1K−4

GHC導出支援システム

田中二郎　富士通・国際研
的野文夫　富士通SSL

1. はじめに

並列論理型言語GHC は、論理型言語Prologに並列実行の機能を持たせたものである［Ueda 85］. GHC を並列環境で実行すると、ゴールの実行順序、節の選択、またそれに伴って生ずる変数の束縛などに再現性がなく、またユーザはそれを見ることが出来ない. そこで我々は、『原点』に戻り、GHC プログラムの実行を、論理式の導出であると考え、その過程を視覚的、かつ対話的に支援する『GHC 導出支援システム』を開発した.

『GHC 導出支援システム』は以下のような特色をもっている.

①［プログラムの実行過程表示］
GHC プログラムの実行過程を木構造で表示する. 実行過程のシミュレートは、GHC によるGHC のメタインタプリタをエンハンス（富裕化）することにより実現している.

②［対話的インタフェイス］
ゴールの実行順序や節の選択をユーザが対話的に実行し、それにより導出されるゴールや、それに伴って生ずる変数の束縛などを表示するインタフェイスを提供している.

③［メタの階層、リフレクション機構］
メタインタプリタを多層段組合わせることにより、オブジェクト・レベル、メタ・レベル、メタ・メタ・レベル、... といった、多層段からなる推論システムを実現している. またメタの階層を動的に動くリフレクション機構を実現している.

2. GHC 導出支援システムの概要

本システムは、逐次型推論マシンPSI-Ⅱ上のGHC システムでインプリメントした. すなわち、本システムは、まずPSI-Ⅱを立ち挙げて、そこでGHC システムを動かし、その上でアプリケーション・プログラムとして本システムを起動する.

本システムを実行している時のビットマップ・イメージを図1に示す. すなわち、本システムを起動すると、画面の左下側にI/O ウインドウが表れるのでユーザは、そこからコマンドを入力する. ここでは、まずappendの定義を含む"app.ghc" というファイルをコンサルトしており、その結果として、success(コンサルトが成功した）というレスポンスを得ている. このとき、I/O ウインドウの右横のpredicatesウインドウに定義された述語（この場合はappend/3）が示される. ここでは、参考までにappendの定義を、画面右下のpmacs ウインドウ上に表示してある.

次に、I/O ウインドウで、snlw(append([1,2], [3], X))というゴールを実行しており、これにより、画面左上に位置したSNL Resolution tree Windowの上で、append([1,2], [3],X) ゴールがユーザとの対話により解かれる. SNL (Selective Negative Linear) Resolutionの言葉は［Nagao 83］から採用したが、これは、いわゆるSLD Resolutionの事である.

SNL Resolution tree Windowの上では、プログラムの実行過程が木構造（実行木）で表示される. ユーザは、導出したいゴールを、実行木の中からマウスで選択する. これは導出が終わってないゴールであればどれでも自由に選択できる.

ゴールが選択されると、画面中央にclause choice ウインドウが開き、登録された定義節の中から、そのゴールに適用される可能性のある節（この場合にはappendの定義二つ）が表示される. この表示でも分かるように、定義節の変数は、@(n)という特殊な形式（基底表現）で格納されている.

ユーザは、その中から適用したい定義節を選択する. すると、新しく導出されたゴールが実行木に付け加えられる. 実行木のすべての枝がtrueになると、実行は成功して終了する.

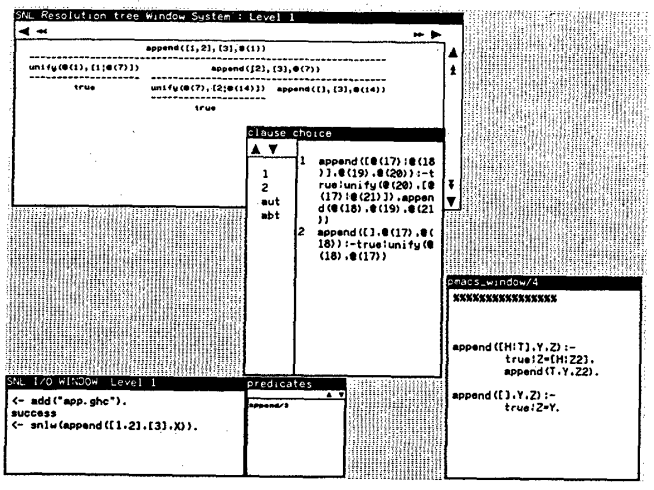

図1　GHC 導出支援システムの実行時のビットマップ・イメージ

GHC Deduction System
Jiro Tanaka,[1] Funio Matono[2]
1) Fujitsu Limited, 2) Fujitsu Social Science Laboratory Limited

3. リフレクション機能の実現

メタ機能、リフレクション機能については、最近とみに関心が高まっている［Tanaka 88 ］．ここでは、とくにFOL［Weyhrauch 80］風のリフレクションを、GHC 導出支援システム上で実現することを試みた．

本システムでは、プログラムの実行過程表示を、GHC によるGHC のメタインタプリタを拡張することにより実現しているが、メタインタプリタを多層段組合わせることにより、オブジェクト・レベル、メタ・レベル、メタ・メタ・レベル、．．．といった、多層段からなる推論システムを実現することができる．この場合、オブジェクト・レベルとはユーザからのゴールを推論（実行）するレベルであり、メタ・レベルとは、オブジェクト・レベルの推論についての推論を行うレベルと考えることができる．同様に、メタ・メタ・レベル、メタ・メタ・メタ・レベル、... と理論的には無限段からなる推論システム（リフレクティブ・タワー）を考えることができる．［Smith 84］

ここで、リフレクション機構とは、こうしたメタの階層を動的に動く機構のことで、リフレクト述語の使用により、例えばオブジェクト・レベルから動的にメタ・レベルに上がり、あるゴールを実行したあと、再びオブジェクト・レベルに戻ることができる．図2にリフレクト述語の実行例を示す．ここでは、まず画面の左下側のI/O ウインドウから、append_merge というゴールを実行している．(append_merge の定義は、参考までに画面右下のpmacs ウインドウ上に表示してある.)

それにより、まずオブジェクト・レベル（レベル1）のSNL Resolution tree Window が開かれ、ユーザは、まずオブジェクト・レベルのゴール実行を行うことができる．通常のゴールは、通常通り処理されるが、リフレクト・ゴールを選択した場合には、リフレクト・ゴールとはメタ・レベルで実行されるべきゴールと考えられるので、メタ・レベル（レベル2）のSNL Resolution tree Windowが動的に作られる．メタ・レベルの推論もオブジェクト・レベルの推論とまったく同様に行われ、実行木のすべての枝がtrueになると、メタ・レベルの実行は成功して終了し、オブジェクト・レベルのゴール実行に戻る．

4.まとめ

本論文では、GHC プログラムの実行を、論理式の導出であると考え、その過程を視覚的、対話的に支援する『GHC 導出支援システム』について述べた．本システムの実行イメージの詳細については［Tanaka 90 ］を参考にされたい．

また、本システムの持つリフレクション機構について述べた．こうしたメタ機能、リフレクション機能については、今後、一層の研究が必要と考えている．

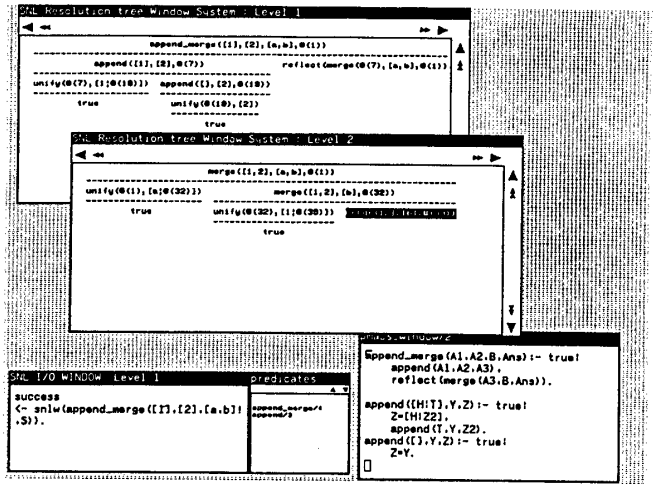

図2　リフレクト述語の実行

［謝辞］

本研究は第5 世代コンピュータ・プロジェクトの一環として行われたものである．本研究に関連して日頃ディスカッションの相手になってくれる国際研の同僚である神田陽治、前田宗則の両氏に感謝する．また富士通SSL の太田祐紀子氏に感謝する．

［参考文献］

［Nagao 83］　長尾真、淵一博：論理と意味、岩波講座、情報科学７巻、1983年．

［Smith 84］　B.C. Smith: Reflection and Semantics in Lisp, 11th. POPL, Salt Lake City, Utah, pp.23-35, 1984.

［Tanaka 88 ］　田中二郎：メタプログラミングとリフレクション，bit, Vol.20, 5, pp.41-50, 1988.

［Tanaka 90 ］　田中二郎：メタ・リフレクション実験環境FE '92、第5 世代コンピュータに関するシンポジウム、デモンストレーション資料、ICOT、1990.6.12-13.

［Ueda 85 ］　K. Ueda: Guarded Horn Clauses, ICOT Technical Report TR-103, 1985.

［Weyhrauch 80］　R. Weyhrauch: Prolegomena to a Theory of Mechanized Formal Reasoning, Artificial Intelligence, Vol.13, pp.133-170,1980.