第 17 講 文法

林　恒俊

言語の定義

- ここまで形式言語、特に正規言語の定義法がいくつか提案された。
  - 集合演算
  - 正規表現
  - 記憶が有限な抽象機械すなわち DFA や NFA や EFA
- これらの手法は正規言語を超える言語を定義するためには十分でないことが知られている。実際にこれらの手段では正規表現自身を定義することすらできない。しかし抽象機械に有限でない記憶能力を付加すれば PDA や TM のように正規言語より複雑な言語を定義できる。

  課題 正規表現それ自身を正規表現で定義してみよ。
- ここではより高度な言語 (それが何かは後に説明される) を定義するための手段である**文法** (**grammar**) について説明する。
- 形式的文法は**句構造文法** (**phrase structure grammar**) として Noam Chomsky が考察し自然言語の構文を説明する手段として発展した。自然言語の文の階層構造を巧みに表現することができたため広く研究された。後にプログラミング言語が設計されるとその構文を定義する手段としても利用された。**バッカス・ナウアー形式** (**Backus Nauer form**, **BNF**) は後述する文脈自由文法をプログラミング言語定義向きに表現したもので Algol60 をはじめとして多数の言語の構文定義に広く活用されている。
- 句構造文法はパターンマッチと置換を利用して記号列を生成する方法である。文法の記述では一連の記号列を保持するため変数が使われる。これらの変数は抽象機械の状態に類似した役割を果たす。また文法の形式的定義も抽象機械の定義に似たところがある。

文法の定義

- 文法 G は 4 つの要素からなる組 $G = (N, \Sigma, P, S)$ として定義される。
- N は有限な記号の集合で**非終端記号集合** (**nonterminal symbol set**) と呼ばれる。その要素は記号列を値として保持する変数である。
- Σ は有限な記号の集合で**終端記号集合** (**terminal symbol set**) と呼ばれる。実は今までアルファベットと呼ばれたものと同一である。なお N と Σ は要素を共有しない。$(N \cap \Sigma = \emptyset)$
- P は**生成規則** (**production rule**) と呼ばれ次のように定義される。

  $$P = \{(\lambda, \rho) \mid \lambda \in (N \cup \Sigma)^+, \rho \in (N \cup \Sigma)^*\}$$

  すなわち $P \subseteq (N \cup \Sigma)^+ \times (N \cup \Sigma)^*$ である。なお生成規則の要素 (λ, ρ) は $\lambda \to \rho$ と書かれることが多い。生成規則の個別要素も生成規則と呼ぶことがある。
- S は N の特定の要素 $(S \in N)$ であり**開始記号** (**start symbol**) と呼ばれる。

導出

- **導出** (**derive**, **derivation**) とは文法で定義された生成規則を解釈する操作である。
- 終端記号及び非終端記号からなる任意の記号列を α, β とする。すなわち

  $$\alpha, \beta \in (N \cup \Sigma)^*$$

  である。
- 生成規則が $\lambda \to \rho$ という要素を含んでいる時、記号列 $\alpha\lambda\beta$ の λ の部分を ρ で置換えて $\alpha\rho\beta$ とする操作を導出といい

  $$\alpha\lambda\beta \Rightarrow \alpha\rho\beta$$

  と表示する。これは一種のパターンマッチによる置換である。すなわち、ある記号列中にいずれかの生成規則の左辺にマッチする部分

列があれば、その部分列を当該規則の右辺の記号列で置換えることをいう。

- 導出された記号列に生成規則をさらに適用して導出を重ねてもよい。いま

$$\alpha, \alpha_1, \alpha_2, \ldots, \alpha_i, \beta \in (\Sigma \cup N)^+$$

とする。ここで α から α_1 が導出され、α_1 から α_2 が導出され、最後に β が導出される一連の導出の列があるものとすると

$$\alpha \Rightarrow \alpha_1, \alpha_1 \Rightarrow \alpha_2, \ldots, \alpha_i \Rightarrow \beta$$

である。これを

$$\alpha \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_i \Rightarrow \beta$$

と表示する。またさらに

$$\alpha \overset{*}{\Rightarrow} \beta$$

とも表示し、0 回以上の導出を繰返して α から β が導出されたことを意味する。このような導出の列を**導出列** (**derivation**) と呼ぶ。

- 最終的に β が終端記号のみで構成される場合 ($\beta \in \Sigma^*$) これを**文** (**sentence**) といい、導出途中の終端記号と非終端記号の両方からなる記号列を**文形式** (**sentential form**) という。

- 文法 $G = (N, \Sigma, P, S)$ が定義する言語を $L(G)$ とすると

$$L(G) = \{\sigma \mid S \overset{*}{\Rightarrow} \sigma \wedge \sigma \in \Sigma^*\}$$

である。すなわち開始記号から導出されるすべての文の集合である。

文法の類別と言語

- この文法の定義は一般的すぎて漠然としているので生成規則に制限を加えてもう少し実用的な言語を定義する。

- 制限のない場合の言語を帰納可算集合という。この言語は TM が定義する言語 $\mathcal{L}_{\mathbf{TM}}$ と同じクラスである。

- $|\lambda| \leq |\rho|$ すなわち規則の左辺の長さより右辺の長さが等しいか大きいという制限を付加した場合の文法を**文脈依存文法** (**context sensitive grammar**, **CSG**) と呼ぶ。文脈依存文法 G_{CSG} が定義する言語 $L(G_{CSG})$ は空列を含まないのでこれに空列を追加した言語 $L(G_{CSG}) \cup \{\Lambda\}$ のクラスを文脈依存言語という。$L(G_{CSG})$ は TM に制限を加えて定義された**線形拘束機械** (**linear bounded automaton**, **LBA**) に対応している。

- $\lambda \in N$ の場合、すなわち左辺が 1 個の非終端記号のみの文法を**文脈自由文法** (**context free grammar**, **CFG**) といい定義される言語が文脈自由言語である。導出時に 1 個の非終端記号とパターンマッチすればよいため前後の記号列の影響を受けない。このため文脈自由といわれる。

  なお文脈依存文法では記号列パターンマッチにより導出が行われるので前後の記号列が影響する。そのため文脈依存と呼ばれる。

- 上記の制限に加えてさらに $\rho \in \Sigma^* N \cup \Sigma^*$ としたものを**正規文法** (**regular grammar**, **RG**) といい定義される言語が正規言語である。正規文法の生成規則の左辺は文脈自由文法と同じで 1 個の非終端記号、右辺は

  - 終端記号列と最後尾の 1 個の非終端記号か
  - 終端記号列のみ

  から構成される。

- この正規言語は今まで説明してきた正規言語と全く同じものである。すなわち正規言語を定義する文法が正規文法である。ただしこれについては検証しなければならない。

- これらの言語のクラス間には

$$\mathcal{L}_{\mathbf{TM}} \supseteq \mathcal{L}_{\mathbf{CSL}} \supseteq \mathcal{L}_{\mathbf{CFL}} \supseteq \mathcal{L}_{\mathbf{RL}}$$

  という関係が成立する。

文法例

- 文法 G_1 を次のように定義する。

$$G_1 = (\{S\}, \{a\}, P, S) \text{ ただし } P = \{S \to \Lambda, S \to aS\}$$

文法 G_1 は RG でその定義する言語 $L(G_1)$ は正規言語

$$L(G_1) = \{a^n \mid n \geq 0\} = \{\Lambda, a, aa, \ldots\}$$

である。G_1 の導出例を次に示す。

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aa$$

- 文法 G_2 を次のように与える。

$$G_2 = (\{S\}, \{a, b\}, P, S) \text{ ただし } P = \{S \to \Lambda, S \to aSb\}$$

文法 G_2 は CFG でその定義する言語 $L(G_2)$ は文脈自由言語

$$L(G_2) = \{a^n b^n \mid n \geq 0\} = \{\Lambda, ab, aabb, \ldots\}$$

である。G_2 の導出例を次に示す。

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

- 文法 G_3 を次のように定義する。

$$G_3 = (\{S, A, B\}, \{a, b, c\}, P, S)$$

ただし

$$\begin{aligned} P = \{&S \to abc, \\ &S \to aAbc, \\ &Ab \to bA, \\ &Ac \to Bbcc, \\ &bB \to Bb, \\ &aB \to aa, \\ &aB \to aaA\} \end{aligned}$$

文法 G_3 は CSG でその定義する言語 $L(G_3)$ は文脈依存言語

$$L(G_3) = \{a^n b^n c^n \mid n \geq 1\} = \{abc, aabbcc, \ldots\}$$

である。G_3 の導出例を次に示す。

$$S \Rightarrow aAbc \Rightarrow abAc \Rightarrow abBbcc \Rightarrow aBbbcc \Rightarrow aabbcc$$

より複雑な文法例

- アルファベット $\{a, b, c\}$ 上の言語 $L_{\text{sum}} = \{a^m b^n c^{m+n} \mid m, n \geq 0\}$ は整数の加算演算を定義するものと見なすことができる。

$$L_{\text{sum}} = \{\Lambda, ac, bc, abcc, \ldots\}$$

であり c の数が a の数と b の数の和になっている。

- L_{sum} は CFL で次のように与えられる。文法を G_{sum} とすると

$$G_{\text{sum}} = (\{S, T\}, \{a, b, c\}, P, S)$$

ただし

$$\begin{aligned} P = \{&S \to aSc, \\ &S \to T, \\ &T \to bTc, \\ &T \to \Lambda\} \end{aligned}$$

G_{sum} の導出例を次に示している。

$$S \Rightarrow aSc \Rightarrow aTc \Rightarrow abTcc \Rightarrow abbTccc \Rightarrow abbccc$$

- 導出過程から判断できるように S に関する導出では文の前後に a と c を同一数追加する。S から T に導出後 T は b と c を文中に同一数付け加える。最終的に T は空列を導出するため消去されるが a, b の数の和と c の数は常に同じである。

- L_{sum} で常に $m = n$ とした言語 $\{a^n b^n c^{2n} \mid n \geq 0\}$ は CFG で定義できない。これは $\{a^n b^n c^n \mid n \geq 0\}$ と同様に文脈依存言語である。

課題　言語 $\{a^m b a^n b a^{m+n} \mid m, n \geq 0\}$ を生成する CFG を考察せよ。