4M-7

グラフィック表現における Prolog実行過程のアニメーション化

四野見　秀明
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

前回の報告[1]ではPrologプログラムの構造（述語呼出し）や論理的な関係（and, or）をグラフィック表示することによりプログラムの解析を支援するシステムを提案した。今回はプログラムの静的な解析に加えて、実行時の述語呼出しをグラフ上に動的に表示することによるプログラム解析支援の機能を提案し、実現したので報告する。

2．実行過程アニメーションのねらい

Prologプログラムの静的な解析結果のグラフィック表示により、述語間の関係や再帰構造などをディスプレイ上で知ることができる。しかし、プログラムにあるデータが与えられた場合にプログラムのどの部分が起動され、どの部分が実行に大きくかかわるかを知ることができない。また、表示されたグラフを見ることで、個々の節（clause）を書いた時には予想しえなかったプログラムの全体構造を知り、無限ループを生じうる再帰呼出しを発見することができる。しかし、どの再帰呼出しが無限ループの原因かを静的な解析グラフを見ただけでは知ることは不可能である。また、バックトラックにより生じる止まらない繰り返しについては、プログラムの構造だけでは生じうる部分を知ることさえできない。

そこで、Prologプログラムへの質問（question）が起動されてからサブゴールが起動されるたびに、その述語に相当するグラフ上のノード（ボックス）内の述語名を次々とハイライトすることでPrologの実行過程アニメーションを行うシステムが有効であると考えた。その機能によりプログラムの実行部分や繰り返し部分が明らかになる。

現在、ほとんどのPrologシステムには箱モデル（box model）[2]に基づいたトレース機能があり、実行過程を出力することができる。しかし、それらのトレース機能では述語の呼出し、成功、失敗に関して知ることができる反面、出力情報が詳細すぎ、ディスプレイ上で一度に把握できるプログラムの動作のスコープが小さ過ぎる。一画面で見ることができるトレース結果は、数個の述語呼出しに関する情報に限られてしまう。グラフ上でトレースに相当する情報をアニメーションすることで、プログラムの構造と動作を同時に見ることができ、繰り返しなどの過程を感覚的に捉えることにも役立つ。

3．システムの構成と実現

表示されたグラフ上でPrologの実行過程を動的に見せようとする場合、実行履歴を全て保存し実行後グラフ上で再現する方法も考えられる。しかし、止まらないPrologプログラムには、その方法を用いることはできない。したがって、インタープリタがサブゴールを起動するたびに、同時にその述語に相当するグラフ上のノードをハイライトする方法を採用した。

前回報告[1]したグラフを表示するブラウザの環境はPL/Iで実現されていて、VM/Prologで書かれたPrologのステップ実行インタープリタとの間でFig.1 で示された以下のようなやりとりが行われる。

まずブラウザの環境で、答えを求めるべきゴールが与えられるとインタープリタを起動する。サブゴールが発見されるとインタープリタは停止しサブゴールの情報をブラウザに渡す。ブラウザは、その情報にしたがってグラフ上のハイライトを移動する。その後、ブラウザが強制的にバックトラックを生じさせると、インタープリタは実行を再開し、次のサブゴールを探し始める。この繰り返しを初めに与えられたゴールが成功、失敗あるいはあらかじめユーザによりセットされた実行ステップ数に達するまで続ける。このブラウザとステップ実行インタープリタのコミュニケーションによりアニメーションが実現されている。

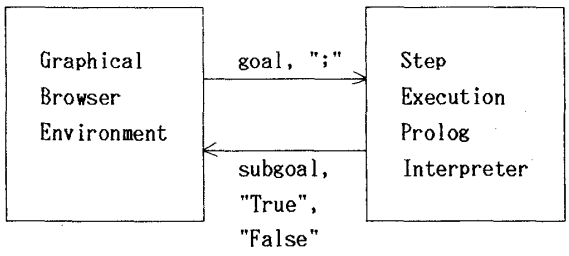

Fig.1 Communication between
Graphical Browser Environment and
Step Execution Prolog Interpreter

Animation of Prolog Execution on Graphical Representation
Hideaki SHINOMI
Science Institute, IBM Japan, Ltd.

４．システムの機能

前回報告した機能のほかに、アニメーションによるプログラム解析支援のために以下の機能が追加された。

1）アニメーションと同時に、サブゴールがディスプレイに表示され、インスタンシエイトの詳細を知ることができる。同時に、"call"、"redo"も表示される。

2）呼ばれた述語を表わすハイライトは、省略時にはサブゴールが"call"の時は黄色、"redo"の時は緑色といったように異なった色のリバースが用いられ、ひとめで区別がつく。

3）グラフの全体像とメインウインドウのスコープを表示しているサブウインドウ中でも、同時にハイライトの移動が確認でき、メインウインドウのスコープから外れた見えない部分での実行過程も小さいながら確認できる。

4）サブウインドウ中のグラフ上では、一度でも通過した部分はその痕跡が色の違いとして残され、実行の軌跡を見ることができる。

5）停止せずにアニメーションを続ける回数を自由に設定できる。その実行ステップ数ごとにハイライトの移動が一時停止する。この機能により止まらないPrologプログラムに対しても、システム自身の実行が終わらない状況を避けることができる。

6）実行ステップ数の指定によりアニメーションが一時停止した時、メインウインドウのスコープを変え、ユーザの関心のある場所をズームし、実行を継続できる。

7）Graph-Ⅰ(述語呼出しの関係)、Graph-Ⅱ(論理的関係)の両方のタイプのグラフ上で実行可能である。

５．適用例(例 構文解析プログラム)

構文解析プログラムに対して本システムを適用した例をFig.2、Fig.3に示した。メインウインドウの下の"==>"から始まる行はコマンド・ラインでPrologの質問が入力される。メインウインドウの上の行はメッセージ・ラインであり、起動されたサブゴールが"call"または"redo"と共に出力される。そのすぐ左の数値は、コマンド・ラインに入力された質問に対してのその時点での述語呼出しの回数である。

Fig.2 には、ユーザの実行ステップ数の指定により一時的に停止した画面のハード・コピーを示した。その時点で関心のある部分をズームした後、実行を再開したのがFig.3 である。ステップ数が16から17へと増加し、ハイライトが移動しているのがわかる。

６．まとめ

Prologプログラムの構造をグラフィカルに表示する機能に加え、グラフ上で実行過程をアニメーションする機能を提案した。Prologプログラミングで頻繁に生じる、個々の節を書いている時には思いもよらないループや繰り返しの把握をアニメーションが支援してくれる。実行ステップ数を自由に変えることによって、プログラムの重要な部分の実行では１ステップごとの詳細な実行の確認ができる。感覚的な動作の把握と詳細にわたる実行の把握の両方が可能なわけである。

これらの機能を用いれば、既存のプログラムに対してもまず実行に重要な部分を把握してから、プログラムの構造を意識しながらプログラムの詳細を解析できる。本システムを用いることで、Prologプログラムの保守、デバッグ、その他を支援することができる。

参考文献

[1] 四野見 秀明，『グラフィック表現を用いたPrologプログラム保守支援ツール』，情報処理学会第32回全国大会，1986.

[2] W.F. Clocksin, C.S. Mellish 著，中村 克彦 訳，『Prologプログラミング』，マイクロソフトウェア，1983.

[3] 中島 秀之，『Prolog』，産業図書，1983.

[4] VM/Programming in Logic, Program Description/Operations Manual.

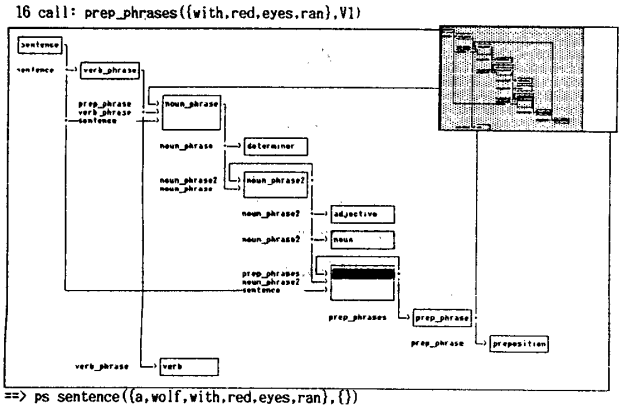

Fig.2

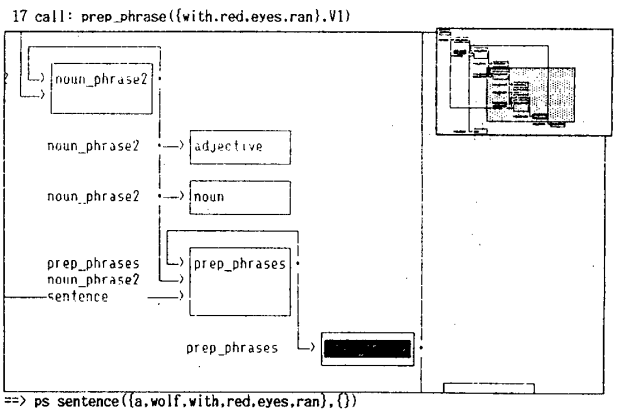

Fig.3