数理モデル化と問題解決 33－4
（2001．3．15）

Prologプログラム図の属性グラフ文法に基づく定式化

安達由洋†

プログラムやその仕様を可視化したプログラム図は，ソフトウェアの開発や教育のための役立つ道具である．我々はPrologプログラムの実行過程を判りやすく可視化するプログラム図を開発してLogichartを名づけた．本論文では，Logichartの生成規則を定式化したLogichart属性グラフ文法について論じる．この文法はLogichart図のグラフ構文規則を定式化したedNCEグラフ文法と，Logichart図を描画するために必要な情報を属性値として計算する意味規則の集合からなる．そして，この意味規則にはある描画制約条件のもとで面積最小となるLogichart図のレイアウト規則が埋め込まれている．Logichart属性グラフ文法に基づいてPrologプログラムの実行過程を可視化するL-VISシステムを実現して，本文法の有効性を確認している．

Formalization of Prolog Program Diagrams on the Basis of an Attribute Graph Grammar

YOSHIHIRO ADACHI†

Program diagrams visualizing programs or program specifications are very useful in software development and education. We have developed an intelligible program diagram called Logichart for Prolog visualization. Its syntax rules and layout rules are formalized as a Logichart attribute graph grammar. This grammar is underlain by an NCE graph grammar whose productions are defined to formalize the graph-syntax rules of Logichart diagrams. Semantic rules attached to the grammar's productions are defined in such a way that they can extract the layout information needed to display a Logichart diagram by means of attributes atttached to the nodes of graphs derived by the grammar. The semantic rules are formalized so as to get Logichart diagrams of the least area under a specific drawing condition. We have implemented our Prolog visualization system (called L-VIS) on the basis of the Logichart attribute graph grammar.

1. はじめに

Prologプログラムとその実行過程を視認性よく可視化するプログラム図記述言語Logichart (Logic flowchart)は，Prologプログラムの開発や教育のための役立つ道具である．Logichartで生成されるLogichart図は，図式的には木構造図[1]として定義され，プログラム記述の観点からはPrologの構文規則と標準的なPrologインタプリタの計算規則(最左ゴール優先実行，深さ優先探索)を考慮して定義されている．この結果，Logichart図はユーザがPrologプログラムとの対応を容易に取ることができて，理解しやすいものとなっている．

本論文では，Logichartの生成規則を厳密に定式化した属性グラフ文法（Logichart属性グラフ文法）を定義する．このLogichart属性グラフ文法は，Logichart図のグラフ構文規則を定式化したedNCEグラフ文法[2]と，Logichart図を描画するために必要な情報を属性値として導出する意味規則の集合からなる．そして，意味規則の評価順序はedNCEグラフ文法が生成する順序グラフとしてのノードの線形順序を用いて明示的に定義されている．また，この意味規則にはある特定の条件のもとで面積が最小となるLogichart図を描画するレイアウト規則が埋め込まれている．

Pascalや DXLなどの手続き型プログラムに対しては，そのプログラム図の構文規則を属性グラフ文法を用いて定式化する研究が既に幾つか報告されている[3),4)]．しかしながら，これらの研究では文法の意味規則を評価する順序が明示的に定式化されておらず，属性評価手続きを実現する際に順次属性値が決るように計算順序を定める必要がある．また，Logichart図の描画条件は文献3)あるいは文献4)で示された手続き型プログラム図の描画条件と全く異っている．その結果，Logichart属性グラフ文法の意味規則も従来の手続き型言語に対する属性グラフ文法の意味規則と異っている．

2. Prologプログラムの表示

2.1 Logichart図

Logichart図を用いたPrologプログラムの可視化は次のように行われる．ユーザの質問文のゴール列 '$G_1, G_2, \cdots, G_n$' に対して，節 'prolog_program :- $G_1, G_2, \cdots, G_n$.' をプログラムに追加する．この節のヘッドに対応する

† 東洋大学工学部
Faculty of Engineering, Toyo University

'prolog_program' の文字列を含むラベルの付いたノードが Logichart 図のルートとなる．Logichart 図では節を構成するヘッドとボディのゴール列は，Prolog の構文規則に準じて水平方向に左から右に一列に配置される．一方，ゴールおよびそのゴールとユニフィケーション可能なヘッドを持つ節はプログラム内での節の順序で垂直方向に上から下に向かって一列に配置される．このように Logichart は Prolog の構文規則と標準的な Prolog インタプリタの計算規則 (最左ゴール優先実行，深さ優先探索) を考慮して定義されている．その結果，Logichart 図で可視化するとプログラム構造の理解が容易になり，元の Prolog プログラムとの対応も容易にとることができる．また，Logichart 図には 'prolog_program' をルートとするユーザの質問文に対する実行木が部分木として必ず含まれる．

次のプログラムに対して，'?- test.' を入力したときの Logichart 図を図 1に示す．

```
test(X,Y,Z) :- appendList(X,Y,Z),
      write((X,Y,Z)),nl.
test(_,_,_) :- write(end),nl.
appendList([],X,X).
appendList([X|L1],L2,[X|List]) :-
      appendList(L1,L2,List).
```

2.2 Logichart 図の描画条件

Logichart 図を見やすく描画するためのノードの配置条件を木構造図のレイアウト制約条件として議論する．

L- 木構造図 $T = (V, E, L_E, width, depth)$ とはエッジラベルを持つ木構造図であり，かつ (V, E) が 2 分木である．ここで，V はノードの集合，E はエッジの集合，そして $L_E : E \to \{h, v\}$ はエッジラベリング関数である．写像 $width : V \to R$ は幅関数である．ノード p の水平方向の長さを $width(p)$ で表し，ノードの幅という．また，写像 $depth : V \to R$ は深さ関数である．ノード p の垂直方向の長さを $depth(p)$ で表し，ノードの深さという．

L- 木構造図 T に対する配置とは対 (T, π) のことである．ただし，$\pi : \{\text{T のノード}\} \to R \times R$ である．T のノード p に対して配置 π はノードの左上の角を R^2 の点に写す．p が写された点の座標を $\pi(p) = (x, y)$ で表す．また，$\pi_x(p)$ と $\pi_x(p)$ によりそれぞれ $\pi(p)$ の x 座標と y 座標を表す．

π で配置された L- 木構造図 T の幅 $width(T, \pi)$，深さ $depth(T, \pi)$ および面積 $area(T, \pi)$ を次のように定義する．

$$width(T, \pi) = max\{\pi_x(p) + width(p) - \pi_x(q) \mid p \text{と} q \text{は} T \text{のノード}\}$$

$$depth(T, \pi) = max\{\pi_x(p) + depth(p) - \pi_x(q) \mid p \text{と} q \text{は} T \text{のノード}\}$$

$$area(T, \pi) = width(T, \pi) \times depth(T, \pi)$$

L- 木構造図のノード s と t について s から t へ向かうラベル v のエッジが存在するとき，t は s の v- 子供という．また，s から t へ向かうラベル h のエッジが存在するとき，t は s の h- 子供という．s から t へ向かうラベル v を持つエッジだけからなる有限の長さのパスが存在するとき，t は s の v- 子孫 という．同様に，s から t へ向かうラベル h を持つエッジだけからなる有限の長さのパスが存在するとき，t は s の h- 子孫 という．

L- 木構造図のノード s，s の v- 子供 t，s から t への v エッジ，および t をルートノードとする L- 木構造図の部分木構造図から構成される木構造図を，s をルートノードとする v- 部分木構造図という．ただし，s に v- 子供がない場合は，s 自体を s をルートノードとする v- 部分木構造図という．同様に，s，s の h- 子供 t，s から t への h エッジ，および t をルートノードとする L- 木構造図の部分木構造図から構成される木構造図を，s をルートノードとする h- 部分木構造図という．ただし，s に h- 子供がない場合は，s 自体が s をルートノードとする h- 部分木構造図である．特に，ラベル h を持つ入力エッジのないノードをヘッドノードといい，ラベル h を持つ入力エッジのあるノードをゴールノードという．

L- 木構造図では，常にラベル h を持つエッジを水平方向に，ラベル v を持つエッジを垂直方向に描くようにする．以下に L- 木構造図のノード配置に関するレイアウト制約条件を示す．ただし，x 軸を水平方向右向きに，y 軸を垂直方向下向きにとっている．また，$GapX$ と $GapY$ は定数である．

条件 B_1: ノード t が s の h- 子孫のとき，$\pi_y(s) = \pi_y(t)$.

条件 B_2: セル t が s の v- 子孫のとき，$\pi_x(s) = \pi_x(t)$.

条件 B_3: s がゴールノードのとき，$\pi_y(\text{s の v- 子供}) \geq \pi_y(s) + depth(s) + GapY$.

条件 B_4: s がヘッドノードのとき，$\pi_y(\text{s の v- 子供}) \geq max\{\pi_y(t) + depth(t) \mid t \text{ は } s \text{ の } h\text{- 部分木構造図のノード}\} + GapY$.

条件 B_5: s がゴールノードのとき，$\pi_x(\text{s の h- 子供}) \geq max\{\pi_x(t) + width(t) \mid t \text{ は } s \text{ の } v\text{- 部分木構造図のノード}\} + GapX$.

条件 B_6: s がヘッドノードのとき，$\pi_x(\text{s の h- 子供}) \geq \pi_x(s) + width(s) + GapX$.

3. Logichart 属性グラフ文法

Logichart 属性グラフ文法は，Logichart 図のグラフ構文規則を順序グラフを生成する edNCE グラフ文法に基づいて定式化したプロダクションと，前節で導入したレイアウト制約条件を満たすノードの座標を計算する意味規則から構成される．Logichart 属性グラフ文法のノードラベルのアルファベットを図 2に示す．エッジラベルのアルファベットは，$\Gamma = \Omega = \{\text{e,h,v}\}$ である．

Logichart 図のノード配置の情報を導出するために次の属性を定義した．

ノードラベル（スタートラベルを除く）の継承属性：

- π_x：ノードの X 座標

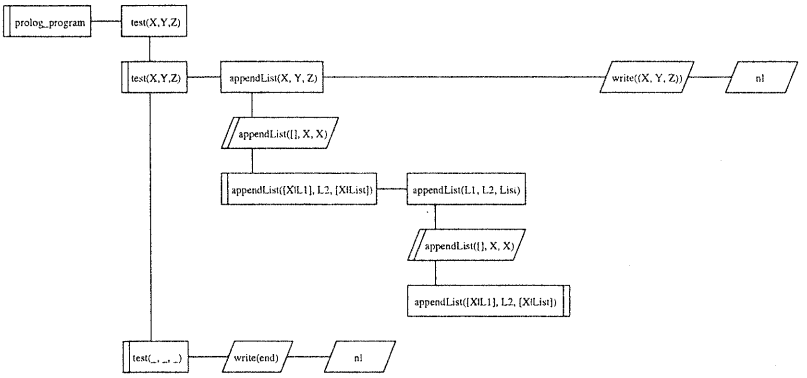


図1 Logichart 図の例
Fig. 1 An example of Logichart diagrams

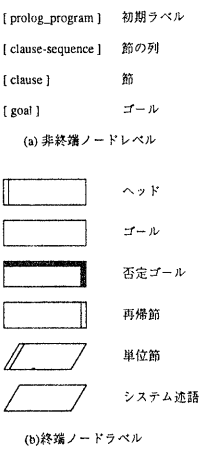


図2 Logichart 属性グラフ文法のノードラベルのアルファベット
Fig. 2 The node label alphabets of the Logichart attribute graph grammar

- π_y：ノードの Y 座標

非終端ノードラベルの合成属性：

- subtree_width：部分木の最大幅
- subtree_height：部分木の最大の高さ

Logichart 属性グラフ文法のプロダクションおよび意味規則の定義中で次の記号および関数を用いる．

- 〈 X 〉：Prolog プログラムにおける X に対応するゴール名の文字列
- "X"：文字列 X
- max(X,Y)：X と Y の最大値を求める関数
- get_width(X)：文字列からセルの幅を求める関数
- get_height(X)：文字列からセルの高さを求める関数

Logichart 属性グラフ文法のプロダクション集合の一部と，各プロダクションに付随させた意味規則集合を図 3から図 9までに示す．各プロダクションの定義の中で，右辺グラフの各ノード右下に添えられた数字はノードを区別する識別子であり，かつ右辺グラフ内でのノードの線形順序も表している．

プロダクション1:

$[\ \text{prolog\_program}\ ]_0 ::=$ "prolog_program" $\xrightarrow[1]{e,h}$ $[\ \text{goal}\ ]_2$,

$C = \emptyset$

意味規則の集合1:

$\pi_x(1) = \text{RootX}$,
$\pi_y(1) = \text{RootY}$,
subtree_width(0) = get_width("prolog_program") + GapX + subtree_width(2),
subtree_depth(0) = max(get_depth("prolog_program"), subtree_depth(2)),
$\pi_x(2) = \text{RootX} + \text{get\_width("prolog\_program")} + \text{GapX}$,
$\pi_y(2) = \text{RootY}$

図3 スタートラベル [prolog_program] に対するプロダクションと意味規則
Fig. 3 The production and semantic rules for the start label [prolog_program]

プロダクション2:

$[\ \text{goal}\ ]_0 ::= [\ \text{goal}\ ] \xrightarrow[1]{e,h} [\ \text{goal}\ ]_2$,

C = {(#,e,e,1,in), (#,e,e,2,out), (#,h,h,1,in), (#,h,h,2,out), (#,v,v,1,in), (#,v,v,1,out)}

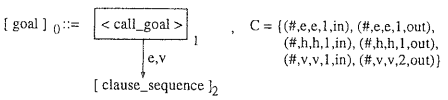
意味規則の集合2:

$\pi_x(1) = \pi_x(0)$,
$\pi_y(1) = \pi_y(0)$,
subtree_width(0) = subtree_width(1) + GapX + subtree_width(2),
subtree_depth(0) = max(subtree_depth(1), subtree_depth(2)),
$\pi_x(2) = \pi_x(0) + \text{subtree\_width}(1) + \text{GapX}$,
$\pi_y(2) = \pi_y(0)$

図4 ゴールの AND に対するプロダクションと意味規則
Fig. 4 Production and semantic rules for conjunctive goals

図 3はスタートラベル [prolog_program] を持つ非終端ノード（スタートグラフ）の書き換えを定義するプロダクションと意味規則である．このプロダクションが導出するルートノードの x 座標と y 座標は，意味規則によりそれぞれ定数 RootX と定数 RootY である．図 5は Prolog のゴール呼び出しに相当する．呼び出しゴールノードの真下に，このゴールと単一化可能なヘッドを持つ節からなる部分木構造図が配置される．システム述語の呼出しは図 6に示すノードラベルを用いて，ユーザ定義述語とは区別して

プロダクション4:

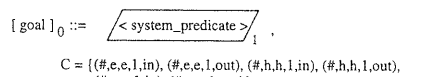


意味規則の集合4:

$\pi_x(1) = \pi_x(0)$
$\pi_y(1) = \pi_y(0)$
subtree_width(0) = max(get_width(< call_goal >), subtree_width(2))
subtree_depth(0) = get_depth(<call_goal>) + GapY + subtree_depth(2)
$\pi_x(2) = \pi_x(1)$
$\pi_y(2) = \pi_y(1)$ + get_depth(<call_goal>) + GapY

図5 ゴール呼び出しに対するプロダクションと意味規則
Fig. 5 Production and semantic rules for goal calling

プロダクション6:

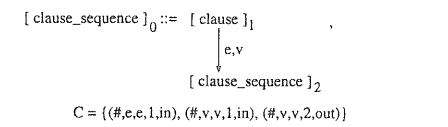


意味規則の集合6:

$\pi_x(1) = \pi_x(0)$
$\pi_y(1) = \pi_y(0)$
subtree_width(0) = get_width(< system_predicate >)
subtree_depth(0) = get_depth(<system_predicate>)

図6 システム述語に対するプロダクションと意味規則
Fig. 6 Production and semantic rules for a system predicate

プロダクション7:



意味規則の集合7:

$\pi_x(1) = \pi_x(0)$
$\pi_y(1) = \pi_y(0)$
subtree_width(0) = max(subtree_width(1), subtree_width(2))
subtree_depth(0) = subtree_depth(1) + GapY + subtree_depth(2)
$\pi_x(2) = \pi_x(1)$
$\pi_y(2) = \pi_y(1)$ + subtree_depth(1) + GapY

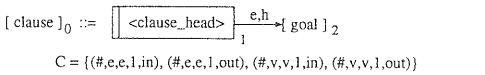
図7 [clause-sequence] を書き換えるプロダクションと意味規則 - 1
Fig. 7 Production and semantic rules for [clause-sequence] - 1

表示する．図 9は再帰的に呼び出された節を書き換える規則である．この再帰節を表すノードラベルを導入したことにより，Prologプログラムの中に再帰的に呼び出される節が存在しても有限領域の木構造図で表示できる．

Logichart 属性グラフ文法のプロダクションの右辺は順序グラフであり，スタートグラフから導出される文形式はすべて順序グラフとなる．Logichart 属性グラフ文法で定義した意味規則は，この順序グラフの線形順序で意味規則の属性評価を行うとすべての属性値が計算できる．Logichart 属性グラフ文法による生成される言語の要素は，'h' と 'v' でラベル付けられたエッジを取り去り，エッジの向きとエッジラベル 'e' を無視することにより Logichart 図となる．一方，'e' でラベル付けられたエッジを取り去ると L-木構造図とみなすことができる．

Logichart 属性グラフ文法により導出される Logichart

プロダクション9:

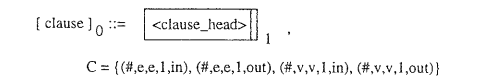


意味規則の集合9:

$\pi_x(1) = \pi_x(0)$
$\pi_y(1) = \pi_y(0)$
subtree_width(0) = get_width(<clause_head>) + GapX + subtree_width(2)
subtree_depth(0) = max(get_depth(<clause_head>), subtree_depth(2))
$\pi_x(2) = \pi_x(0)$ + get_width(<clause_head>) + GapX
$\pi_y(2) = \pi_y(0)$

図8 節を表示するプロダクションと意味規則
Fig. 8 Production and semantic rules for [clause]

プロダクション11:



意味規則の集合11:

$\pi_x(1) = \pi_x(0)$
$\pi_y(1) = \pi_y(0)$
subtree_width(0) = get_width(<clause_head>)
subtree_depth(0) = get_depth(<clause_head>)

図9 再帰的に呼び出された節に対するプロダクションと意味規則
Fig. 9 Production and semantic rules for a recursively called clause

図に対して次の定理が成り立つ．

定理1 Logichart 属性グラフ文法が導出する図は，L-木構造図としてみたとき，レイアウト条件 $B_1 \wedge B_2 \wedge B_3 \wedge B_4 \wedge B_5 \wedge B_6$ を満足し，かつこのレイアウト条件を満たす面積最小の配置となっている． □

4. おわりに

Prologプログラムのためのプログラム図記述言語 Logichart のグラフ構文規則とレイアウト規則を Logichart 属性グラフ文法として定義した．Prologプログラム図に対するグラフ構文規則とレイアウト規則を厳密に定式化する研究は，本研究が初めてのものである．

参考文献

1) 海野浩, 安斎公士, 小倉耕一, 西野哲朗, 中西美智子, 夜久竹夫: 木構造図式の描画問題, 情報処理学会論文誌, Vol.33, No.7, pp.879–886 (1992).
2) J. Engelfriet and G. Rozenberg, "Node Replacement Graph Grammar", in *Handbook of Graph Grammars and Computing by Graph Transformation* (Rozenberg, G., eds), World Scientific (1997), 1–94.
3) Nishino T.:, Attribute graph grammars with applications to Hichart editors, *Adv. Software Sci. & Tech.*, 1, pp.426–433 (1989)
4) 安達由洋, 大井裕一, 大澤優, 二木厚吉, 夜久竹夫: DXL 対応 Hichart プログラム図に対する属性グラフ文法, 日本大学文理学部自然科学研究所「研究紀要」, 第 33 号, pp.149-164 (1998).