解　説

属性文法とその応用—I

属性文法の理論入門†

西野　哲朗††

1. はじめに

D. E. Knuthは1968年に，文脈自由言語の意味記述のために，属性文法（attribute grammar）という形式文法を考案した[7]．属性文法とは，文脈自由型プロダクションに意味規則を付加し，文脈自由文法の導出木上で，意味の計算（属性評価）が行えるようにした枠組みである．

今回と第4回の「属性文法の複雑さ」の2回に分けて，属性文法理論の基本的事項を解説する．まず今回は，属性文法の定義から始めて，属性文法の応用システムを実現する際に必要となる，基本的な属性評価アルゴリズムについて解説する．さらに第4回では，このような属性評価アルゴリズムの計算能力や，属性文法が生成する言語の複雑さを計算量の観点から考察する．

実務家の皆さんへ：まず，本誌93年7月号の「理論と実際のギャップ」から，野崎昭弘氏の示唆に富んだ文章を引用させていただく．

> 実際面に明るい方も，理論をバカにせず，新しい理論を建設するのに協力されるとよい，と思う．また，そのためにも理論屋は「やさしい理論を提供する」努力をするべきである．5年もたてば忘れられる「深い定理」より，皆の理解を助ける「基本的な定義」のほうが重要なのだが，と私は思う．

筆者の力不足から，この哲学がどのくらい実践できたかは，はなはだ疑問ですが，本稿においては，属性文法理論のなかでも特に重要な基本事項をなるべく平易に解説した（つもりです）．

その意味で今回の解説は属性文法の基本公式集であり，第4回は細則集であると考えられます．今回の基本公式集をマスターすれば，それぞれの適用分野に属性文法の考え方を持ち込むことは可能と思われます．しかし，現実の複雑な問題を解決しようとする場合には，細則集やそこに挙げられた参考文献と格闘する必要が生じることでしょう．

理論家の皆さんへ：形式言語理論が下火になって久しいですが，今回と第4回で述べる内容についての理論的研究はほぼ終わっていると思われます．しかし，属性文法と関連した新しい研究分野はいろいろと考えられます．たとえば，属性文法の枠組みにオブジェクト指向，並列計算，確率的計算などの仕組みを取り入れたときの計算論や意味論，さらには属性文法の学習理論などです．今回と第4回の解説を属性文法理論の入門コースととらえ，基本的事項を修得してから個別の問題について考え始めて下さい．

なお，以下では頁数の関係で，特に重要なポイントのみを解説します．さらに詳しい内容を知りたい方は，たとえば文献2)，10)などを参照してください．

2. 属性文法とは

まず，次のような**文脈自由文法** G_1 を考えよう．文脈自由文法の詳細については，たとえば，文献4)，8)などを参照されたい．

$G_1=(N, T, P, Z)$,

$N=\{Z, A, B\}$, $T=\{0, 1\}$,

$P=\{Z\rightarrow A,\ A\rightarrow AB,\ A\rightarrow B,\ B\rightarrow 0,\ B\rightarrow 1\}$

ここで，N, T, P, Z はそれぞれ，文法 G_1 の**非終端記号**の集合，**終端記号**の集合，**プロダクション**の集合，および**開始記号**である．文法 G_1 は終端記号列の導出の仕方を記述しており，その導出は開始記号 Z から始まる．たとえば，終端記号列1011は，以下の順序でプロダクションを適用する

† Introduction to Theory of Attribute Grammars by Tetsuro NISHINO (School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku).

†† 北陸先端科学技術大学院大学情報科学研究科

ことにより導出される（導出の1ステップを⇒で表す）.

$Z \Rightarrow A$　　($Z \to A$ の適用)
$\Rightarrow AB$　　($A \to AB$ の適用)
$\Rightarrow ABB$　　($A \to AB$ の適用)
$\Rightarrow ABBB$　　($A \to AB$ の適用)
$\Rightarrow BBBB$　　($A \to B$ の適用)
$\Rightarrow 1BBB$　　($B \to 1$ の適用)
$\Rightarrow 10BB$　　($B \to 0$ の適用)
$\Rightarrow 101B$　　($B \to 1$ の適用)
$\Rightarrow 1011$　　($B \to 1$ の適用)

この導出の様子を表す導出木を**図-1**に示す(図-1の太線部分が導出木を表している). 容易に分かるように, この文法 G_1 は0と1からなる任意の記号列（ビット列）を導出することができる. このとき, 文法 G_1 はビット列の「構文」を定義しているという. さてビット列は2進数として「解釈」することができ, したがって, ある10進数を表現していると考えることができる. たとえば,

$2^3\times1+2^2\times0+2^1\times1+2^0\times1=11$

であるから, ビット列1011は10進数の11を表している. 多くの人にとって, 2進数よりも10進数のほうが表現として自然であるから, ビット列 α を2進数と解釈した場合には, α の10進数としての値が, その「意味」であると考えることができる.

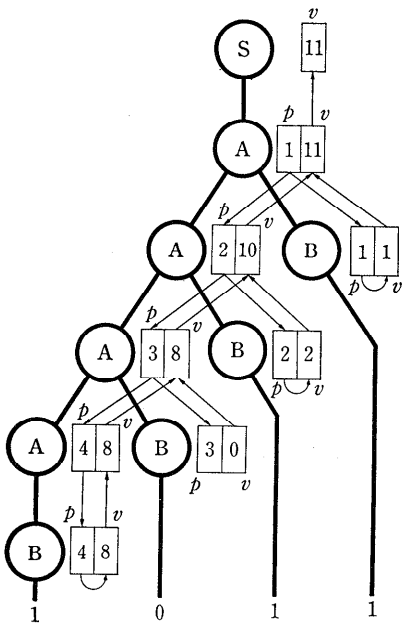


図-1　1011の導出木とその上の属性評価

属性文法とは, 文脈自由文法によって導出される記号列に対し, その「意味」を対応させる規則を定めた文法である. 属性文法は, 文脈自由文法を以下のように2段階で拡張することにより得られる.

1. 文脈自由文法の各非終端記号に**属性**を付加する. 属性とは, 導出木内での意味の計算に用いられる変数である. 属性には**合成属性**と**継承属性**の2種類があるが, その違いについては後で述べる.

2. 文脈自由文法の各プロダクションに, 属性値の計算方法を定める**意味規則**を付加する.

上の文脈自由文法 G_1 を拡張して, 各ビット列に対し, その10進表記の数値を意味として対応させる属性文法 G_2 を構成してみよう. まず, 文法 G_1 の非終端記号 Z, A, B に, 次のような属性を付加する.

非終端記号	合成属性	継承属性
Z	v	な　し
A	v	p
B	v	p

ここで, v は値 (value) を, p は桁 (place) をそれぞれ表現している. 上の表で, たとえば A と B には同じ名前の継承属性 p が付加されているが, 以下では A に付加された属性 p を $p(A)$, B に付加された属性 p を $p(B)$ と表すことにする. 次に, G_1 の各プロダクションに, 属性 v や p の計算方法を規定した意味規則を以下のように付加する.

プロダクション	意　味　規　則
$Z \to A$	$v(Z)=v(A),\ p(A)=1$
$A_1 \to A_2B$	$v(A_1)=v(A_2)+v(B),$ $p(A_2)=p(A_1)+1,$ $p(B)=p(A_1)$
$A \to B$	$p(B)=p(A),\ v(A)=v(B)$
$B \to 0$	$v(B)=0$
$B \to 1$	$v(B)=2^{p(B)-1}$

表中の二つ目のプロダクションを，$A_1 \rightarrow A_2B$ のように A に添字を付けて表記したのは，異なる A の「生起」A_1 と A_2 に付加された同一名の属性 ($v(A_1)$ と $v(A_2)$ など) を区別するためである．

ここで構成した属性文法 G_2 によって，ビット列 1011 からその意味 (10 進数の 11) を得るには，以下のような処理を行う．

1. 終端記号列 1011 の構文解析を行う．すなわち，1011 に対する G_1 の導出木 (図-1 の太線部分) を発見する (構文解析については，たとえば文献 1) などを参照されたい)．

2. 1. の導出木内で用いられているプロダクションに付随した G_2 の意味規則を用いて，導出木内の各頂点に付随している属性の値を評価する (このプロセスを**属性評価**という)．

3. 導出木の根に付随した合成属性 v の値が，求める 10 進数 11 である．

上で述べた処理の様子を図-1 に示す．図-1 から分かるように，継承属性はその値が導出木内を下向きに伝達される属性であり，合成属性はその値が導出木内を上向きに伝達される属性である．

3. 属性文法の定義

Σ を**アルファベット**，すなわち記号の有限集合とする．このとき，Σ^* で Σ の記号からなる有限長の記号列全体の集合を表す．また，集合 Σ の要素数を $|\Sigma|$ で表すものとする．

定義 3.1 **文脈自由文法**とは，以下を満たす 4 項組 $G'=(N, T, P, Z)$ のことをいう．すなわち，N, T, P はそれぞれ G' の**非終端記号**，**終端記号**および**プロダクション**の集合であり，Z は**開始記号**である．また，P のプロダクション p は

$$p: X_0 \rightarrow w_0X_1w_1 \cdots X_nw_n \quad (n \geq 0)$$

という形をしている．ただし，$X_i \in N$, $w_j \in T^*$ $(0 \leq j \leq n)$ とする． □

二つの記号列 $u, v \in (N \cup T)^*$ に対し，P のある一つのプロダクションを適用して，u を v に書き換えることができるならば，$u \Rightarrow v$ と表す．また，P のプロダクションを 0 回以上の有限回適用して u を v に書き換えることができるならば，$u \overset{*}{\Rightarrow} v$ と表す．

文脈自由文法 G' が**既約**であるとは，次の 2 条件が満たされることをいう．

- すべての $X \in N$ に対し，$u, v \in (N \cup T)^*$ が存在して，$Z \overset{*}{\Rightarrow} uXv$ となる．
- すべての $X \in N-\{Z\}$ に対し，$w \in T^*$ が存在して，$X \overset{*}{\Rightarrow} w$ となる．

すなわち，G' が既約ならば，G' のすべての非終端記号は開始記号から到達可能であり，かつ，終端記号列を生成することができる．言い換えれば，G' には無用な非終端記号は存在しないことになる．

また，文脈自由文法 G' が**サイクルなし** (cycle-free) であるとは，どの $X \in N$ に対しても，$X \overset{+}{\Rightarrow} X$ となることがないときをいう．今回の解説を通じ，現れる文脈自由文法は，すべて既約かつサイクルなしであると仮定する．

定義 3.2 **属性文法** G とは，以下の 1〜3 を満たす 3 項組 (G', A, F) のことをいう．

1. $G'=(N, T, P, Z)$ は既約な文脈自由文法であり，G の**基底文脈自由文法**と呼ばれる．

2. G' の各非終端記号 X に対し，交わりをもたない二つの有限集合 $I(X)$ と $S(X)$ が付随している．$I(X)$ の要素を**継承属性**，$S(X)$ の要素を**合成属性**という．X の属性全体の集合を $A(X)$ で表す．すなわち，$A(X)=I(X) \cup S(X)$ である．$A=\cup_{X \in N} A(X)$ と定義し，A を G の**属性集合**と呼ぶ．X の属性 a を $a(X)$ と表す．また，属性 a が取りうる値の集合を $V(a)$ と表す．

3. P の各プロダクション

$$p: X_0 \rightarrow w_0X_1w_1 \cdots X_nw_n \quad (n \geq 0)$$

に対し，

$$S(X_0) \cup I(X_1) \cup \cdots \cup I(X_n)$$

に含まれるすべての属性を定義するような，**意味規則**の集合 F_p が付随している．属性 $a(X_k)(0 \leq k \leq n)$ を定義する意味規則は，次のような形をしている．

$$a(X_k) := f(a_1(X_{i_1}), \cdots, a_m(X_{i_m})) \quad (*)$$

ただし，f は以下の型の関数とする．

$$V(a_1(X_{i_1})) \times \cdots \times V(a_m(X_{i_m})) \rightarrow V(a(X_k))$$

上の式 (*) が成り立っているとき，$a(X_k)$ は p において $a_1(X_{i_1}), \cdots, a_m(X_{i_m})$ に**依存**しているという．$F=\cup_{p \in P} F_p$ と定義し，F を G の**意味規則集合**と呼ぶ． □

上のように属性文法 G を定義する場合，通常，以下のような仮定が置かれる．

- すべての $X \in N$ に対し，$A(X) \neq \emptyset$，すなわち，どの非終端記号 X にも，少なくとも一つの

属性が付随している.

● $I(Z)=\emptyset$ かつ $|S(Z)|=1$, すなわち, 開始記号 Z には, ただ一つの合成属性が付随している.

● G の任意の意味規則 (*) において, 右辺の各属性 $a_j(X_{ij})$ $(1\leq j\leq m)$ は,

$$I(X_0)\cup S(X_1)\cup\cdots\cup S(X_n)$$

の要素である. この条件が満たされるとき, 意味規則 (*) は **Bochmann 正規形**であるという.

練習問題 1　2. の属性文法 G_2 が, 定義 3.2 の条件をすべて満たしていることを確認せよ.

属性文法のプロダクション

$$p: X_0\to w_0X_1w_1\cdots X_nw_n \quad (n\geq 0)$$

において, 集合

$$\{a(X_j)\mid a\in A(X_j), 0\leq j\leq n\}$$

の各要素を p の**属性生起**と呼ぶ. すなわち, p の左辺または右辺に現れる非終端記号に付随した属性すべてを, p の属性生起という. 次の点に注意されたい. もし p において X_k と X_l が同一の非終端記号だった場合には, 当然 $A(X_k)=A(X_l)$ であり, X_k に属性 a が付随していれば, X_l にも属性 a が付随している. これら二つの属性は a という同一の属性名をもつが, 属性生起としては, それぞれ $a(X_k)$ と $a(X_l)$ であり, 異なっている.

4. 非循環属性文法と絶対非循環属性文法

属性文法 G は, G の任意の導出木において, その各頂点に付随した属性値が循環定義に陥らないときに**非循環**であるという. 任意の導出木において, その頂点に付随したすべての属性値が求まるためには, 属性文法は非循環でなければならない. したがって, 属性文法が与えられたときに, その非循環性を判定する問題は大変重要である. しかし, この判定問題は, 解くために指数時間を必要とする非常に難しい問題であることが知られている[5].

そこで以下では, 非循環性の定義を強めて, 絶対非循環属性文法を定義する. 属性文法が与えられたときに, その絶対非循環性の判定は多項式時間で行うことができる[6]. まず, いくつかの概念を定義する. 属性文法の

$$p: X_0\to w_0X_1w_1\cdots X_nw_n \quad (n\geq 0)$$

なる形のプロダクション p に現れる属性生起全体の集合を, A_p で表すものとする.

定義 4.1　属性生起間の**直接依存関係** DDO (Direct Dependencies between attribute Occurrences) を次のように定義する.

$$DDO=\bigcup_{p\in P} DDO_p$$

ただし, $DDO_p=\{(a(X_i), b(X_j))\mid a(X_i)$ に依存して $b(X_j)$ を定義する意味関数が F_p 内に存在する$\}\subseteq A_p\times A_p$* とする. □

定義 4.2　属性生起間の**誘導依存関係** IDO (Induced Dependencies between attribute Occurrences) を次のように定義する.

$$IDO=\bigcup_{p\in P} IDO_p$$

ただし, $IDO_p=DDO_p\cup\{(a(X_i), b(X_i))\mid i>0, q\in P$ が存在して, $q: X_i\to w$, かつ, $(a(X_i), b(X_i))\in IDO_q^+\}\subseteq A_p\times A_p$ とする. ここで, IDO_q^+ は関係 IDO_q の推移的閉包**を表す. □

属性文法 G は, 各プロダクション $p\in P$ に対し, 関係 IDO_p が非循環的であるときに, **絶対非循環**であるという. p が使われている G のすべての導出木において, IDO_p のすべての 2 項関係が同時に現れることはないかもしれないので, 絶対非循環性は意味規則の非循環性を強めに評価していると考えることができる. すなわち属性文法は, 絶対非循環ならば必ず非循環であるが, 非循環だとしても絶対非循環であるとは限らない.

例 4.1　**2.** の属性文法 G_2 について考える. G_2 の 5 つのプロダクションに, $p_1: Z\to A$, $p_2: A_1\to A_2B$, $p_3: A\to B$, $p_4: B\to 0$, $p_5: B\to 1$ と名前を付ける. このとき, まず G_2 の属性生起間の直接依存関係は次のようになる.

$$DDO_{p1}=\{(v(A), v(Z))\},$$
$$DDO_{p2}=\{(v(A_2), v(A_1)), (v(B), v(A_1)), (p(A_1), p(A_2)), (p(A_1), p(B))\},$$
$$DDO_{p3}=\{(p(A), p(B)), (v(B), v(A))\},$$
$$DDO_{p4}=\emptyset,\quad DDO_{p5}=\{(p(B), v(B))\}$$

さらに, G_2 の属性生起間の誘導依存関係は次のようになる.

$$IDO_{p1}=DDO_{p1}\cup\{(p(A), v(A))\},$$
$$IDO_{p2}=DDO_{p2}\cup\{(p(A_2), v(A_2)),$$

---

* A_p の直積空間を表している.

** 二つの関係 $R: A\to B$, $S: B\to C$ に対し, R と S の**合成** $S\cdot R$ を次のように定義する: $S\cdot R=\{(a, c)\mid aRb, bSc$ なる $b\in B$ が存在する$\}\subset A\times C$. さらに, 関係 R の**べき乗**を次のように定義する: $R^1=R, \cdots, R^{n+1}=R^n\cdot R$. このとき, 関係 R^+ を以下のように定義し, R の**推移的閉包**と呼ぶ: $R^+=\cup_{n\geq 1}R^n$. ここでは, $A=B=A_p$, $R=IDO_q$ と考えればよい.

$(p(B), v(B))\}$,

$IDO_{p3}=DDO_{p3}\cup\{(p(B), v(B))\}$,

$IDO_{p4}=\emptyset$, $IDO_{p5}=DDO_{p5}$

上の5つの関係はすべて非循環的なので，G_2 は絶対非循環である． □

多くの応用で実際に用いられている属性文法は，ほとんど絶対非循環のようであるが，以下では，最も一般的な非循環属性文法に対する属性評価アルゴリズムについて解説する．

5. 純粋多重戦略属性文法

本章では，**純粋多重訪問属性文法**，**純粋多重スイープ属性文法**，**純粋多重交互パス属性文法**，および**純粋多重パス属性文法**を紹介する．これら4つの属性文法のクラスを統一的に定義するために，**図-2** の非決定性手続き NEVAL (Nondeterministic EVALuation, 非決定性評価) を用いる．手続き NEVAL では，型 **strategy**＝{visit, sweep, rpass, lpass} を用いており，変数 Y を **strategy** 型として宣言している（すなわち，$Y\in$**strategy** である）．まず，NEVAL について説明する．

N_0 をある導出木 t 内の頂点とし，そのラベルを X_0 とする．t 内で N_0 の子頂点を導出するのに用いられたプロダクションを，

$$X_0\rightarrow w_0X_1w_1\cdots X_nw_n$$

とする．ここで，$X_0, \cdots, X_n$ は非終端記号であり，$w_0, \cdots, w_n$ は終端記号列であったことに注意する．つまり，X_i は非終端記号1文字であり，w_i はいくつかの終端記号からなる記号列である．たとえば，$w_0=10$, $X_1=A$ であったとすると，X_0 の子頂点のうち左から三つのものは，順に 1, 0, A をラベルとする．

X_i でラベル付けされた頂点は内部頂点であるから，それ自身が再び子頂点をもつが，終端記号でラベル付けされた頂点は葉頂点なので子頂点をもたない．したがって，非終端記号 $X_1, \cdots, X_n$ でラベル付けされた内部頂点のみが属性をもつので，属性評価においては，これらの内部頂点のみを訪れればよい．

属性評価における N_0 への1回の**訪問** (visit) とは，以下のことをいう．

1. (NEVAL の2行目) N_0 を訪れて N_0 のいくつかの継承属性の値を評価する．

2. (NEVAL の3行目) N_0 のいくつかの内部子頂点を任意の順番で訪れるために，訪れる子頂点の個数と位置を決める．具体的には，m 個の数 $v_1, \cdots, v_m$ を任意に選ぶ (数 m も任意に選ぶ)．ただし，$1\leq v_i\leq n\,(1\leq i\leq m)$ とする．これらの数は，7行目で X_{v_i} という形で，N_0 の内部子頂点を指定するのに用いられる．

3. (NEVAL の7行目) N_0 の m 個の内部子頂点を3行目で指定した順番に訪れて，そこで属性評価を行う．7行目の操作は再帰的に書かれているので，実際には，N_0 を根とする t の部分木 t' 内を巡回して，NEVAL を再帰的に適用していくことになる．

4. (NEVAL の8行目) 再び N_0 に戻ってき

```
Procedure NEVAL (Y, N0); strategy Y; node N0;
{N0 に適用されたプロダクションを X0→w0X1w1…XnwnとするX}
  begin
1   count (N0) := count (N0)+1;
2   X0 のいくつかの継承属性の値を求める; {非決定性ステップ}
    case Y of
3       visit: m 組 v=(v1, …, vm) を一つ任意に選ぶ．ただし，m≥0 かつ，
                各 1≤i≤m に対し 1≤vi≤n とする; {非決定性ステップ}
4     sweep: n 組 (1, …, n) の一つの順列 v=(v1, …, vn) を任意に選び，
                m := n とする; {非決定性ステップ}
5       lpass: v=(1, …, n) とし，m := n とする;
6       rpass: v=(n, …, 1) とし，m := n とする
    end
7   for i := 1 to m do NEVAL (Y, Xvi);
8   X0 のいくつかの合成属性の値を求める; {非決定性ステップ}
  end
PureVisit {純粋多重訪問属性評価用メインプログラム}
  begin
    while count (root)<k0 do NEVAL (visit, root)
  end
PureSweep {純粋多重スイープ属性評価用メインプログラム}
  begin
    for c := 1 to k do NEVAL (sweep, root)
  end
PurcAltPass {純粋多重交互パス属性評価用メインプログラム}
  begin
    k 組 d=(d1, …, dk) を任意に選ぶ．
      ただし，di∈{rpass, lpass} (1<i<k) とする; {非決定性ステップ}
    for c := 1 to k do NEVAL (dc, root)
  end
PurePass {純粋多重パス属性評価用メインプログラム}
  begin
    for c := 1 to k do NEVAL (lpass, root)
  end
```

図-2 純粋多重属性評価手続き

て，N_0のいくつかの合成属性の値を評価し，最後にN_0を出て他の頂点を訪れる．3行目の指定の仕方から，この後，N_0の親や兄弟またはN_0自身を訪れることができる．

N_0に対する1回の**スイープ** (sweep) とは，訪問よりも制限された頂点の巡回方法であり，上の2.のステップを以下のように変更することにより定義される．

(NEVAL の4行目) N_0 の n 個の内部子頂点すべてを1回ずつ訪れるために，子頂点を訪れる順番を任意に決める．具体的には，$m=n$ に固定し，$v_1, \cdots, v_m$ の任意の順列を選ぶ．ただし，$1 \leq v_i \leq n$ $(1 \leq i \leq m)$ とする (m を任意に選べない点が訪問よりも制限されている)．

N_0に対する**左パス** (left pass) による1回の巡回は，スイープよりもさらに制限された頂点の巡回方法であり，上の2.のステップを以下のように変更することにより定義される．

(NEVAL の5行目) N_0 の n 個の内部子頂点すべてを左から順番に1回ずつ訪れる．具体的には，$m=n$ に固定し，$v_1=1, \cdots, v_n=n$ とする ($v_1, \cdots, v_m$ の順列を任意に選べない点が，スイープよりも制限されている)．

N_0に対する**右パス** (right pass) による1回の巡回は，左パスによる巡回と類似の頂点の巡回方法であり，上の2.のステップを以下のように変更することにより定義される．

(NEVAL の6行目) N_0 の n 個の内部子頂点すべてを右から順番に1回ずつ訪れる．具体的には，$m=n$ に固定し，$v_1=n, \cdots, v_n=1$ とする．

上で述べた手続き NEVAL は再帰的に定義されているので，属性文法 G の導出木 t の根頂点 root に対して，たとえば NEVAL (visit, root) という形で呼び出すと，t のすべての頂点を1回ずつ訪問する．このような頂点の巡回の最中に，NEVAL の 2, 8 行目に従って，各頂点において属性評価が行われる．2, 8 行目では「いくつかの属性の値を求める」となっているが，巡回の回数はなるべく少ないほうが良いので，通常は「その時点で評価可能なすべての属性値を求める」とする．

NEVAL では，導出木内の頂点 X を訪れた回数を，count (X) という変数に保持している．属性文法 G は，G の各導出木 t に対し，t の各ノード X において count (X) の値を k より大きくせずに (つまり X を k 回以上訪れずに)，図-2 の手続き PureVisit によって t 内のすべての属性値を求めることができるときに，**純粋 k 訪問**であるという．属性文法は，それがある $k \geq 1$ に対して純粋 k 訪問であるときに，**純粋多重訪問**であるという．純粋多重訪問属性文法のクラスは，非循環属性文法のクラスと一致することが知られている[9]．

同様に，属性文法 G は，G の任意の導出木 t 内のすべての属性値を，t の各ノード X において count (X) の値を k より大きくせずに，手続き PureSweep, PureAltPass, または PurePass によって求めることができるときに，それぞれ，**純粋 k スイープ**，**純粋 k 交互パス**，または**純粋 k パス**であるという．属性文法は，ある $k \geq 1$ に対して純粋 k スイープ，純粋 k 交互パス，または純粋 k パスであるときに，それぞれ**純粋多重スイープ**，**純粋多重交互パス**，または，**純粋多重パス**であるという．

練習問題 2　**2.**の属性文法 G_2 について考える．記号列 1011 に対する導出木に対して，図-2 の手続き PureVisit, PureSweep, PureAltPass, PurePass を実際に動作させてみよ．机上でシミュレーションを行ってもよいし，計算機プログラムを実行してもよい．

6. 単純多重戦略属性文法

本章では，前章と同様な方法で，**単純多重訪問属性文法**，**単純多重スイープ属性文法**，**単純多重交互パス属性文法**，および**単純多重パス属性文法**を定義する．単純多重訪問属性文法は，純粋多重訪問属性文法における非決定性のステップが，すべて決定性となるように制限を加えることにより得られる．スイープ，多重交互パス，多重パスについても同様である．これら4つの属性文法のクラスを統一的に定義するために，**図-3** の決定性手続き DEVAL (Deterministic EVALuation) を用いる．

属性文法 G は，次の条件を満たすとき**単純 k 訪問**であるという．

(1) G の各非終端記号 X に対し，その

属性集合 $A(X)$ の分割 $A_1(X), \cdots, A_{l(X)}(X), l(X) \leq k$ が存在し，かつ

(2) G の各プロダクション $X_0 \rightarrow w_0X_1w_1 \cdots X_nw_n$ と $1 \leq j \leq l(X_0)$ に対し，訪問列 $v_j=(v_j(1), \cdots, v_j(m_j))$ が存在して（ただし，$m_j \geq 0$ であり，かつ，各 $1 \leq i \leq m_j$ に対し，$1 \leq v_j(i) \leq n$ とする），

図-3 に示す手続き SimpleVisit により，(1)の分割と(2)の訪問列を用いて，G の任意の導出木のすべての属性値を評価することができる．

単純 k 訪問の場合，G の導出木内の属性評価においては，ラベルが X の頂点 N を j 回目 $(1 \leq j \leq l(X))$ に訪れたときには，集合 $A_j(X)$ に含まれる属性をすべて評価しなければならない（図-3参照）．したがって今回の場合には，NEVAL (図-2) の 2, 8行目のように，頂点を訪れるごとに評価する属性を任意に選ぶことはできず，この部分の非決定性が排除されている．

また，(2)の条件から，ラベルが X の頂点 N を j 回目 $(1 \leq j \leq l(X))$ に訪れたときには，N の内部子頂点をあらかじめ定まっている訪問列 $v_j=(v_j(1), \cdots, v_j(m_j))$ で指定された順番に訪れなければならない．したがって今回の場合には，NEVAL (図-2) の3行目のように，訪れる頂点の個数や，頂点を訪れる順番を任意に決めることはできず，この部分の非決定性も排除されている．

その他の単純巡回戦略も同様に定義できる．属性文法 G は，次の条件を満たすとき**単純 k スイープ**であるという．

(1) G の各非終端記号 X に対し，その属性集合 $A(X)$ の分割 $A_1(X), \cdots, A_k(X)$ が存在し，

(2) G の各プロダクション $X_0 \rightarrow w_0X_1w_1 \cdots X_nw_n$ と $1 \leq j \leq k$ に対し，系列 $(1, \cdots, n)$ の順列である訪問列 $v_j=(v_j(1), \cdots v_j(n))$ が存在して，

図-3 に示す手続き SimpleSweep により，(1)の分割と(2)の訪問列を用いて，G の任意の導出木のすべての属性を評価することができる．

```
Procedure DEVAL (Y, N0); strategy Y; node N0;
{N0 に適用されたプロダクションを X0→w0X1w1…XnwnとするX}
  begin
1   count (N0) := count (N0)+1;
2   j := count(X0); {Y≠visit のときは不要}
3   Aj(X0) のすべての継承属性の値を求める;
    case Y of
4       visit: vj=(vj(1), …, vj(mj)). ただし, mj≥0 かつ,
               各 1≤i≤mj に対し 1≤vj(i)≤n とする;
5     sweep: n 組 vj=(vj(1), …, vj(n)) は (1, …, n) の一つの順列とし,
               mj := n とする;
6      lpass: vj=(1, …, n) とし, mj := n とする;
7      rpass: vj=(n, …, 1) とし, mj := n とする
    end
8   for i := 1 to mj do DEVAL (Y, Xv_j(i));
9   Aj(X0) のすべての合成属性の値を求める
  end

SimpleVisit {単純多重訪問属性評価用メインプログラム}
  begin
    while count (root)<k (root) do DEVAL (visit, root)
  end

SimpleSweep {単純多重スイープ属性評価用メインプログラム}
  begin
    for c := 1 to k do DEVAL (sweep, root)
  end

SimpleAltPass {単純多重交互パス属性評価用メインプログラム}
  begin
    for c := 1 to k do DEVAL (dc, root)
  end

SimplePass {単純多重パス属性評価用メインプログラム}
  begin
    for c :=1 to k do DEVAL (lpass, root)
  end
```

図-3 単純多重属性評価手続き

属性文法 G は，次の条件を満たすとき**単純 k 交互パス**であるという．

(1) G の各非終端記号 X に対し，その属性集合 $A(X)$ の分割 $A_1(X), \cdots, A_k(X)$ が存在し，

(2) 系列 $d=(d_1, \cdots, d_k)$, $d_i \in \{\text{lpass, rpass}\}$ が存在して，

図-3 に示す手続き SimpleAltPass により，(1)の分割と(2)の系列 d を用いて，G の任意の導出木のすべての属性を評価することができる．

属性文法 G は，G の各非終端記号 X に対し，その属性集合 $A(X)$ の分割 $A_1(X), \cdots, A_k(X)$ が存在し，図-3 に示す手続き SimplePass により，G の任意の導出木のすべての属性を評価することができるときに，**単純 k パス**であるという．

属性文法は，ある $k \geq 1$ に対して単純 k 訪問，単純 k スイープ，単純 k 交互パス，または単純 k

パスであるときに，それぞれ**単純多重訪問**，**単純多重スイープ**，**単純多重交互パス**，または，**単純多重パス**であるという．

練習問題 3　2.の属性文法 G_2 について考える．記号列 1011 に対する導出木に対して，図-3 の手続き SimpleVisit, SimpleSweep, SimpleAltPass, SimplePass を実際に動作させてみよ．机上でシミュレーションを行ってもよいし，計算機プログラムを実行してもよい．その際，属性の分割と訪問列をいろいろに設定してみよ．

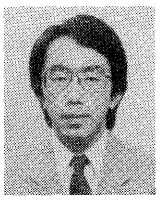
7. おわりに

本稿では，主要な属性文法のクラスを，対応する属性評価戦略に基づいて定義した．読者のなかには，これらの属性文法が表現能力の点でどのように異なるのかということに，疑問をもたれた方も多いのではないかと思う．そのことについては，本連載の **4.**「属性文法の複雑さ」で詳しく解説する予定である．とりあえず今回は，属性文法の定義と，基本的な属性評価戦略について理解していただければ幸いである．

参考文献

1) Aho, A. V. and Ullman, J. D.: *The Theory of Parsing, Translation and Compiling*, Prentice-Hall Inc., Englewood Cliffs (1972).
2) Deransart, P., Jourdan, M. and Lorho, B.: Attribute Grammars—Definitions, Systems and Bibliography, *Lecture Notes in Computer Science* 323, Springer (1988).
3) Engelfriet, J. and Filé, G.: Passes, Sweeps and Visits, *Lecture Notes in Computer Science* 115, Springer, pp. 193-207 (1981).
4) Hopcroft, J. E. and Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley (1979).
5) Jazayeri, M., Ogden, W. and Rounds, W.: The Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars, *Comm. ACM* 18, pp. 697-706 (1975).
6) Kennedy, K. and Warren, S. K.: Automatic Generation of Efficient Evaluators for Attribute Grammars, *Proc. 3rd ACM Symp. on POPL*, pp. 32-49 (1976).
7) Knuth, D. E.: Semantics of Context-Free Languages, *Mathematical Systems Theory* 2, pp. 127-145 (1968). Correction: *ibid.* 5, pp. 95-96 (1971).
8) Moll, R. N., Arbib, M. A. and Kfoury, A. J.: *An Introduction to Formal Language Theory*, Springer-Verlag (1988).
9) Riis, H. and Skyum, S.: *k*-Visit Attribute Grammars, *Math. Syst. Theory*, Vol. 15, pp. 17-28 (1981).
10) 佐々政孝：チュートリアル—属性文法，コンピュータソフトウェア，Vol. 3, pp. 73-91 (1986).

(平成5年9月16日受付)



西野　哲朗（正会員）

昭和34年生．昭和57年早稲田大学理工学部数学科卒業．昭和59年同大学院理工学研究科博士前期課程修了．同年日本アイ・ビー・エム(株)入社．昭和62年東京電機大学理工学部情報科学科助手．平成4年北陸先端科学技術大学院大学助教授．理学博士．属性文法，自然言語の基礎理論，計算量理論などの研究に従事．電子情報通信学会，日本ソフトウェア科学会，LA 各会員．