1Q-8

ＪＳＩ　ＡＩワークステーション（８）
― マルチウィンドゥ・デバッガー ―
ＰＲＯＥＤＩＴ２の機能

森下真一・沼尾雅之・横井伸司

日本アイ・ビー・エム株式会社　サイエンス・インスティチュート

１．はじめに

ＪＳＩ・ＡＩワークステーションの Prolog のプログラミング環境の１つである、マルチウィンドゥ・デバッガー PROEDIT2 を紹介する。PROEDIT2 は過去我々が開発してきた視覚的デバッガー PROEDIT ［１］を整備し拡張したものである。

２．従来のデバッガーとの差異と機能の概略

従来のデバッガーを使用する場合、バグへのアプローチは、探索木を計算(depth first search)がおこなわれる順番に沿ってなされていた(図１)。この際ユーザーはバグに近づく為に、調べなくてもよい探索木の枝を見たり、探索木を深く潜ったりまた浅い方へと戻る、といった作業をくり返す。この過程で、今、探索木の全体でどこをデバッグしているかを見失ったり、あるゴールの何回目の再実行を調べているのかが分らなくなる、といった混乱に陥ることもある。またプログラムをトレースしたとしても、その情報が整理されて出てこないため、役にたたない事も多い。

PROEDIT2 は以上の問題点の解決するため開発されたデバッガーである。PROEDIT2 では図２ａのようにトップレベルのゴールを一度実行した上で探索木を節単位にまとめたビュー(plane と呼ぶ)をユーザーに提供する。plane 中では計算の実行過程が、Prolog の視覚的計算モデル BPM (Box and Plane Model) にしたがって整理され表現される(図２ｂ)。この表現方式では、ゴール間の制御の流れやゴールの再実行の過程が把握しやすい為、意図していないプログラムの動きを見つけやすい。従って無駄な探索木の枝を調べることなく、バグを含む可能性のある枝を正しく選択して、速やかにバグのありかへと到達できる。

このようなデバッグ作業を円滑に行うため、PROEDIT2 では必要となる異なる複数の plane を画面上にウィンドゥとして任意個表示できる。そして任意の箇所に視点を移し、変数束縛の状態やその時点でのルールベースの状態を参照できる。また視点を移す作業は計算過程を遡行的に進むものであっても構わない。

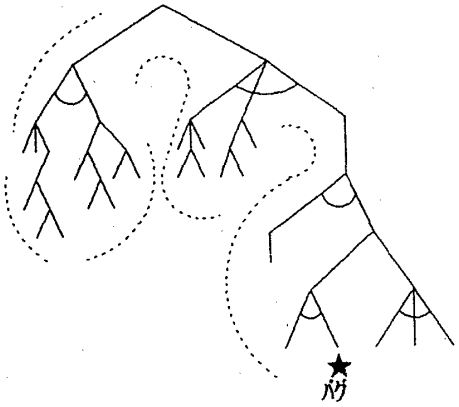


図１．従来のデバッグ作業

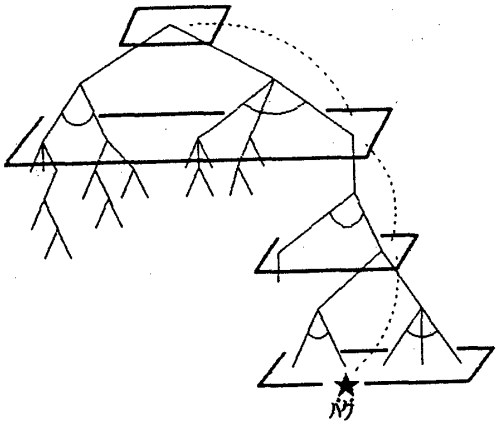


図２ａ．PROEDIT2 によるデバッグ

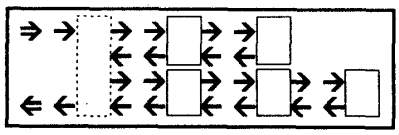
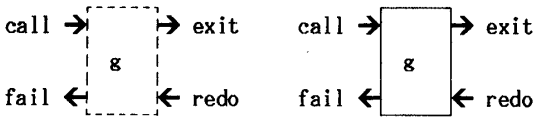


図２ｂ．ＢＰＭによる計算過程の表現

JSI-AI Workstation(8) - Functions of Multi Window Debugger 'PROEDIT2'
Shinichi MORISHITA・Masayuki NUMAO・Shinji YOKOI
Science Institute, IBM Japan, Ltd.

３．計算の実行過程の表現－－－ＢＰＭ

計算の実行過程の表現に使われる視覚的計算モデル BPM [2]について簡単に述べる。BPM では基本的に２種類のボックス head-box, goal-box に４種類の矢印(arrow)を配置して実行過程を表現する。



head-box　　goal-box

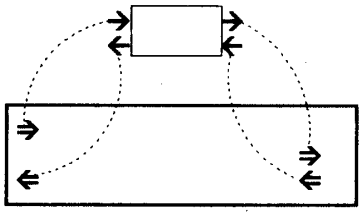
head-box はゴール g(T) とユニファイ可能な節の取出し過程を表現するのに使われる。arrow の意味は、

call:　初めにユニファイ可能な節を要求する
exit:　ユニファイ可能な節を返す
redo:　次にユニファイ可能な節を要求する
fail:　ユニファイ可能な節が尽きたことを示す

goal-box はゴール g(T) の実行過程を表現する。arrow の意味は、

call:　実行の要求する
exit:　実行結果 g(T)θをかえす。θはユニファイヤー
redo:　再実行の要求する
fail:　実行は失敗したことを示す

plane は goal-box 内部の実行の動きを head-box と goal-box を２次元的に配置して表現したものである(図２ｂ)。goal-box に配置された arrow は、plane に於ては二重の arrow に書換られ plane の入口と出口を正確に認識することができる。



４．PROEDIT2 の機能

先に述べたように PROEDIT2 の機能には大きく次の２つがある。

- 任意個の調べる必要となる plane を画面上にウィンドゥとして表示できる
- 表示された plane の任意の arrow へ視点を移し、変数の束縛状態、計算全体での視点の位置、その時点でのルールベースの状態を参照できる。

この機能を実現するウィンドゥが４種類ある(図３)。

Type1.　plane を表示するウィンドゥ
Type2.　視点のある arrow の情報を表示するウィンドゥ
　　－－変数の束縛状態を知ることができる。例えば図３におけるようにゴール g(T) の exit arrow に視点を移すと(⇛で示されている)、
　　exit: g(T)θ　　[θはユニファイヤー]
　　が表示される。
Type3.　計算全体での視点の位置を示すウィンドゥ
　　－－トップレベルの plane から視点のある plane の間にある plane の名前を表示する。
Type4.　ルールベースの状態を表示するウィンドゥ
　　－－視点のある arrow の時点でのルールベースの状態を表示する。

５．まとめ

PROEDIT2 は現在ホスト上の VMPROLOG(VM/ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ･ｲﾝ･ﾛｼﾞｯｸ, IBMのProlog処理系)で開発されており、今後はワークステーション上の Prolog 処理系が開発された時点で、それへの移植を考えている。

参考文献

[１]　沼尾　PROEDIT - A Screen Oriented Prolog Programming Environment　ﾛｼﾞｯｸ･ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ･ｺﾝﾌｧﾚﾝｽ 1985
[２]　森下・沼尾　Prolog の視覚的計算モデル BPM　ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 言語研究会　資料 86-PL-7

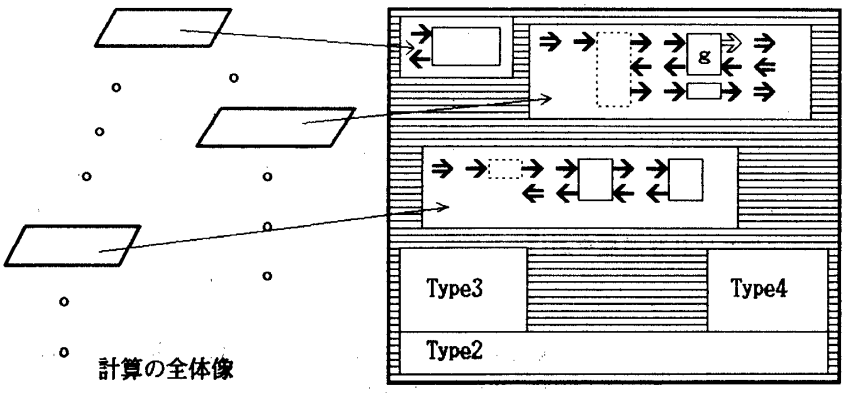


図３．　PROEDIT2 の画面表示