A-029

プログラム図を用いたProlog学習支援環境

A System for Prolog Programming Learning using Program Diagrams

古澤　雄大 †　　安達　由洋 †
Yuudai Furusawa‡　Yoshihiro Adachi‡

1. はじめに

Prolog はユニフィケーションや自動バックトラック機能等の特徴的な機能を持つ論理型プログラミング言語で主として知識処理や制約プログラミングの分野で用いられている。一方、プログラムの可視化はプログラムの解読、デバッグあるいは学習を支援する有用な手段である。Prolog プログラムの可視化システムとして、プラグラムを AND/OR 木で記述する Transparent Prolog Machine（TPM）が知られており [1]、Tamir ら [2] は、TPM を視覚的デバッガに拡張した。しかし、これらの研究を含む Prolog の可視化とその応用に関するほとんどの研究には、Prolog の構文規則を反映していない AND/OR 木を用いてプログラムの図表示をしているという大きな欠点がある。

Logichart は Prolog の構文規則と標準的な Prolog 処理系の計算規則（最左ゴール実行、深さ優先探索）を考慮して定義されたプログラム図記述言語である [3]。これらの特徴により、Logichart 図は元のプログラムとの対応も容易に取れ、視認性の高いものとなっている。安達等は Logichart 図の生成規則とレイアウト規則を属性グラフ文法に基づいて厳密に定式化した Logichart 属性グラフ文法を定義した [4]。また、Logichart 属性グラフ文法に準拠した Prolog プログラム可視化システムを IFProlog[5] を用いて実現している [4]。

本研究では、Logichart 属性グラフ文法に準拠した Prolog プログラム学習支援環境を、世界中で広く利用されている SICStus Prolog[6] を用いて構築する。Logichart 属性グラフ文法に準拠することで任意の Prolog プログラムに対して Logichart 図が表示されることと、表示された Logichart 図がレイアウト条件を満足していることが保証される。

2. Logichart[3][4]

2.1 Logichart 図

Logichart は Prolog プログラムの質問に対する応答を視認性の高い擬似木構造図として表現するプログラム図記述言語である。Logichart 図による 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に対し、'prolog_program' をヘッド部に付与した節をプログラムに追加する。この追加した節のヘッド部に対応する'prolog_program' ラベルを持つノードが Logichart 図のルートとなる。Logichart 図ではヘッド部とボディ部のゴール列は水平方向に、あるゴールとそれとユニフィケーション可能なヘッドを持つ節は垂直方向に、それぞれ一列に配置される。また、Logichart 図には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質問'? :- test(X,Y,Z)' を与えたときの Logichart 図を図.1 に示す。

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

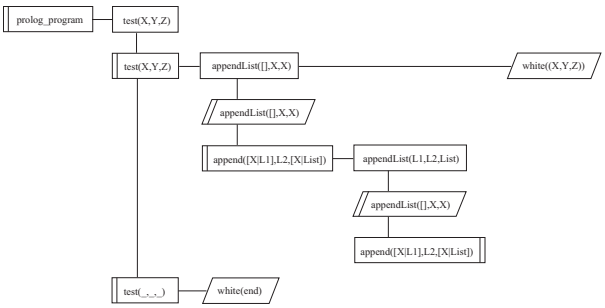

図 1　Logichart 図の例

2.2 Logichart 属性グラフ文法

本論文で用いる Logichart 属性グラフ文法は Logichart 図のグラフ構文規則を属性グラフ文法で定式化したプロダクションと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 属性グラフ文法は 13 のプロダクションと 88 の意味規則からなる。図.2 から図.4 までにそれらの一部を示す。

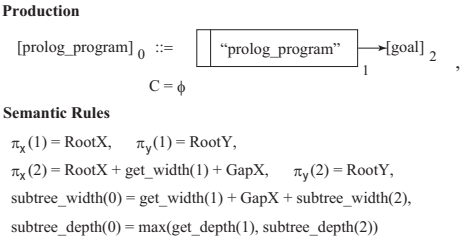

図 2　スタートラベルに対するプロダクションと意味規則

図.2 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の定数 RootX と定数 RootY となる。また、図.2 のプロダクションと意味規則は Prolog におけるゴールの AND に対応し、図.3 のプロダクションと意味規則は

† 東洋大学大学院工学研究科情報システム専攻
‡Department of Open Information Systems Graduate School of Engineering, Toyo University

Production

$[goal]_0$ ::= $[goal]_1 \longrightarrow [goal]_2$,

C = {(#,*,*,1,in), (#,*,*,2,out)}

Semantic Rules

$\pi_x(1) = \pi_x(0)$, $\pi_y(1) = \pi_y(0)$,

$\pi_x(2) = \pi_x(1) + subtree\_width(1) + GapX$, $\pi_y(2) = \pi_y(1)$,

$subtree\_width(0) = subtree\_width(1) + GapX + subtree\_width(2)$,

$subtree\_depth(0) = \max(subtree\_depth(1), subtree\_depth(2))$

図 3　ゴールの AND に対するプロダクションと意味規則

Production

$[goal]_0$ ::= $[goal]_1$, C = {(#,*,*,1,in), (#,*,*,1,out),
(#,*,*,2,in), (#,*,*,2,out)}
$[goal]_2$

Semantic Rules

$\pi_x(1) = \pi_x(0)$, $\pi_y(1) = \pi_y(0)$,

$\pi_x(2) = \pi_x(1)$, $\pi_y(2) = \pi_y(1) + subtree\_depth(1) + GapY$,

$subtree\_width(0) = \max(subtree\_width(1), subtree\_width(2))$,

$subtree\_depth(0) = subtree\_depth(1) + GapY + subtree\_depth(2)$

図 4　ゴールの OR に対するプロダクションと意味規則

Prolog におけるゴールの OR に対応する。これらのプロダクションと意味規則のように、Logichart 属性グラフ文法は Prolog の構文規則を参考に定義されているため、元のプログラムとの対応が取りやすいものになっている。

3. Prolog プログラム開発支援環境

本論文では、Logichart 属性グラフ文法を用いて SICStus Prolog プログラムのビジュアルな学習支援環境を実現する。本システムは、処理部は SICStus Prolog、GUI 表示部は tcl/tk を用いて構築される SICStus Prolog 準拠の学習支援環境であり、プログラム図の描画やビジュアルトレース機能、トレース中の変数の代入値の表示といった機能を持つ。以下に本システムの主要な機能を挙げる。

プログラムの構文解析： 一般的な Prolog 処理系における Consult/Reconsult の処理と同じようにユーザの選択した Plorog プログラムの構文解析を行い、システム内に展開する。すでに展開された節と同じヘッド部を持つ節が読み込まれた場合の処理は SICStus Plorog 処理系に準拠する。

プログラムの可視化： 展開されたプログラムを Logichart グラフ文法に基づいて可視化する。質問の入力を行うことにより prolog_plogram 節を根とする Logichart 図が描画される。

プログラムの編集： 展開されたプログラムのテキストを表示し、編集を行うことができる。テキストは展開されたプログラム全体と個々のファイル毎の表示を行う。変更されたプログラムは再描画コマンドによってプログラム図に反映される。

プログラムのビジュアルトレース： 本システムではプログラム図に対し実行中に呼び出された節に対し成功/失敗に対応する色付けを行い、変数の代入値を表示する事でプログラムのビジュアルトレースを行う。

プログラム図の表示粒度の変更： 大規模なプログラムを可視化する際、プログラムの全てを描画しようとするとプログラム図が非常に大きくなってしまい、プログラム図の視認性が悪くなってしまう。そのため、本システムではあるセルから呼び出されたり展開されるゴールの表示/非表示を切り替え、プログラム図の表示粒度を変更する機能を搭載する。ユーザの注目したい部分のみを細かく表示し、それ以外の部分を隠す事でプログラム図がシンプルで見やすいものになる。

図.5 に本システムのユーザインターフェイスを示す。テキスト、変数の代入値の表示は別ウィンドウになっており、表示/非表示を選択することができる。

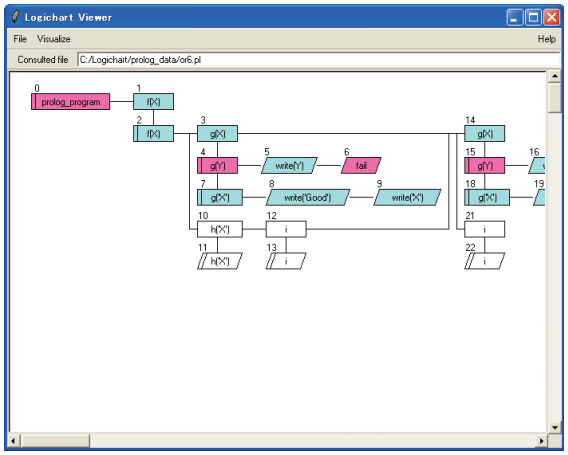

図 5　本システムのユーザインターフェイス

4. 現状と今後の課題

本システムは現在、Consult と構文解析、プログラム図の描画、テキストの表示と編集、ビジュアルトレース、トレース中の変数の代入値の表示といったシステムの骨組みとなる機能を実装している。

今後の課題として、2 つ以上のプログラムの Consult/Reconsul、テキストの変更をプログラム図に反映、プログラム図のプリントアウト、プログラムの表示粒度の変更機能の実装、ユーザインターフェイスの洗練などが挙げられる。

参考文献

[1] Brayshaw, M. and Eisenstadt, M.: A practical graphical tracer for Prolog, J. Man-Machine Studies, 35, pp.597-631 (1991).
[2] Tamir, D.E.: A visual debugger for pure Prolog, INFORMATION SCIENCES, Vol.3, No.2, pp.127-147 (1995).
[3] Y. ADACHI, K. TSUCHIDA, T. IMIKI and T. YAKU.: Logichart - Intelligible Program Diagram for Prolog and its Processing System, Electronic Notes in Theoretical Computer Science, Volume 30, Issue 4, Elsevier Science (2000).
[4] 安達由洋: Prolog プログラム図の属性グラフ文法に基づく定式化, 情報処理学会, 第 33 回数理モデル化と問題解決研究会 (2001).
[5] http://www.ifcomputer.co.jp/IFProlog/home_jp.html
[6] http://www.sics.se/isl/sicstuswww/site/index.html