記号処理・第2回

2017年4月18日

今回の内容

2.1 BNF 記法

コンパイラがソースプログラムの構文解析を行うためには、そもそも、そのプログラミング言語がどのような言語であるのか、つまり、どのような文字や記号の並びがプログラムとして許されるのかが明確に定義されている必要があります。コンパイラは、この定義にしたがって構文解析をしていかなければなりません。そこで、まずプログラミング言語の文法を定義する方法の1つである **BNF 記法**について解説します。BNF 記法 (Backus and Naur Form あるいは Backus Normal Form) は記号列の集合を再帰的に定義するときに用いられる記法で、プログラミング言語の文法などを記述する時に広く用いらます。

メモ

BNF記法の例　次は BNF 記法を使った定義した文法の例です。

$$\langle 変数\rangle ::= “x” \mid “y” \mid “z” .$$
$$\langle 項\rangle ::= \langle 変数\rangle \mid “(” \langle 式\rangle “)” \mid \langle 項\rangle “*” \langle 変数\rangle \mid \langle 項\rangle “*” “(” \langle 式\rangle “)” .$$
$$\langle 式\rangle ::= \langle 項\rangle \mid \langle 式\rangle “+” \langle 項\rangle .$$

この例中の「変数」のように ⟨ ⟩ で囲まれた記号を**非終端記号**と呼び、「x」のように “ ” で囲まれた記号を**終端記号 (トークン)** と呼びます。::= の左辺には非終端記号が1つあり、右辺には非終端記号または終端記号からなる記号列が ｜ で区切られて置かれていることに注意してください。各行を、**導出規則** (あるいは **生成規則**)、または、左辺の非終端記号の**定義**と呼びます。各導出規則は . (period) で終ります[1]。非終端記号と終端記号の混同のおそれが無い場合には、⟨ ⟩ や “ ” を省

[1] 通常、BNF 記法ではこの「.」を書かずに、行末で導出規則の終わりを示すのが一般的ですが、この科目では、導出規則を複数行に渡って書く場合に、導出規則の終わりが明確になるように「.」を書くことにします。

略して、導出規則を

$$\text{式} ::= \text{項} \mid \text{式} + \text{項}.$$

のように書くこともあります。

メモ

2.1.1 導出

非終端記号または終端記号からなる記号列の中に現れている1つの非終端記号を、その非終端記号を左辺に持つ導出規則を使って、その右辺の $\mid$ で区切られているいずれかの記号列に置き換える操作を**導出**といい、記号列 α が導出によって記号列 β に書き換わることを $\alpha \to \beta$ で表します。たとえば、

$$\begin{aligned} &\langle\text{式}\rangle \ \to\ \langle\text{式}\rangle\texttt{+}\langle\text{項}\rangle \ \to\ \langle\text{項}\rangle\texttt{+}\langle\text{項}\rangle \ \to\ \langle\text{変数}\rangle\texttt{+}\langle\text{項}\rangle \ \to\ \texttt{z+}\langle\text{項}\rangle \\ &\to\ \texttt{z+}\langle\text{項}\rangle\texttt{*}\langle\text{変数}\rangle \ \to\ \texttt{z+}\langle\text{変数}\rangle\texttt{*}\langle\text{変数}\rangle \ \to\ \texttt{z+y*}\langle\text{変数}\rangle \ \to\ \texttt{z+y*y} \end{aligned}$$

となります。この記号列の間の二項関係 $\to$ の反射的推移的閉包[2]を $\stackrel{*}{\to}$ で表します。つまり、$\alpha \stackrel{*}{\to} \beta$ とは、記号列 α から何回か (0回以上[3]) 導出行うと β になる、という意味です。例えば、上の例によると $\langle\text{式}\rangle \stackrel{*}{\to} \texttt{z+y*y}$ ということになります。

メモ

2.1.2 非終端記号が生成する集合

N が非終端記号であるとき、次のような (記号列の) 集合を、非終端記号 N が**生成する (記号列の) 集合**、あるいは非終端記号 N が**表す集合**と呼びます。

$$\{\ \alpha \mid \langle N\rangle \stackrel{*}{\to} \alpha \ \text{かつ}\ \alpha\ \text{は非終端記号を含まない}\ \}$$

メモ

[2] 二項関係 $\to$ の反射的推移的閉包 $\stackrel{*}{\to}$ とは、$\to$ を含み、次の2つの条件を満たす二項関係の内で最小のものをいいます。

(反射律) 任意の α について、$\alpha \stackrel{*}{\to} \alpha$

(推移律) 任意の α、β、γ について、$\alpha \stackrel{*}{\to} \beta$ かつ $\beta \stackrel{*}{\to} \gamma$ ならば $\alpha \stackrel{*}{\to} \gamma$

[3] 0回でもよいので、どんな記号列 α についても、$\alpha \stackrel{*}{\to} \alpha$ が成り立ちます。

2.1.3 空列

読みやすくするため、空の記号列、つまり長さ 0 の記号列を ε で表すことにします。

メモ

2.1.4 いくつかの表記法

以下、次のような表記 (略記) 法を導入して BNF 記法を拡張[4]することにします。

- α が記号列のとき、BNF 記法の各導出規則の右辺で、$\{\alpha\}$ を次のように定義される非終端記号として扱うものとします[5]。

$$\{\alpha\} \ ::= \ \varepsilon \mid \alpha \, \{\alpha\} \, .$$

つまり、$\{\alpha\}$ は、記号列 α から導出される終端記号列を何回 (0 回以上) か繰り返すことによって得られるような記号列全体の集合

$$\{\ \beta_1\beta_2\ldots\beta_n \mid n \geqq 0,\ \ \alpha \xrightarrow{*} \beta_i \text{ かつ } \ \beta_i \text{ は非終端記号を含まない } \ (i = 1,\ 2\ \ldots,\ n)\ \}$$

を表します。空な記号列 ε も $\{\alpha\}$ の表す集合に含まれている ($\{\alpha\} \xrightarrow{*} \varepsilon$) ことに注意してください。たとえば、1 ページの ⟨式⟩ の導出規則は

$$\langle\text{式}\rangle ::= \langle\text{項}\rangle\ \{\ \text{“+”}\ \langle\text{項}\rangle\ \}\,.$$

のように書いても ⟨式⟩ が表す (終端記号列の) 集合は変わりません。

メモ

- 同様に $[\,\alpha\,]$ を次のように定義される非終端記号として扱います。

$$[\,\alpha\,] \ ::= \ \varepsilon \mid \alpha\,.$$

つまり、$[\,\alpha\,]$ は、空であるか、あるいは記号列 α から導出されるような終端記号列全体の集合

$$\{\,\varepsilon\,\} \cup \{\ \beta \mid \alpha \xrightarrow{*} \beta \text{ かつ } \ \beta \text{ は非終端記号を含まない }\}$$

を表します。

メモ

[4] このように拡張された BNF 記法を **拡張 BNF 記法 (EBNF)** と呼ぶことがあります。

[5] 集合を表すための記法 (例えば、$\{0,\ 1,\ 2,\ 3,\ \ldots\}$ や $\{x \mid x \text{ は素数}\}$ など) と混同しないように注意してください。

- $(\alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \ldots \mid \alpha_n)$ を次のように定義される非終端記号 $\langle N \rangle$ として扱います。

$$\langle N \rangle ::= \alpha_1 \mid \alpha_2 \mid \alpha_3 \mid \ldots \mid \alpha_n .$$

たとえば、1 ページの ⟨項⟩ の導出規則は

$$\langle 項 \rangle ::= \langle 変数 \rangle \mid \text{“(”} \ \langle 式 \rangle \ \text{“)”} \mid \langle 項 \rangle \ \text{“*”} \ (\ \langle 変数 \rangle \mid \text{“(”} \ \langle 式 \rangle \ \text{“)”}\ ) .$$

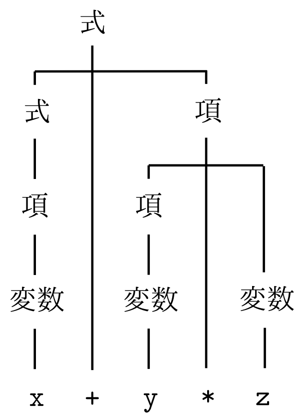
のように書いても ⟨項⟩ が表す (終端記号列の) 集合は変わりません[6]。

メモ

2.2 構文木

ある終端記号の列が 1 つの非終端記号から導出される様子を木構造で表現したものを**構文木**、あるいは **解析木** と呼びます。構文木の葉でない節にはそれぞれ非終端記号が記され、葉には終端記号が記されます。各節からでる枝は、その節の非終端記号が (ある導出規則によって) 別の記号列に置き換えられることを表します。このため、1 つの節から出る枝の順番を無視することはできません。

右の図は (1 ページ目の文法における) 記号列 `x + y * z` の非終端記号 ⟨式⟩ に対する構文木の例です。



メモ

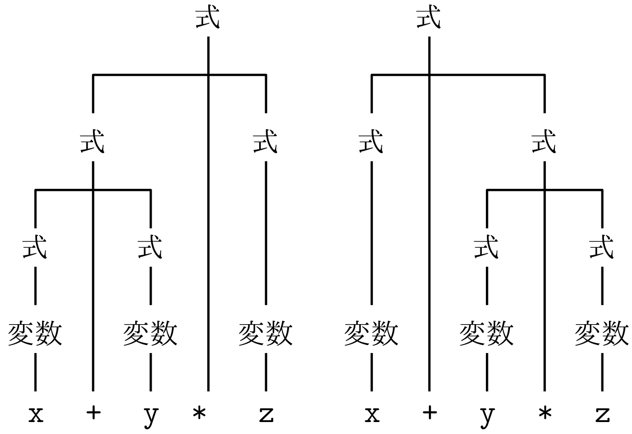
2.3 文法の曖昧性

非終端記号に対する (ある終端記号列の) 構文木が複数存在するとき、その非終端記号の定義は**曖昧性を持つ** といいます。また、曖昧性を持つ非終端記号で与えられる文法があったとき、その文法は**曖昧性を持つ**といいます。例えば、

$$\langle 変数 \rangle ::= \text{“x”} \mid \text{“y”} \mid \text{“z”} .$$
$$\langle 式 \rangle ::= \langle 変数 \rangle \mid \langle 式 \rangle \ \text{“*”} \ \langle 式 \rangle \mid \langle 式 \rangle \ \text{“+”} \ \langle 式 \rangle \mid \text{“(”} \ \langle 式 \rangle \ \text{“)”} .$$

[6] “*” の後の (⟨変数⟩ | “(” ⟨式⟩ “)”) の部分にこの記法が用いられています。“(” ⟨式⟩ “)” の部分に現れている “(” や “)” は単なる終端記号ですので混同しないでください。

という文法は曖昧性を持っています。なぜなら、`x + y * z` という記号列は、非終端記号 ⟨式⟩ に対して、下図のような 2 つの構文木を持つからです。



この文法と 1 ページ目の冒頭に挙げた文法を比べると、⟨式⟩ が表す記号列の集合は全く同じになりますが、この文法が曖昧性を持つのに対して、1 ページ目の文法は曖昧性を持たないことに注意してください。

メモ

2.4 演習問題

1. 1 ページの BNF 記法で与えられた文法に関して、非終端記号 ⟨変数⟩、⟨項⟩、⟨式⟩ それぞれから、以下の記号列が導出できるかどうかを判定しなさい。また、導出可能である場合はその構文木を作りなさい。

(1)	`y`	(2)	`+ x`	(3)	`y * z`
(4)	`x + x`	(5)	`x + y + z`	(6)	`x * y * z`
(7)	`x + y * z`	(8)	`x * y + z`	(9)	`( z )`
(10)	`( y * z )`	(11)	`x + ( y * z )`	(12)	`( x + y ) * z`

2. 英文字で始り、その後に英文字か数字が 0 個以上続く文字列全体を表す非終端記号 ⟨変数名⟩ の文法を BNF 記法で与えなさい。

3. 次の BNF 記法で与えられる文法に導出規則を追加して、非終端記号 ⟨正整数⟩ が、`1`、`3`、`1028` など、10 進表記の正の整数をすべて生成するようにしなさい。そのとき、`001` や `0123` など、省略可能な `0` を含む冗長な表記が導出できないようにしなさい。

⟨非ゼロ数字⟩ ::= “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9” .

⟨数字⟩ ::= “0” | ⟨非ゼロ数字⟩ .

4. 上の文法にさらに導出規則を追加して、非終端記号 ⟨整数⟩ が、`0`、`3`、`1028`、`-23` など、10 進表記の整数をすべて生成するようにしなさい。そのとき、`-0`、`0123`、`-0123` など、省略可能な記号を含む冗長な表記が導出できないようにしなさい。

5. 上の文法にさらに導出規則を追加して、非終端記号 ⟨実数⟩ が、`23.641`、`395`、`-0.39` など、10 進小数表記をすべて生成するようにしなさい。そのとき、`034.2` や `1.0`、`+3.45`、`28.9500` など、省略可能な符号、`0`、小数点が含まれている冗長な表記が導出できないようにしなさい。

6. BNF 記法で与えられた次のような文法を、[ ] や { } の記法を用いずに書き直しなさい。

⟨変数⟩ ::= "x" | "y" | "z" .
⟨演算子⟩ ::= "+" | "-" .
⟨式⟩ ::= [⟨演算子⟩] ⟨変数⟩ { ⟨演算子⟩ ⟨変数⟩ } .

7. 次の BNF 記法で与えられる文法が曖昧であることを示しなさい。

⟨2進数字⟩ ::= "0" | "1" .　　⟨2進数⟩ ::= ⟨2進数字⟩ | ⟨2進数⟩ ⟨2進数⟩ .

記号処理・第 2 回・終わり