。このとき $\omega \Rightarrow_G^p \omega'$ と表わし、p, Gについて特に言及する必要のないとき、それらを省略し、$\omega \Rightarrow \omega'$ などと表わす。さらに、関係⇒の反射的推移的閉包を $\Rightarrow^*$ と表わす。

言語(language)とは、Σ^* の任意の部分集合であり、文法Gによって生成される言語L(G)とは、

$$L(G)=\{\omega \mid S \overset{*}{\underset{G}{\Rightarrow}} \omega, \ \omega \in \Sigma^*\}$$

のことである。

文法Gのすべての書きかえ規則 $p: \alpha \to \beta$ について、何の制約もないとき、帰納的に可算な（recursiveley enumerable）文法、αがNの元である、すなわち、$p: A \to \beta$ であるとき、文脈自由（context-free）文法といい、それらが生成する言語を、それぞれ、帰納的に可算な言語、文脈自由言語という。そして、これらの言語のクラスをそれぞれREL, CFLと表わす。

プッシュダウンオートマトン（pushdown automaton,pda）とは、$M=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$ である。ここで、$Q,\Sigma,\Gamma,q_0,Z_0,F$ は、それぞれ状態集合、入力アルファベット、スタックアルファベット、初期状態、初期スタック記号、最終状態集合であり、δ は、$Q\times\Sigma'\times\Gamma$ から $Q\times\Gamma^*$ の部分集合への写像（動作関数）である（$\Sigma'=\Sigma\cup\{\varepsilon\}$ とする）。pdaMが状態qで、入力テープ、プッシュダウンスタックのそれぞれから、$w(\in\Sigma^*)$，$\gamma(\in\Gamma^*)$ の左端の文字を読んでいることを、$Q\times\Sigma^*\times\Gamma^*$ の元 $c=(q,w,\gamma)$ で表わし、これをMの様相という。$a\in\Sigma'$ としてMの様相が $c=(q,aw,Z\gamma)$ と表わされ、$\delta(q,a,Z)\ni(p,\gamma')$ であるとき、Mによって $c'=(p,w,\gamma'\gamma,\rho)$ に推移するといい、$c\vdash_M c'$ と表わす。このようにして定まるMの動作による様相上の関係 $\vdash_M$ の反射的推移的閉包を $\vdash^*_M$ と表わす。

pdaMの受理する言語を、

$L(M)=\{w\mid(q_0,w,Z_0)\vdash^*_M(q_f,\varepsilon,\gamma),\ q_f\in F,\gamma\in\Gamma^*\}$

と定義する。

さらに、δ について、$|\delta(q,a,Z)|+|\delta(q,\varepsilon,Z)|\leq 1$ を満すとき、すなわち、任意の様相cに対して、cから推移しうる様相が常に1個以下であるとき、Mを決定性（deterministic)pdaという。

そして、pda、決定性pdaによって受理される言語のクラスをそれぞれPDA，DPDAと表わす。このとき、次の命題が知られている。

〔命 題 1〕 $DPDA\subsetneq PDA=CFL\,\theta$

このことから、DPDAの元を決定性文脈自由言語といい、そのクラスをDCFLと表わすことが多い。

帰納的に可算な言語の特徴付けについては、前節でいくつか述べたが、本論文の証明に用いるものを定理として述べておく。

〔定 理 1〕(3) 任意の帰納的に可算な言語Lに対して、2つの決定性文脈自由言語 L_1，L_2 と、準同型写像hが存在して、

$$L=h(L_1\cap L_2)$$

となる。

また、CFL，DCFLのいくつかの基本的な演算の閉包性について次のことが知られている。

〔命 題 2〕 CFLは、和集合をとる演算、準同型写像で閉じているが、共通部分をとる演算、補集合をとる演算で閉じていない。

〔命 題 3〕 DCFLは、和集合をとる演算準、同型写像では閉じていないが、補集合をとる演算で閉じている。

3．結　　果

本節で、帰納的に可算な言語の共通部分をとる演算を用いない特徴付けを行う。命題2で述べたように、CFLの元は和集合をとっても準同型写像をとってもCFLの元であるから、これらをどのように組み合わせてもRELの元にはならない。そこで、補集合をとる演算について検討してみると次の結果が得られる。

〔定 理 2〕 任意の帰納的に可算な言語Lに対して、ある文脈自由言語L'と準同型写像hが存在して、

$$L = h(\overline{L'})$$

と表わされる。

（証 明） 定理1により、Lはある決定性文脈自由言語L_1, L_2と準同型写像hとによって、$L = h(L_1 \cap L_2)$と表わされる。

ところで、L_1, $L_2 \in$ DCFLであるから、命題3により、$\overline{L_1}$, $\overline{L_2} \in$ DCFL（$\subset$CFL）である。従って、命題2により、$\overline{L_1} \cup \overline{L_2} \in$ CFLとなる。そこで、L'を$L' = \overline{L_1} \cup \overline{L_2}$とすると、$\overline{L'} = \overline{\overline{L_1} \cup \overline{L_2}} = \overline{\overline{L_1}} \cap \overline{\overline{L_2}} = L_1 \cap L_2$であるから、

$$L = h(\overline{L'})$$

を得る。

定理2のL'を決定性文脈自由言語とすると、命題2，3により、$\overline{L'}$は、決定性文脈自由言語であり、それの準同型像はCFLの元に止まるから、決定性文脈自由言語を用いるのでは、このような形の特徴付けはできない。しかし、準同型写像を2回用いることを許せば、次のような結果が得られる。

〔系 1〕 任意の帰納的に可算な言語Lに対して、ある決定性文脈自由言語L_D'と、2つの準同型写像h_1, h_2とが存在して、

$$L = h_1(\overline{h_2(L_D')})$$

と表わされる。

（証 明） 任意の文脈自由言語L_Fに対して、Dyck言語Dと、ある正規言語Rと準同型写像fが定まり、$L_F = f(D \cap R)$となることが知られている。ここで、Dは決定性文脈自由言語である。DCFLは正規言語との共通部分をとる演算で閉じているから、$D \cap R$もDCFLの元である。

定理2より、$L = h(\overline{L'})$なるL'が存在する。このL'は、上に述べたことより、あるDとRとfによって、

$$L' = f(D \cap R)$$

と表わされる。従って、

$$L = h(\overline{f(D \cap R)})$$

を得る。ここで、$L_D' = D \cap R$, $h_1 = h$, $h_2 = f$ とすると、

$$L = h_1(\overline{h_2(L_D')})$$

となる。

4. むすび

本論文で、帰納的に可算な言語の特徴付けについて、共通部分をとる演算の代わりに他の演算を導入することを考え、補集合をとる演算を導入すれば、特徴付け可能であることを示した。これは、共通部分をとる演算を用いずに帰納的に可算な言語が得られるという点で、従来のものと異なる。さらに、決定性文脈自由言語にしたときには、準同型写像を1つ増すことによって同様に特徴付け可能であることを示した。

今後の課題として、1) L', L_D' を与える文法を具体的に示すこと、2)補集合をとる演算を用いてRELを特徴付けることの可能なより小さい言語のクラスを見出すこと、3)他の演算を用いた特徴付けについて検討することなどがあげられる。

参考文献

1) N.Chomsky, "Context-free grammars and pushdown strage", MIT Res.Lab.Electron.Quart.Prog.Report 65 pp.187-194(1962).

2) B.Baker and R.Book, "Reversal-bounded multipushdown machines", J.CSS, 8, pp.315-332(1974).

3) S.Ginsburg, S.Greibach and M.Harrison, "One-way stack automata", J.ACM, 14, pp.389-418(1967).

4) 大川 知、広瀬貞樹、“線形言語の部分族と言語の特性化”、信学論(D-I)J73-D-1,8,pp.649-650(平2.8).

5) 大川 知、“帰納的に加算な言語の同長線形文脈自由言語による特徴付けについて”、八戸工業大学情報システム工学研究所紀要第2巻、pp.14-19(平2.3).

6) 本多波雄、“オートマトン・言語理論”、コロナ社(昭47).