順位グラフ文法によるプログラム仕様書の定式化

Formalization of Software Documents Using A Precedence Graph Grammar

有田 友和　　杉田 公生　　松本 亮

日本大学大学院総合基礎科学研究科 地球情報数理科学専攻

東海大学理学部数学科

日本大学大学院理工学研究科 情報科学専攻

あらまし 表形式のグラフの生成文法について考察する．色々な文法の中で順位グラフ文法が表形式を定義する効率よい文法であることが知られている．表形式の定型用紙は作成事項の確認・参照・管理の点で文字列形式より優れているため，実用上定型文書のフォーマットとして良く利用されている．我々はプログラム仕様書システム Hiform96 を表形式により定義し，プログラミング教育で利用している．本論文では順位グラフ文法によりこれらを形式的に定義することを試みる．

キーワード 順位グラフ文法，Hiform96，仕様書様式

1 はじめに

高品質なソフトウエアプログラムの開発を効率的に行うための近年の研究においてはソフトウエアプロセスの記述法と支援システムは基本的な問題であることが認識されている．とくに，ソフトウエアプログラム仕様記述支援環境は PSL/PSA(Problem Statement Language and Problem Statement Analyzer:1970 〜) 等多くの研究がなされてきた．さらに近年では品質保証の国際規格 ISO-9000 シリーズや電子化統合事務文書システム CALS(Computer-aided Acquisition and Logistics Support) における仕様書そのものの管理文書としての持つ役目の重要性が強調されている．その中で仕様書記述支援システムの厳密な定義は世界的な問題になっており言語理論により，仕様書記述支援システムを定式化することは重要と思われる．そこで我々は仕様書の様式間の順序の定式化に文字列文法を適用する定型用紙 Hiform96 を提案した．そのなかで杉田らは仕様書を表形式の１７種類の様式の集まりにして，各様式１ページに仕様書の各項目や内容などを収めた．[6,7,8]

一般に, 順序を持ったプログラム言語や表言語などのように順序を持った対象は構文チェック，翻訳，編集に対して言語理論的方法が有効と考えられている．しかし現在のところ，仕様書言語に対してグラフ言語理論を適用する研究はそれほど多くはない．

一方、仕様書等の順序付けられた表言語の形式的な定義にはグラフ文法が有効と思われる．複合グラフに関して，Franck は順位グラフ文法を提案した．[1] で順位グラフ文法は線形時間で構文解析可能であることが示されている．

仕様書の様式は大別して「項目の並びと大きさを固定するもの」と「項目の並びと大きさを自由にしたもの」に分けることができる. 前者は固定することによる自由度の低さ, 後者は自由度が高すぎるため正しく仕様書を定められない，描画が困難になるなどの問題点がある．本研究では２つの中間を考え，順位グラフ文法により定式化することで項目の並びと大きさを自由にした上で正しい仕様書が厳密に定まり，描画が正確に行えるようにすることを目的とする．[6]

本論文では，2.1 節で Hiform96[6,7,8] の説明をする．2.2 節で順位グラフ文法 [1] 等の定義を解説し，さらに 3 節にこれらの Hiform96 の様式への応用に示した．巻末の付録 1 に構文解析アルゴリズム [1]，付録２に様式 A1，B2 とそれを表わすグラフ，さらに A1 に対する生成規則, 順位関係表を示した．

2 準備

2.1 プログラム仕様書 [6,7,8]

Hiform96 の特徴

プログラム仕様書様式の中で表形式の定型用紙を用いたものは作成 ・参照・管理・教育等の様々な使用目的に適している．プログラミング教育を目的に開発した表形式の仕様書システムが Hiform96 であり，仕様書の作成・管理機能をＧＵＩを用いて使用することができる．記入項目について ISO6592-1985(対応規格 JISX0126 − 1987：応用システム文書化要領) のガイドラインにある項目を用いている．[3,5]

次にプログラム仕様書の記述形式は視覚的な表形式による記述形式を採用する．仕様書全体を表形式にするためには仕様書の各項目を内容や開発プロセスを考慮して A4 サイズ１ページに収まるよう再編成した個々の定型用紙が Hiform96 の基本構成単位である．Hiform96 では 17 種類の定型用紙を定義する．

Hiform 96 の様式

カテゴリＡ：プログラム文書 (全 6 様式) このカテゴリには主に JISX0126 付属文書「プログラム文書化要領」に関するものである．

カテゴリＢ：データ文書 (全 3 様式) このカテゴリには JISX0126 付属書２「データ文書化要領」に関するものである．このカテゴリではデータに関する詳細な記述が行われる．

カテゴリＣ：作業手順文書 (全 6 様式) 第三のカテゴリは JISX0126 付属書３「作業手順文書化要領」に関するものである．

カテゴリＤ：プログラム構造図 (全 2 様式)JISX0126 付属書１の内容の内でプログラム構造については別に記述するように指示してある．しかしプログラム構造は再帰的な構造を成す可能性があるためにカテゴリＡのプログラム文書の様式では記述が不可能である．そのためのカテゴリ D である．

2.2 順位グラフ文法 [1]

マーク付きグラフ文法・文脈自由グラフ文法

定義 1 マーク付きグラフ (marked graph) は $G = (K, R, k, r)$ の 4 項組である．ただし，K は **頂点**の有限集合 ($K \neq \phi$)，R は **有向辺**の有限集合 ($\subset K \times K$)，V は **頂点のマーク**の有限集合，M は **辺のラベル**の有限集合，k は $\mathbf{K} \rightarrow \mathbf{V}$ **への関数** (頂点をマーク付けする写像)，r は $\mathbf{R} \rightarrow \mathbf{M}$ **への関数** (辺をラベル付けする写像) である．

□

定義 2 文脈自由生成規則 (context-free production) は $p = (A, H, p^e, p^s)$ の 4 項組である．ただし，A は **一つの頂点のグラフ** (p の左辺という)，$H = (K_h, R_h, k_h, r_h)$ は **空でないグラフ** (p の右辺という)，p^e と p^s は $\mathbf{M}$ **から** $\mathbf{K_h}$ **への部分関数**である．

□

定義 32 つのマーク付きグラフ $G = (K, R, k, r)$,$G' =$

(K', R', k', r') および生成規則 $p = (A, H, p^e, p^s)$ に対して，関係 $G \overset{\Rightarrow}{}_p G'$ が成立するとき，G の A によりラベル付けされた頂点および A を始端 (p^e) または終端 (p^s) とする辺を置き換えることで G' は G から**導出** (derivation) されるという．すなわち

・$k(i) = A$ であるような $i \in K$ が存在する

・$K' = (K - i) \bigcup K_h$

・$k'(a) = \begin{cases} a \in K \text{ に対し } k(a) \\ a \in K_h \text{に対し } k'(a) \end{cases}$

・$R' = R - \{(a,b) | a = i \text{ 又は } b = i\} \bigcup R_h$

$\bigcup \{(a,x) | m \in M \text{ に対し}, \bigvee_{(a,i) \in R} r((a,i) = m \text{ かつ } p_e(m) = x \in K_h\}$

$\bigcup \{(x,b) | m \in M \text{ に対し}, \bigvee_{(i,b) \in R} r((i,b) = m \text{ かつ } p_s(m) = x \in K_h\}$

・$r'((a,b)) = \begin{cases} (a,b) \in R \text{ に対し } r((a,b)) \\ (a,b) \in R_h \text{に対し } r_h((a,b)) \\ R' \text{で新たに追加された辺については} \\ G \text{ の辺のラベルを継承する．} \end{cases}$

が成り立つことである．

□

例 1 ラベル X でマーク付けされた 1 つの頂点で構成されるグラフ G(左辺) に対して，生成規則 $p = (X, H, p^e, p^s)$ を生成規則の例を図 1 に示す．グラフ H(右辺) はラベル A, B, C, D でマーク付けられた 4 つの頂点で構成されるグラフである．左辺と右辺は p^e, p^s を表す点線矢印でつながっている．m でラベル付けされた H の頂点 i から頂点 X への矢印は $p^s(m) = i$，頂点 X から H の頂点 i への矢印は $p^e(m) = i$ である．

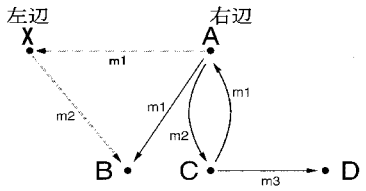

図 0.

定義 4 文脈自由グラフ文法 (a context-free graph grammar) は $GG = (V, T, M, P, S)$ の 5 項組である．だだし，V は**頂点をマーク付けする記号**の有限集合，T は**終端記号** $(\subset V)$，M は**辺のラベル**の有限集合，P は**生成規則** $p = (A, H, p^e, p^s)$ の有限集合，S は**開始記号** $(\in V - T)$ である．

□

定義 5 グラフ文法 $GG = \{V, T, M, P, S\}$ に対して

1.$p \in P$ に対する関係 $\overset{\Rightarrow}{}_p$ の反射推移閉包を $\overset{\Rightarrow}{}_{GG}$ で表す．

2.$S \overset{\Rightarrow}{}_{GG} G$ のとき G は GG によって**導出される**という．

3.G が GG の**グラフ言語に属する**とは G が GG によって導出されかつ G のすべての頂点が終端記号によりマーク付けられていることである．

□

定義 6 グラフ文法 GG の**構文木** (parse tree) は次の条件を満たす木である．

(i) 根は開始グラフに対する生成規則によりラベル付けされる

(ii) 頂点 $1, 2, \ldots, n$ のラベルがちょうど生成規則 p の右辺の非終端記号 $A_1, A_2, \ldots, A_n$ になっているとき部分写像 $f : \{1, 2, \ldots, n\} \to \{s_1, s_2, \ldots, s_n\}$ が存在する．ただし，s_i は A_i を左辺に持つ生成規則である．

□

定義 7 構文木の**産物** (yield) は以下のように再帰的に定義される

(i) 木の根 p が葉ならば木の産物は p の右辺である．

(ii) 木の根 p が葉でないとき，構文木中の p の子を $p_1, p_2, \ldots, p_n$ とおく．p の右辺の非終端記号を $A_1, A_2, \ldots, A_n$ とする．$p_j (1 \leq j \leq n)$ の構文木中の深さは p より小さいので，産物 $G_j (1 \leq j \leq n)$ を持つ．次に，アルゴリズム 0 により $A_1, A_2, \ldots, A_n$ を $G_1, G_2, \ldots, G_n$ に置き換えて埋め込みを行ったものが p の産物である．

□

アルゴリズム 0．辺の埋め込み

[入力]:辺を構成する頂点 a, b および辺のラベル rel

[出力]:置き換えた辺 (x, rel, y) を埋め込む

[手続き]

procedure *additional edge(a,rel,b);*
node a,b;edgelabel rel;
begin node *x,y;x:=a;y:=b;*
while *there exists a replacement of(x)*
do *//x* **を生成規則適用による置き換えとする**
$p = (A, H, p^e, p^s)$;
$x := p^s(rel)$
od;
while *there exists a replacement of (y)*
do *//y* **を生成規則適用による置き換えとする**
$p = (B, F, q^e, q^s)$;
$x := q^e(rel)$
od;
add to the graph the edge(x,rel,y)
end;

□

定義 8 グラフ文法は**曖昧でない** (unambiguous) とはすべての導出可能なグラフ G に対して、G を産出する構文木がただ一つ存在することである．

□

順位関係・順位グラフ文法

記法 9 $m \in M$ に対し，

$$\doteq_m \overset{def}{=} \left\{ (A,B) \middle| \begin{array}{l} A, B \in V \text{ かつ辺 } (x,y) \text{ を持つ} \\ \text{規則が存在する．ただし } x, y \\ \text{はそれぞれ } A, B \text{ によりマーク} \\ \text{付けされ } (x,y) \text{ はラベル } m \end{array} \right.$$

□

記法 10 $m \in M$ に対し，

$$\rightarrow_m \overset{def}{=} \left\{ (A,B) \middle| \begin{array}{l} A, B \in V \text{ かつ} \\ \text{生成規則 } p = (A, H, p^e, p^s) \\ \text{が存在かつ } B \text{ は } H \text{ の頂点} \\ p^e(m) \text{ のマークである} \end{array} \right.$$

□

記法 11 $m \in M$ に対し，

$< \cdot_m \overset{def}{=} \doteq_m \cdot \overset{+}{\rightarrow}_m$ ただし+は推移的閉包を表す

□

記法 12 $m \in M$ に対し，

$$\leftarrow_m \overset{def}{=} \left\{ (B,A) \middle| \begin{array}{l} A, B \in V \text{ かつ} \\ \text{生成規則 } p = (A, H, p^e, p^s) \\ \text{が存在かつ } B \text{ は } H \text{ の頂点} \\ p^s(m) \text{ のマークである} \end{array} \right.$$

□

記法 13 $m \in M$ に対し，

$\cdot >_m \overset{def}{=} \overset{+}{\leftarrow}_m \cdot \doteq_m$

□

記法 14 $m \in M$ に対し，

$< \cdot > \overset{def}{=} \overset{+}{\leftarrow}_m \cdot \doteq_m \cdot \overset{+}{\rightarrow}_m$

□

定義 15 グラフ文法が**矛盾しない** (conflictless) とは すべての $m \in M$ に対して関係 $< \cdot_m, \doteq_m, \cdot >_m, < \cdot >_m$ が互いに素であることである

□

定義 16 順位関係 (precedence relation) は以下のように定義される

$< \cdot \overset{def}{=} \bigcup_{m \in M} < \cdot_m$　　$\cdot > \overset{def}{=} \bigcup_{m \in M} \cdot >_m$

$< \cdot > \overset{def}{=} \bigcup_{m \in M} < \cdot >_m$　　$\doteq \overset{def}{=} \bigcup_{m \in M} \doteq_m$

□

定義 17 グラフ文法が**順位グラフ文法** (a precedence graph grammar) であるとは次の 3 つを満たすことである．

(i) 順位関係に矛盾がない．

(ii) 各還元が一意に定まる．
(iii) 文法において再帰的非終端記号が存在しない．

□

ハンドル, 順位ハンドル
定義 18 GG をグラフ文法,G をグラフとする．G の**ハンドル** (handle) は (i),(ii) を満たす部分グラフ $U \subset G$ である．
(i) U と H と同型であるような GG における $p = (A, H, p^e, p^s)$ が存在する．
(ii) U と G の残余グラフ間のすべての辺は p の埋め込みにより存在する. すなわち U の頂点 a と残余グラフ内の頂点 b に対し, m でラベル付けされた辺 (a,b) は $a = p^s(m)$，辺 (b,a) は $a = p^e(m)$ である．

□

定義 19 G をグラフ,$U \subset G$ を少なくとも 1 つ頂点を持つ部分グラフとする．U が**順位ハンドル** (precedence handle) であるとはすべての頂点に対し,A, B でマーク付けられた頂点 a, b とラベル m の辺 (a,b) が以下の条件を満たすことである．
(i) a, b が U に属するならば $(A, B) \in \doteq_m$
(ii)a が U に属し,b が属さないならば $(A, B) \in \cdot >_m \cup < \cdot >_m$
(iii)b が U に属し,a が属さないならば $(A, B) \in < \cdot_m \cup < \cdot >_m$

□

定理 20 すべての順位グラフ文法は**曖昧でない** (unambiguous)

□

次に生成規則および順位関係で定義される文法を辺と頂点の数に依存する線形時間で構文解析するアルゴリズム [1] を付録 1 に示す.

3 結果

ここでは 2 節で解説した文脈自由グラフ文法，順位関係，順位グラフ文法，構文解析アルゴリズムを仕様書 Hiform96 の各様式に応用する．Hiform96 生成文法定義の順序としてはまず第一に 3.1 節で Hiform96 の様式を導出する文脈グラフ自由文法を決定づける生成規則の作成を行う．第二に 3.2 節で作成した生成規則に対し，順位関係を定める．最後に 3.3 節で構文解析アルゴリズムを用い，実際に作成した生成規則で定義される文法により導出される Hiform96 の様式が線形時間で構文解析されるか検証する．

3.1 文脈自由生成規則

様式 A1 を導出する生成規則は短時間で構文解析が完了するように規則数を少なくし, 項目間の位置関係を示す Label の数を記述能力を残した上で最小限と考えられる 3 つ in，over(ov)，left of(lf) にした．また各様式には同じ項目を含んでいる部分すなわち共通部分がある．よって様式 A1 だけでなく他の様式との対応も考え，共通部分 (head) と独自部分 (body) にすることを考慮にいれ作成した．下に例として共通部分および独自部分の図を示す.

プロジェクトコード:	A
プログラム名:	プログラム概要書
ライブラリ登録コード:	版名:
著作者:	初版発行日:
文書責任者:	現行版発行日:

図 1．共通部分 (head)

キーワード:	CR 分類コード:
目的・範囲:	
背景情報:	
記述言語:	所要ソフトウエア:
操作:	所要ハードウエア:
関連文書:	
機能:	
例:	

図 2．独自部分 (body)

以上を考慮に入れて作成した生成規則表および規則で定義される文脈自由グラフ文法から様式 A1 が導出を表す構文木を合せて付録 2 の図 7,8,9 に示す．なお，A1 に対する我々の生成規則は 34 種である．

3.2 順位関係表

作成した生成規則に基づき，定義 16 を元に順位関係を定義した．この作業から作成した生成規則が正しいか否かも合せて判定することができる．順位関係表を付録 2 の表 1 にまとめる．なお，A1 終端記号間および共通部分終端記号間の順位関係は同じになったためスペースの都合上 1 つ載せた．終端記号すべてを列挙した順位関係表のサイズは 24 × 72，20 × 60，7 × 21 である．

3.3 構文解析の流れ

付録 1 の構文解析アルゴリズム [1] が様式 A1 を表すグラフが構文解析する様子を付録 2 の図 10 示す．ただし，スペースの都合上共通部分 (head) のみを示した．なお様式 A1 および A1 を表すグラフを様式と合わせて付録 2 の図 3,4,5,6 に示す

主張
Hiform96 の様式 A1 は生成規則 (図 7,8) で生成可能である．逆に，生成規則と順位関係表 (表 1) により A1 は構文解析可能である．B2 に関しても同様である．

4 今後の課題

以上で Hiform96 の様式 A1 が作成した生成規則 (図 7,8) で定義される文法によって導出でき，かつその導出グラフの構文解析は解析すべきグラフの辺と頂点の数に線形依存する時間でなされることを示した．現段階の生成規則では Hiform96 の様式を構文解析できることがわかった．しかし 2 次元の表のような格子状のグラフを扱う場合が残っている．今後の研究課題として (1) この問題の解決，(2) 項目の大きさと位置を属性文法を用いて定めること，そして (3) 最終目標である描画，文法に基づいての編集，計算機への実装などがあげられる．

謝辞

御指導をいただいた日本大学 戸川隼人教授, 同 夜久竹夫教授, 東洋大学 安達由洋 助教授, 同 土田賢省助教授に感謝をいたします.

参考文献

[1] Reinhold Franck;A Class of Linearly Parsable Graph Grammars;Acta Infomatica 10,1978

[2] John E Hopcroft,Jeffrey D Ulman; Introduction to Automata theory,Languages,and Computaion;Addison-Wesley,1979

[3] ISO6592-1985 Guidelines for the documentation of computer-based application systems;1985

[4] A.V. エイホ,J.D. ウルマン著, 土居範久訳; コンパイラ; 培風館,1986

[5] JIS X 0126-1987 応用システムの文書化要綱;1987

[6] 夜久竹夫, 杉田公生, ニ木原吉, 守屋悦郎;Hichat とプログラム開発環 境 ETA_AIDE; 構造エディタ (原田編)P165-P182 共立出版,1987

[7] K.Sugita,Y.Adachi,Y,Miyadera,K,Tsuchida,T.Yaku; Advanced Software Mechanisms for Computer-Aided Instruction information Literacy APEC-CIL'97,1997

[8] 杉田公生, 安達由洋, 土田賢省, 夜久竹夫; 表形式による仕様書記述支 援法とその支援系; プレプリント 1997

付録1. 構文解析アルゴリズム[1]

アルゴリズム 1. 構文解析
[入力]:解析すべきグラフ
[出力]:グラフの導出を記述する構文木
[手続き]

```
procedure precedence analysis(G);graph G;
begin  // 変数宣言および初期化
        graph diagram:=G,
        treenode parse tree := none,
        node a:=G の任意の 1 頂点;
    while diagram G が開始グラフでない
    do reduce handle with(a,parse tree) od;
    parse tree を構築
end;
```

□

アルゴリズム 2. ハンドルの初期化・連続完備化・還元
[入力]
①第一引数に頂点 a をハンドルを還元させるために入力する.
②第二引数にその時点での構文木を構築していく.
[出力]:ハンドルを還元し, 発生した頂点を tree に加える
[手続き]

```
procedure reduce handle with(a,tree);
            reference graphnode a;
            reference treenode tree;
begin graphnode node:=a,node of plateau;
    edgelabel actual label;
    boolean handle incomplete:=true,
            increasing precedence:=false,
            edge pointing to node;
    treenode newtree:=
            新たに treenode を作成・初期化;
    graph plateau:=空のグラフ;
    insert as successor of(newtree,tree);
    while increasing precedence
                or handle incomplete
    do if increasing precedence
       then treenode successor:=none;
            reduce along(node of plateau,actual label
            ,node,edge pointing to node,successor);
       fi;
       add to (plateau,node);
       edge analysis (plateau,node of plateau,
       acutual label,node,edge pointing to node,
       increasing precedence,handle incomplete)
    od;
    search for the suitable rule
                -and mark(newtree);
    部分グラフ plateau を a に還元;
    tree:=newtree
end;
```

□

アルゴリズム 3. 辺の解析
[入力]:*plateau* を第一引数
[出力]:すべての頂点と残余グラフ間の辺の解析結果
・node of plateau(*plateau* 内の頂点)
・actual label(ラベル)
・outside node(*plateau* 外の頂点)
・edge starting in plateau
(*plateau* 内の頂点を始点とする辺か T or F)
・increasing precedence
(辺で順位が増加しているか T or F)
・handle incomplete(ハンドルが完全か T or F)
[手続き]

```
procedure edge analysis(plateau,node of plateau,
        acutual label,outside node
        ,edge starting in plateau,
        increasing precedence,handle incomplete);
        graph plateau;
        reference graphnode node of plateau,
                    outside node;
        reference edge label acutual label;
        reference boolean edge starting in plateau,
                increasing precedence:=false,
                handle incomplete:=false;
begin
   while
  ∀plateau の頂点 A および plateau でない頂点 B に
  対し, ラベル m を持つ辺 (A,B) が存在する
    do if (A, B) ∈�doteq
       then node of plateau:=A;actual label:= m;
          outside node :=B;
          edge starting in plateau:=
                handle incomeplete:=true;
       else if (A, B) ∈<・
       then node of plateau:=A;actual label:= m;
            outside node :=B;
            edge starting in plateau:=
            increasing precedence:=true;
            goto exit
       fi
   od;
   while
  ∀plateau の頂点 A および plateau でない頂点 B に
  対し, ラベル m を持つ辺 (A,B) が存在する
    do if (B, A) ∈�doteq
       then node of plateau:=A;
            actual label:= m;
            outside node :=B;
            edge starting in plateau:=
            handle incomeplete:=true;
       else if (B, A) ∈・>
       then node of plateau:=A;
            actual label:= m;
            outside node :=B;
            edge starting in plateau:=false;
                increasing precedence:=true;
            goto exit
       fi
   od;
exit;
end;
```

□

アルゴリズム 4. 順位増加している辺の再帰的還元
[入力]:*node1,m,edge from 1 to 2*
[出力]:*node2,subtree*
[手続き]

```
procedure reduce along(node1,m,node2,
                edge from 1 to 2,
                subtree);
        graphnode node1;
        edgelabel m;
        reference graphnode node2;
        boolean edge from 1 to 2;
        reference treenode subtree;
begin graphnode node:=node2;
    if edge from 1 to 2;
    then while relation of edge(node1,node)∈<・m;
         do reduce handle with(node,subtree) od;
    else while relation of edge(node,node1)∈・>m;
         do reduce handle with(node,subtree) od;
    fi;
    node2:=node;
end;
```

□

付録 2. 順位関係表，生成規則，構文木，構文解析過程図

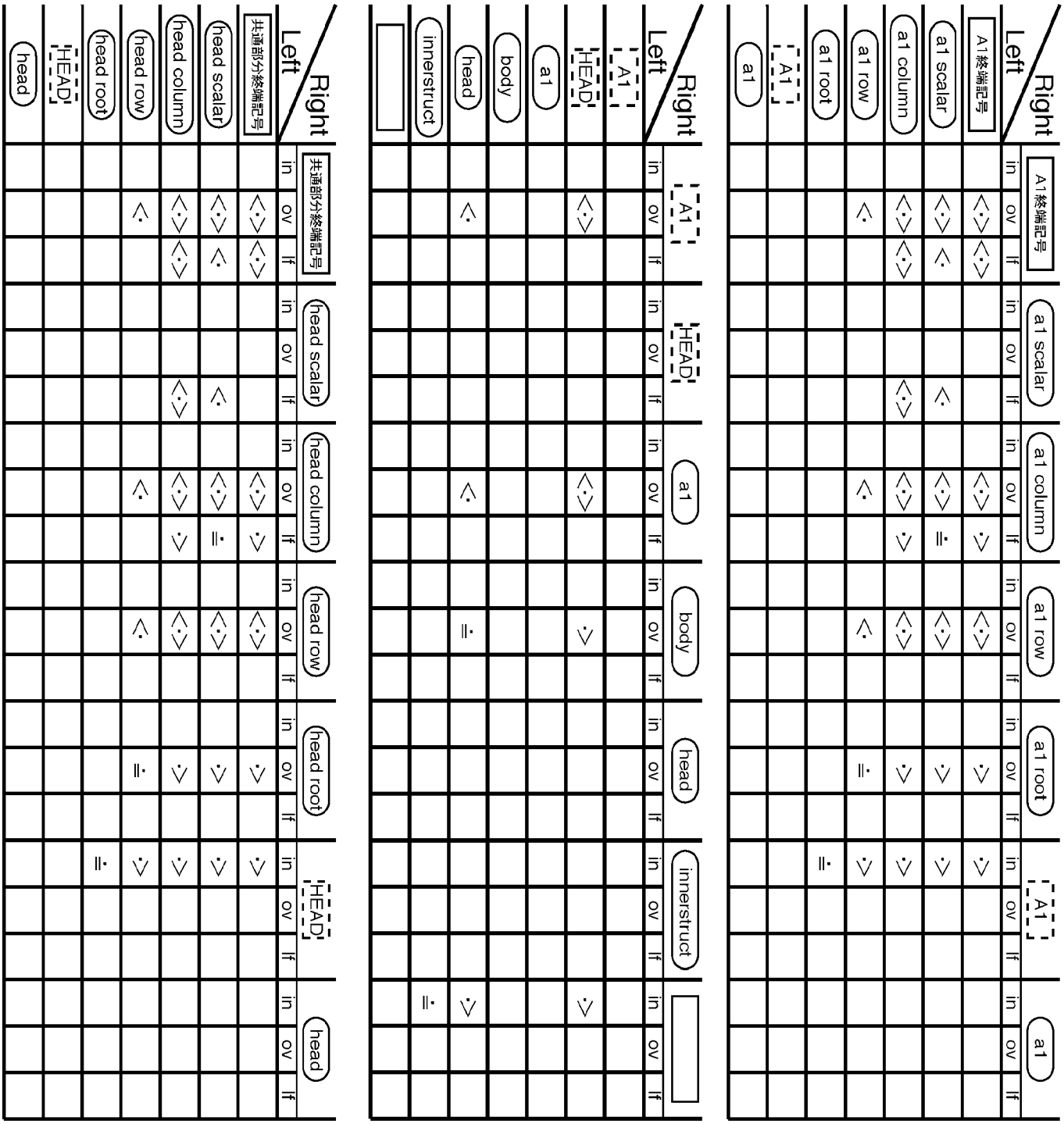

表 1. 順位関係表

プロジェクトコード:	A
プログラム名:	プログラム概要書
ライブラリ登録コード:	版名:
著作者:	初版発行日:
文書責任者:	現行版発行日:
キーワード:	CR 分類コード:
目的・範囲:	
背景情報:	
記述言語:	所要ソフトウエア:
操作:	所要ハードウエア:
関連文書:	
機能:	
例:	

図３．様式 A1 プログラム文書概要書

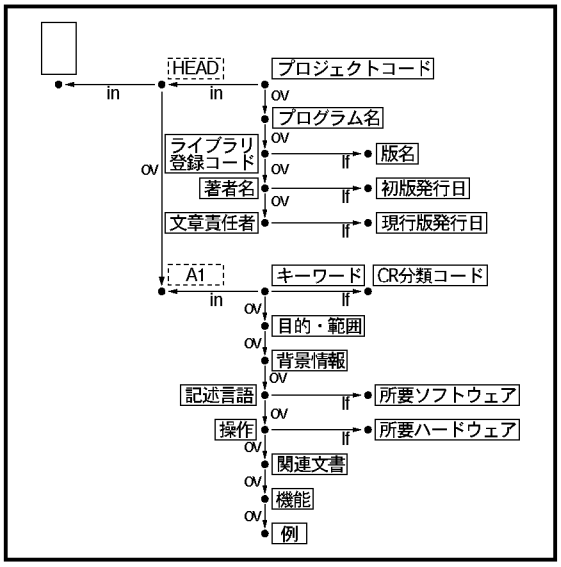

図４．様式 A1 を表すグラフ

プロジェクトコード:	B
プログラム名:	プログラム仕様書 2
ライブラリ登録コード:	版名:
著作者:	初版発行日:
文書責任者:	現行版発行日:
有効期限:	傍系
文書適用範囲:	
発生の権限:	
読み出しの権限	
改良の権限	
伝達の権限	
使用の権限	
データの管理	
適用業務に応じた責任	
組織上の責任	
技術上の責任	
保管上の責任	
データ保管条件 (方法, 場所, 目的, 期限, 頻度)	
回復方法 (手順・作業量)	
暗号化	
使用及び使用方法	

図５．様式 B2 プログラム仕様書 2

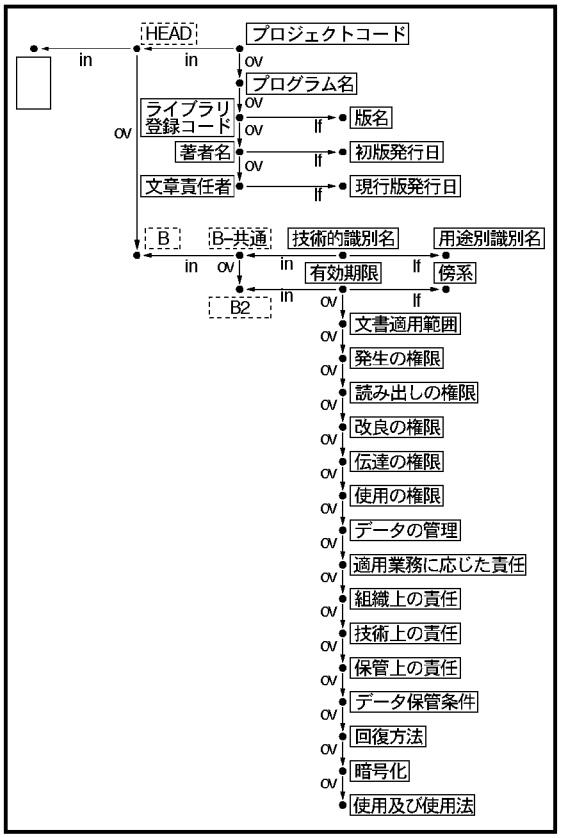

図６．様式 B2 を表すグラフ

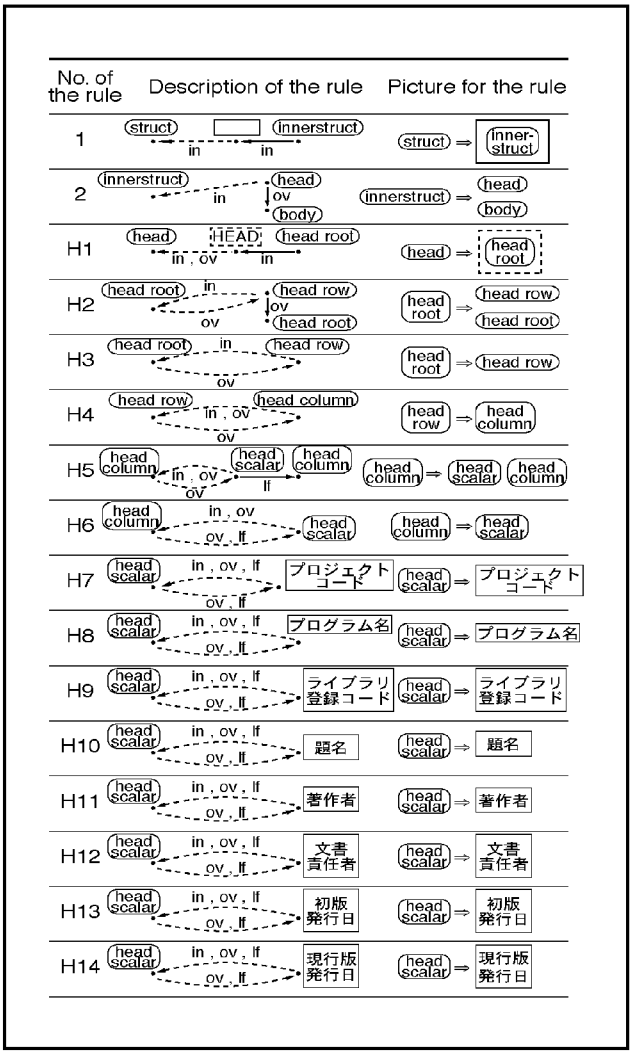

図7 生成規則－共通部分

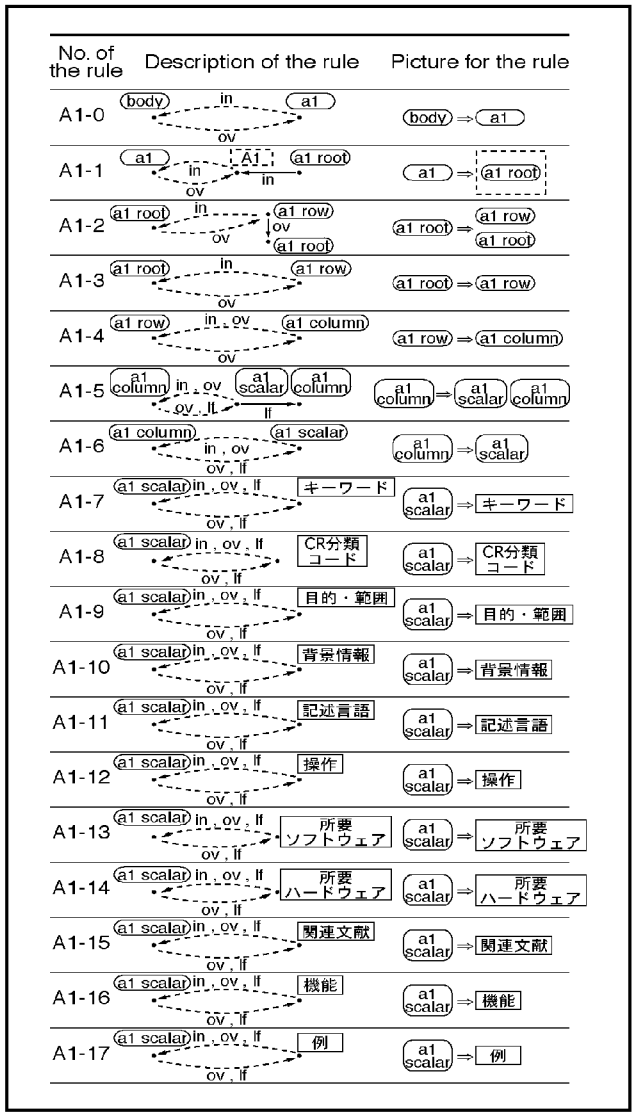

図8 生成規則－様式A1

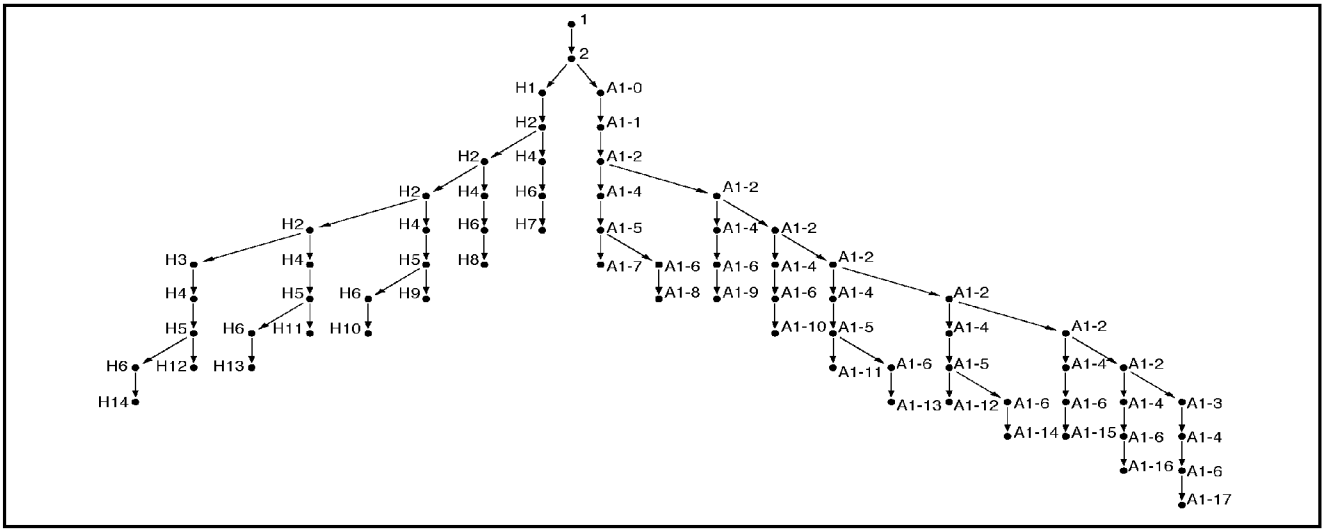

図9 様式A1の構文木

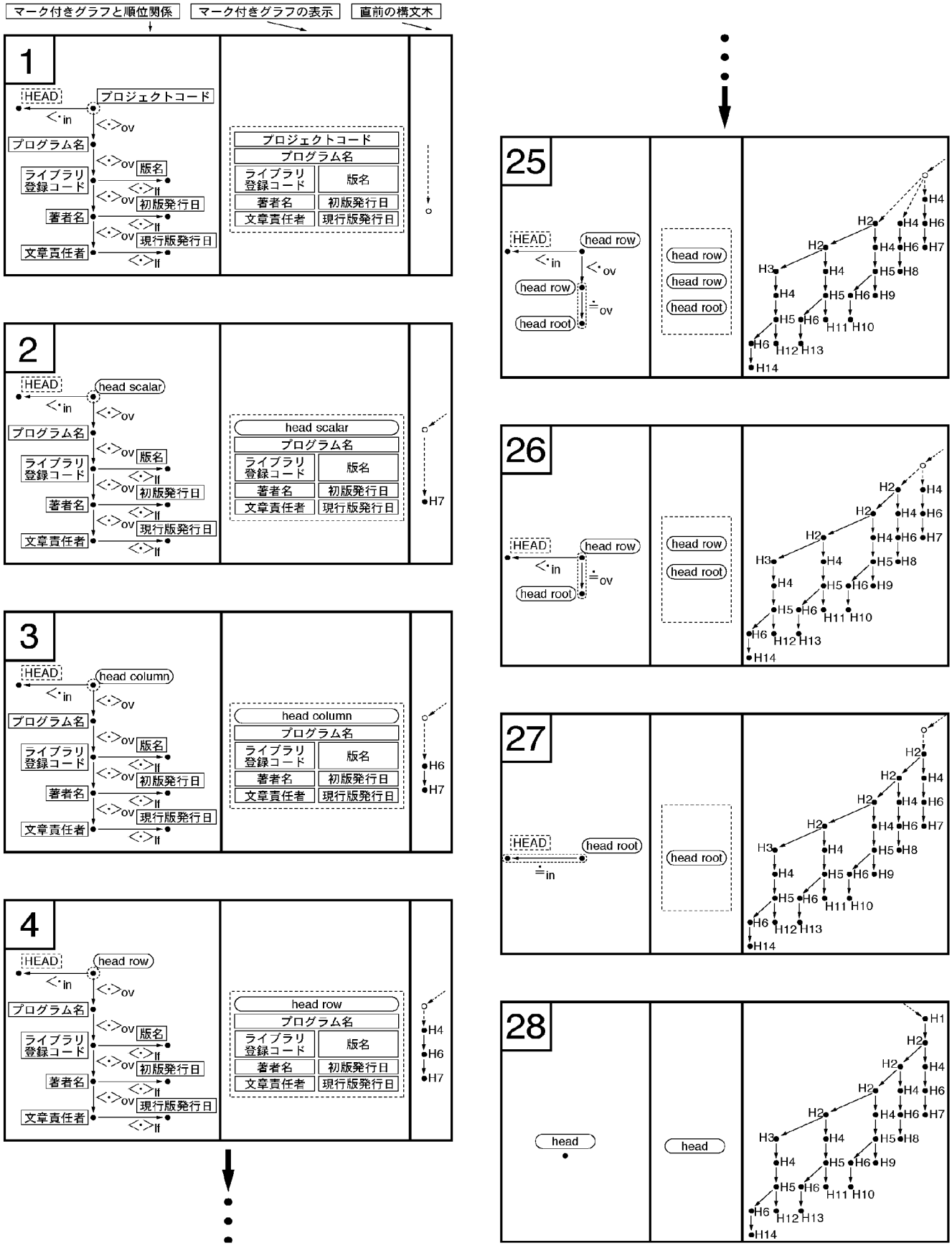
マーク付きグラフと順位関係
マーク付きグラフの表示
直前の構文木
1
HEAD
プロジェクトコード
in
ov
プログラム名
版名
ライブラリ登録コード
初版発行日
著者名
現行版発行日
文章責任者
2
head scalar
H7
3
head column
H6
H7
4
head row
H4
H6
H7
25
head row
head root
26
27
28
head
H1
H2
H3
H4
H5
H6
H7
H8
H9
H10
H11
H12
H13
H14

図１０．構文解析過程図