2.3 バッカス記法

プログラミング言語をきちんと定義するには，正しいプログラムはどんな形をしていなければならないか (これを言語の**構文**とか**シンタックス**という) に関する規定と，その構文規則に従って書かれたプログラムはどのように動作しどのような機能を果すかという，プログラムの意味に関する規定 (これを**セマンティックス**という) とを明確に述べる必要がある．例えば，多くのプログラミング言語では **if 文**と呼ばれるものがあり，次のような形をしている‖．ただし，p は条件を表わす '式' であり，S_1 と S_2 は '文' と呼ばれるものである．

$$\textbf{if} \quad p \quad \textbf{then} \quad S_1 \tag{2.8}$$

あるいは

$$\textbf{if} \quad p \quad \textbf{then} \quad S_1 \quad \textbf{else} \quad S_2 \tag{2.9}$$

● **バッカス記法**　'式' とは何か，'文' とは何かはしばらくさておき，上のことをバッカス記法と呼ばれる書き方では次のような式 (**超式***という) で表わす：

〈**if** 文〉 ::= **if**〈式〉**then**〈文〉 ｜ **if**〈式〉**then**〈文〉**else**〈文〉

'文'，'式'，'**if** 文' はそれぞれプログラムを構成する要素の 1 つである．これらを〈 〉でくくって表わし，**超変数***と呼ぶ．**if**，**then** はそれぞれそれ以上には分解できない 1 つの記号であり (**語記号**とか**予約語**という)，$A, B, C, \cdots, Z, a, b, c, \cdots, z, 0, 1, \cdots, 9, +, -, *, /, \langle, \rangle, =, (,)$ といった文字や記号とともにプログラムを書くために使われる基本文字である．超式の左辺には定義したい超変数を書き，右辺には基本記号と超変数と記号'｜'を含んだ文字列を書く．左辺はこの右辺に書かれた形をしているものとして定義される．'｜' は「または」を表わす．

このように，バッカス記法を用いると **if** 文の構文を厳密に記述することができるが，構文は **if** 文の字面 (外見的な形) しか規定しない．一方，(2.8), (2.9)

‖以下の例は，Pascal と呼ばれるプログラミング言語に似せたものを使っている．

*超式も式であるが，式とは何かについても超式を用いて定義するので，1 つ次元が高いものという意味で超 (meta) を付けて呼ぶのである．超変数についても同様．

の形をした **if** 文が果たす機能はそれぞれ「もし条件 p が成り立つなら S_1 を実行せよ」,「もし条件 p が成り立つなら S_1 を実行せよ．p が成り立たないなら S_2 を実行せよ」である．この「機能」，すなわち，**if** 文の実行によって何が起こるかという「意味」が **if** 文のセマンティックスである．

〔例 2.7〕 **if** 文の使用例

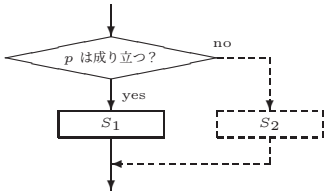
（1） $|x|$ をあらためて x とする

if $x < 0$ **then** $x \leftarrow -x$

（2） $max = x, y$ の大きい方

if $x \geqq y$ **then** $max \leftarrow x$

else $max \leftarrow y$



‘式’ とは何か，‘文’ とは何かをもう少し見てみよう．‘式’ には，数式 (例えば，$a + 2 * b$ とか $3 * x \uparrow 4 - 5 * y/(6 + z)$ など[†]) や，条件を表わすための論理式 (例えば，$x \leqq y$ とか $(D = b \uparrow 2 - 4 * a * c) \wedge (D > 0)$ など) がある．‘数式’ や ‘変数’ の構文については後述する．文には，次に示すような 8 種がある[‡]．

〈式〉 ::= 〈数式〉 | 〈論理式〉

〈文〉 ::= 〈代入文〉 | 〈呼出し文〉 | 〈複合文〉 | 〈**if** 文〉 | 〈**while** 文〉

〈**for** 文〉 | 〈**goto** 文〉 | 〈空文〉

このうち，‘代入文’ は一般に

変数 ← 式

という形をしており，← の右辺の ‘式’ の値を左辺の ‘変数’ に代入するという機能をもつ．

‘**if** 文’ を定義する際に ‘文’ とは何かを使った．その ‘文’ を定義する上式では ‘**if** 文’ とは何かを使っている．これは，‘文’ を定義するために ‘文’ 自身が間接的に使われている再帰的な定義になっていることに注意したい．

バッカス記法 (**バッカス・ナウア記法**とか， Backus-Naur form の頭文字を

[†]この例のように，乗除算は記号 $*$, / を使って明示的に記述する．$\uparrow$ は ‘累乗’ を表わす．例えば，$2 \uparrow 3 = 2^3 = 8$.

[‡]本書では，それぞれの ‘文’ の詳細について知る必要はない．

とって**BNF** ともいう) は，Algol60 と呼ばれるプログラミング言語の構文を規定するために V. Backus と P. Naur が初めて使用した．

〔例 2.8〕 バッカス記法による定義

（1） 10 進数とは，0, 1, ⋯ , 9 を 1 個以上任意に並べた文字列のことである (000 のようなものも許す)．これは次のように定義することができる：

$$\langle 10\text{ 進数}\rangle ::= \underbrace{0}_{①} \mid \underbrace{1}_{②} \mid \cdots \mid \underbrace{9}_{③} \mid \underbrace{\langle 10\text{ 進数}\rangle 0}_{④} \mid \cdots \mid \underbrace{\langle 10\text{ 進数}\rangle 9}_{⑤}$$

例えば，②により 1 が 10 進数であるから，④により 10 は 10 進数であり，それと⑤により 109 も 10 進数である，⋯，といった具合に 10 進数が定まる．

（2） 多くのプログラミング言語では，プログラムの中で使われる変数などに付ける名前 (識別子ともいう) は，英字で始まり，2 文字目以降が英字か数字であるような文字列である．英字または数字のことを英数字という：

$$\left.\begin{array}{l} \langle \text{名前}\rangle ::= \langle \text{英字}\rangle \mid \langle \text{名前}\times\text{英数字}\rangle \\ \langle \text{英数字}\rangle ::= \langle \text{英字}\rangle \mid \langle \text{数字}\rangle \\ \langle \text{英字}\rangle ::= A \mid B \mid C \mid \cdots \mid Z \mid a \mid b \mid c \mid \cdots \mid z \\ \langle \text{数字}\rangle ::= 0 \mid 1 \mid 2 \mid \cdots \mid 9 \end{array}\right\} \quad (2.9)$$

例えば $AB1$ が名前であることは，次ページの図 (左図) に示したようにわかる．ただし，$\overset{\alpha}{\underset{\beta_1}{\wedge}\!\!\underset{\beta_k}{\cdots}}$ は $\beta_1 \cdots \beta_k$ が α であることを表わしている．

（3） バッカス記法を拡張して，$\lambda \mid \alpha \mid \alpha\alpha \mid \alpha\alpha\alpha \mid \cdots$ を $\{\alpha\}^*$ で，$\alpha \mid \alpha\alpha \mid \alpha\alpha\alpha \mid \cdots$ を $\{\alpha\}^+$ で表わすことがある．これを使うと，例えば ‘2 進数’ や ‘名前’ は

$$\begin{array}{l} \langle \text{基本数字}\rangle ::= 0 \mid 1 \\ \langle \text{2 進数}\rangle ::= \{\langle \text{基本数字}\rangle\}^+ \quad \text{あるいは} \\ \langle \text{2 進数}\rangle ::= \{0 \mid 1\}^+ \\ \langle \text{名前}\rangle ::= \langle \text{英字}\rangle\, \{\langle \text{英数字}\rangle\}^* \end{array} \quad (2.10)$$

のように簡潔に表わすことができる (厳密に言うと，(2.10) が (2.9) の解であることは定理 1.9 を用いて証明されることである．各 $\langle\cdots\rangle$ は言語である)．

（4） ‘数式’ を次のように定義する．$\uparrow$ は累乗演算子である．

〈数式〉 ::= 〈数式〉+〈項〉 | 〈数式〉−〈項〉 | 〈項〉

〈項〉 ::= 〈項〉∗〈因子〉 | 〈項〉/〈因子〉 | 〈因子〉

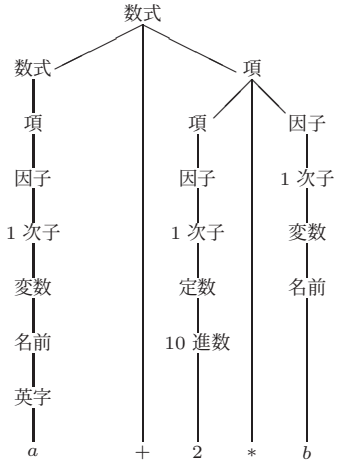
〈因子〉 ::= 〈1 次子〉↑〈因子〉 | 〈1 次子〉

〈1 次子〉 ::= −〈1 次子〉 | (〈数式〉) | 〈変数〉 | 〈定数〉

〈変数〉 ::= 〈名前〉

〈定数〉 ::= 〈10 進数〉

$a+2*b$ が数式であることは次の図 (右図) に示したようにわかる．この図は，$a+2*b$ が上記の BNF による定義 (文法) に関してどのような構文構造をしているかを表わしているので，**構文木**と呼ばれる[§]．構文木を描いてみると，乗法 $*$, / の方が加法 $+$, $-$ より先に計算されるべきであることが BNF によって規定されていることも明瞭になる．さらに，BNF は，例えば $a+b+c$ は $(a+b)+c$ のように「左から右に」計算されるべきであることも規定している ($a+b+c$ の構文木を書いて確かめよ)．$a+2*b$ の表わし方はこの 1 通りしかないが (確かめよ)，それは数式の計算順序が一意的に定まることを意味し，重要なことである (問 2.19 参照)． □



[§]構文木は，後述の文脈自由文法の観点からは，$a+2*b$ が生成 (導出) される様子を表わしていると見ることができるので，**導出木**とも呼ばれる．

● (**文脈自由**) **文法**　BNF に現れる超変数それぞれは，実はその超変数によって表わしたいものを文字列の集合すなわち言語として定義する．例 2.8 (2) を例にして考えてみよう．登場する超変数は〈名前〉, 〈英数字〉, 〈英字〉, 〈数字〉の 4 つである．簡単のために，これらを記号 **N**, **A**, **L**, **D** で表わすことにし，

$$\Gamma_{name} := \{\mathbf{N},\, \mathbf{A},\, \mathbf{L},\, \mathbf{D}\} \qquad \text{(超変数の集合)}$$

とおこう．**N**, **A**, **L**, **D** それぞれが言語を定義しているのであるが，一方，基本文字はその 1 文字だけを元とする言語と考えてよい．::= と | の意味を考慮し，::= を = に，| を ∪ に置き換えて (2.9) を書き直すと，

$$\begin{cases} \mathbf{N} = \mathbf{L} \cup \mathbf{NA} & \\ \mathbf{A} = \mathbf{L} \cup \mathbf{D} & \\ \mathbf{L} = \{A, B, C, \ldots, Z, a, b, c, \ldots, z\} & (\mathbf{L} \text{ は基本文字の集合}) \\ \mathbf{D} = \{0, 1, 2, \ldots, 9\} & (\mathbf{D} \text{ も基本文字の集合}) \end{cases} \tag{2.11}$$

という，$\mathbf{N}, \mathbf{A}, \mathbf{L}, \mathbf{D}$ を変数とする連立方程式になる．この連立方程式を満たす $\mathbf{N}, \mathbf{A}, \mathbf{L}, \mathbf{D}$ の解 (それらはアルファベット $\{A, \ldots, Z, a, \ldots, z, 0, \ldots, 9\}$ 上の言語である) が，それらによって定義したいもの (を文字列の集合として表わしたもの) である．定理 1.19 と同様に証明できるが，この連立方程式の解 $\mathbf{N}, \mathbf{A}, \mathbf{L}, \mathbf{D}$ は一意的に定まり ($\mathbf{A}, \mathbf{L}, \mathbf{D}$ の解は明らかであろう)，とくに，

$$\mathbf{N} = \mathbf{LA}^* = \{A, \ldots, Z, a, \ldots, z\}\{A, \ldots, Z, a, \ldots, z, 0, \ldots, 9\}^*$$

である (問 2.17 参照)．ところで，文字列に使うことのできる基本文字の集合は

$$\Sigma_{name} := \mathbf{L} \cup \mathbf{D} = \{A, \ldots, Z, a, \ldots, z, 0, \ldots, 9\}$$

であることにも留意しておこう．

次に，超式について考える．例えば，$\mathbf{N} = \mathbf{L} \cup \mathbf{NA}$ の右辺の **N** にこの右辺自身を代入して展開していく ($\mathbf{N}, \mathbf{L}$, **A** を集合として扱う) と，

$$\begin{array}{rcll} \mathbf{N} & = & \mathbf{L} \cup \mathbf{NA} & = \quad \mathbf{L} \cup (\mathbf{L} \cup \mathbf{NA})\mathbf{A} \\ & = & \mathbf{L} \cup \mathbf{LA} \cup \mathbf{NA}^2 & = \quad \mathbf{L} \cup \mathbf{LA} \cup (\mathbf{L} \cup \mathbf{NA})\mathbf{A}^2 \\ & = & \mathbf{L} \cup \mathbf{LA} \cup \mathbf{LA}^2 \cup \mathbf{NA}^3 & = \quad \cdots \\ & = & \bigcup_{i=0}^{n-1} \mathbf{LA}^i \cup \mathbf{NA}^n & \\ & \supseteq & \bigcup_{i=0}^{n} \mathbf{LA}^i & (\mathbf{N} \text{ に } \mathbf{L} \subseteq \mathbf{N} \text{ を代入}) \end{array} \tag{2.12}$$

となるが，$\mathbf{N}, \mathbf{L}, \mathbf{A}$ を文字として扱い，式変形 (2.12) に対応するように，文字 $\mathbf{N}$ を文字列 $\mathbf{NA}$ に書き換えることを i 回行ない，最後に $\mathbf{N}$ を $\mathbf{L}$ に書き換えると，

$$\begin{aligned} \mathbf{N} &\Rightarrow \mathbf{NA} \Rightarrow \mathbf{NAA} \Rightarrow \mathbf{NAAA} \Rightarrow \cdots \\ &\Rightarrow \mathbf{N}\overbrace{\mathbf{A}\cdots\mathbf{A}}^{i} = \mathbf{NA}^i \Rightarrow \mathbf{LA}^i \end{aligned} \tag{2.13}$$

となる．とくに，$i = 0$ の場合，$\mathbf{N} \Rightarrow \mathbf{L} = \mathbf{LA}^0$ である．(2.12) の言語 $\bigcup_{i=0}^{n} \mathbf{LA}^i$ は，$i = 0, \ldots, n$ について (2.13) を行なって $\{\mathbf{LA}^i \mid i = 0, \ldots, n\}$ を得ることに対応している．以上の考察より，超式

$$\alpha \ ::= \ \beta_1 \mid \cdots \mid \beta_n$$

には，文字列の書き換え規則の集まり

$$\{\alpha \to \beta_1, \ \cdots, \ \alpha \to \beta_n\}$$

($\alpha \to \beta_i$ は α を β_i に書き換えてもよいことを表わす規則) が対応していると考えられる．例えば，(2.9) の場合には，対応する書き換え規則の集合は

$$\begin{aligned} \Pi_{name} \ := \ & \{\mathbf{N} \to \mathbf{L}, \mathbf{N} \to \mathbf{LA}\} \cup \{\mathbf{A} \to \mathbf{L}, \mathbf{A} \to \mathbf{D}\} \cup \\ & \{\mathbf{L} \to A, \ldots, \mathbf{L} \to Z, \mathbf{L} \to a, \ldots, \mathbf{L} \to z\} \cup \\ & \{\mathbf{D} \to 0, \ldots, \mathbf{D} \to 9\} \end{aligned}$$

である．以後，$\alpha \to \beta_1, \ldots, \alpha \to \beta_k$ を，BNF と同様に，$\alpha \to \beta_1 \mid \cdots \mid \beta_k$ と略記する．

最後に，いくつも超式がある場合でも，そもそもその BNF で定義したかったもの (超変数) があることに注意する (‘名前’ の場合には，$\mathbf{N}$ がそれである)．そこで，超変数 (非終端記号ともいう) の集合 Γ，基本文字 (終端記号ともいう) の集合 Σ，超式に対応する書き換え規則の集合 Π，および，そもそも定義したいもの (例えば $\mathbf{S}$ とする) を示せば，それは BNF と本質的に同じものである．BNF をこのように，文字列の書き換え規則の集合として表わしたシステム

$$G := (\Gamma, \Sigma, \Pi, \mathbf{S})$$

のことを**文脈自由文法**(**CFG** と略記する) と呼ぶ[¶]．Γ, Σ は有限のアルファベッ

[¶]CFG: context-free grammar．書き換え規則 $\alpha \to \beta$ の左辺 α を 1 文字に限定せず任意

ト，Π は有限個の書き換え規則の集合，$\mathbf{S}$ は Γ の特定の元である．書き換え規則 $(\alpha \to \beta) \in \Pi$ の左辺 α は 1 つの文字 (Γ の元) であり，右辺 β は $\Gamma \cup \Sigma$ 上の語であることに注意する．

● **生成**　CFG $G = (\Gamma, \Sigma, \Pi, \mathbf{S})$ が定義する (**生成**するともいう) のは Σ 上の語の集合であるが，それは次のように定義される．α が文字列 x の部分語であり ($x = u\alpha v;\ u, v \in (\Gamma \cup \Sigma)^*$ とする)，$\alpha \to \beta$ が Π の書き換え規則であるとき，x の中の α を β に書き換えて文字列 $x' = u\beta v$ が得られるとき，

$$x \Rightarrow x' \quad \text{すなわち} \quad u\alpha v \Rightarrow u\beta v$$

と書く．つまり，$\Rightarrow$ は書き換えの 1 ステップを表わす．また，$\Rightarrow$ を 0 回以上行なって $x_0 \Rightarrow x_1 \Rightarrow \cdots \Rightarrow x_n$ となるとき，途中を省略して

$$x_0 \Rightarrow^* x_n$$

と書く．$\mathbf{S}$ から書き換え $\Rightarrow$ を 0 回以上行なって得られる Σ 上の語の集合

$$L(G) := \{\, x \in \Sigma^* \mid \mathbf{S} \Rightarrow^* x \,\}$$

が，G によって生成される言語 (**文脈自由言語**という) である．

〔例 2.9〕　文脈自由文法・言語

(1)　アルファベット $\{a, b\}$ 上の文字列すべてからなる言語 $\{a, b\}^*$ を生成する文法 $G_1 := (\{S\}, \{a, b\}, \Pi_1, S)$ を考えよう．ただし，

$$\Pi_1 := \{S \to aS,\ S \to bS,\ S \to \lambda\}.$$

$\Pi_1 := \{S \to aS \mid bS \mid \lambda\}$ と略記してもよい．例えば，

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aabS \Rightarrow aab\lambda = aab \quad \text{なので} \quad S \Rightarrow^* aab.$$

$\Pi_1 := \{S \to Sa \mid Sb \mid \lambda\}$ とか $\Pi_1 := \{S \to aS \mid Sb \mid \lambda\}$ なども同じ言語を生

の文字列も許したものを一般に**文法**とか書き換え系という．文法は，言語とは何かを数学的に定義してその数理的性質を研究するために 1950 年代に導入され，自然言語やプログラミング言語などの研究の数理モデルとして研究されたが，現在ではコンピュータサイエンスのための基礎理論になっている．

成する．一方，$\Pi_1 := \{S \to aS \mid bS \mid a \mid b\}$ とすると $\{a,b\}^+$ が生成される．

（2） 例 2.8 (1) の ‘10 進数’ を定義する CFG

$$G_{number} := (\{\mathbf{d}\},\ \{0,1,\ldots,9\},\ \Pi_{number},\ \mathbf{d})$$

ただし，$\Pi_{number} := \{\mathbf{d} \to 0 \mid \cdots \mid 9 \mid \mathbf{d}0 \mid \cdots \mid \mathbf{d}9\}$.

（3） 例 2.8 (4) の ‘数式’ を定義する CFG

$$\begin{aligned} G_{exp} \quad := \quad & (\{\mathbf{E},\mathbf{T},\mathbf{F},\mathbf{p},\mathbf{N},\mathbf{d}\},\\ & \{0,\ldots,9,A,\ldots,Z,a,\ldots,z,+,-,*,/,\uparrow,(,)\},\ \Pi_3,\ \mathbf{E}). \end{aligned}$$

ただし，$\Pi_{exp} := \Pi_{name} \cup \Pi_{number} \cup \Pi_3$ で

$$\begin{aligned} \Pi_3 \quad := \quad & \{\mathbf{E} \to \mathbf{E}+\mathbf{T} \mid \mathbf{E}-\mathbf{T} \mid \mathbf{T},\ \mathbf{T} \to \mathbf{T}*\mathbf{F} \mid \mathbf{T}/\mathbf{F} \mid \mathbf{F},\\ & \mathbf{F} \to \mathbf{p}\uparrow\mathbf{F} \mid \mathbf{p},\ \mathbf{p} \to -\mathbf{p} \mid (\mathbf{E}) \mid \mathbf{N} \mid \mathbf{d}\}. \end{aligned}$$

（4） $G_4 := (\{S\},\ \{a,b\},\ \{S \to aSb \mid \lambda\},\ S)$ は $\{a^n b^n \mid n \geqq 0\}$ を生成する．

（5） $G_5 := (\{S\},\ \Sigma,\ \{S \to aSa,\ S \to b \mid a,b \in \Sigma\},\ S)$ は Σ 上の回文 (前から読んでも後ろから読んでも同じ文字列のこと) の集合を生成する：$L(G_5) = \{xbx^R,\ xx^R \mid x \in \Sigma^*,\ b \in \Sigma\}$. □

記号のまとめ (2 章)

$S(n)$	n の後者 ($S(n) = n+1$)
$\dfrac{\varphi_1\ \varphi_2\ \cdots\ \varphi_n}{\psi}$	仮定 $\varphi_1 \wedge \varphi_2 \wedge \cdots \wedge \varphi_n$ から帰結 ψ が導かれる
$n!$	n の階乗 ($n! = 1 \times 2 \times \cdots \times n$)
$n\ \mathbf{mod}\ m$	n を m で割った余り
$\lvert x \rvert$	語 x の長さ
x^R	語 x の鏡像
D_n	整合した n 対の括弧からなる文字列の集合
$G = (\Gamma, \Sigma, \Pi, \mathbf{S})$	文脈自由文法
$\Rightarrow,\ \Rightarrow^*$	生成の 1 ステップ, 0 ステップ以上
$L(G)$	G が生成する言語
以下，BNF に関するもの	
$<\bigcirc\bigcirc>$	($\bigcirc\bigcirc$を表わす) 超変数
$\alpha ::= \beta$	α を βとして定義する
$\beta_1 \mid \beta_2$	β_1 または β_2
$\{\alpha\}^*$	α の 0 回以上の繰り返し
$\{\alpha\}^+$	α の 1 回以上の繰り返し

2.3 節 理解度確認問題

問 2.15 次の事柄 (または，もの) を BNF で定義せよ．

（1） 10 進数．ただし，0 から始まる 10 進数としては 0 しか許さない．

（2） $\Sigma = \{a, b\}$ 上の文字列 (空語を含める場合と除く場合の両方について)

（3） 数には整数と実数がある．整数は数字を 1 個以上並べた文字列，またはその前に符号 $+, -$ を付けたものである．整数は実数である．実数には整数の他に，符号の付いていない 2 個の整数の中間にピリオド . を書いたもの，その前に符号を付けたもの，こういった実数の後ろに文字 E を書きさらに続けて整数を書いたものがある．

問 2.16 例 2.8 (4) の BNF によって定まる数式に関して答えよ．

（1） $-3 * (4 - 5)$ の構文木を書き，それに従って計算した値を求めよ．

（2） $a - -b$ は数式として許されるか？

問 2.17 (2.11) の解 $\mathbf{N}, \mathbf{A}, \mathbf{L}, \mathbf{D}$ は一意的に定まることを証明せよ．

問 2.18 例 2.8 (4) において，$a + b - c$ は $(a + b) - c$ と計算されるか，または $a + (b - c)$ と計算されるか？ また，$-2 \uparrow 2$, $3 \uparrow 1 \uparrow 2$ の値はいくつか？

問 2.19 例 2.8 の ‘数式’ の定義を変更し，次の 3 通りの BNF で定義する．

文法 1：<数式>::=<変数>｜<定数>｜<数式>+<数式>｜<数式>*<数式>｜(<数式>)

文法 2：<数式>::=<項>+<数式>｜<項>　<項>::=<因子>*<項>｜<因子>　<因子>::=<変数>｜<名前>｜(<数式>)

文法 3：<数式>::=<項>*<数式>｜<項>　<項>::=<因子>+<項>｜<因子>　<因子>::=<変数>｜<名前>｜(<数式>)

（1） 文法 3 に関して $(a + 12) * b + 3$ の構文木を 1 つ示せ．

（2） 文法 1 を CFG で表わし，$(a + 12) * b + 3$ の生成の過程 (導出) を 1 つ示せ．

（3） どの文法がもっとも良いといえるか？ 理由を説明して答えよ．

問 2.20 次の言語を生成する文脈自由文法 (CFG) を示せ．

（1） a^*　（2） D_1 (例 2.6)　（3） $\{a^n b^n c^m \mid n, m \geqq 1\}$

（4） 2 進数 (ただし，0 から始まるものは 0 しか許さない)

（5） $\{a^i b^j c^{i+j} \mid i, j \geqq 0\}$　（6） $\{x \in \{0, 1, 2\}^* \mid x \text{ は } 0 \text{ を } 3 \text{ 個含む}\}$

問 2.21 次の文脈自由文法 (CFG) が生成する言語を求めよ．

（1） $G_1 = (\{X\}, \{0\}, \{X \to 00X \mid 0\}, X)$

（2） $G_2 = (\{X, Y\}, \{1\}, \{X \to XX, X \to YY, Y \to 1\}, X)$

（3） $G_3 = (\{S, B\}, \{a, b\}, \{S \to aBS \mid BaS, B \to b \mid \lambda\}, S)$

（4） $G_4 = (\{A_0, A_1, A_2\}, \{0, 1, \ldots, 9\}, \{A_i \to jA_{(i+j) \bmod 3} \mid 0 \leqq i \leqq 2, 0 \leqq j \leqq 9\}, A_0)$