VISTA: 並列論理プログラム変換/可視化システム

VISTA: A VISualization and Transformation Apprentice for Parallel Logic Programming

奥村 晃, 藤田 博, 上田 和紀, 長谷川 隆三 (ICOT)
吉田 紀彦 (九大), 相川 聖一, 小野 越夫 (富士通)

VISTA は GHC 等並列論理型言語によるプログラミングを支援するために構築中のシステムである. GHC の UR-set 等に基づくプログラム変換/部分計算のツールを中核とする一方, GHC プログラムの記述するプロセス構造や実行状況をビットマップディスプレイ上に可視化するツールを装備し, ユーザフレンドリなプログラム作成環境の実現を目指している. VISTA の現状と今後の展望について述べる.

1. はじめに

ICOT では, 並列論理型言語 GHC をベースに, 仕様記述・検証, 定理証明, プログラム変換, 部分計算, プログラム可視化等の要素技術を統合した並列プログラミング支援システムを検討中である. VISTA はこれらの技術の内, プログラム変換, 部分計算およびプログラム可視化技術をツール化した実験システムであり, 当面並列プログラミングの教育や並列プログラムのデバグ支援等への応用を目指している. VISTA の全体構成を図1に示す. 本システムは, プログラム変換/部分計算器に加えてプログラムの静的構造及び動的振舞を GHC のプロセス (ゴール) レベルで可視化する機能を備えており, レイヤードストリームプログラミングなどの並列プログラミングパラダイムの説明, 部分計算によるプログラム構造の変換過程の説明など, 様々な用途に用いることができる. 本稿では, 現在作成中の VISTA について, 主な要素技術を紹介する.

2. GHCにおけるプログラム変換

VISTA の中核を成すのは, 展開/畳込み操作を基本とするプログラム変換ツールである. 変換規則は古川ら[1], [2] によって提案された UR-set を実装している. GHC プログラムでは展開を不用意に行うとプログラムの振舞いが変わってしまう場合があるので, 適用条件が制限されるが, UR-set では安全な展開を保証する規則を2つ与えている.

一つは「即時実行可能ゴールの展開」と呼ばれるものである. 展開しようとしているゴールの定義節の全てについて, ガードが既に満足されているか若しくは失敗しているかのいずれかである場合には展開が可能である.

もう一つは「条件分岐」と呼ばれるものである. これは通常の展開の様に単一のゴールにおいて展開するのではなく, 被展開節の全てのボディゴールにおいて各々展開された節を集め, その全体で被展開節を置換える. 全ボディゴールについて展開した節を用意することによって, どのボディーゴールから計算が開始されても元のプログラムで可能なあらゆる場合が変換後でも網羅されていることになる.

変換規則としてはこの他に節内の単一化ゴールの計算を行う「標準化」, 「畳込み」の2つがある.

3. GHCプログラムの部分計算

他言語と同様, GHC プログラムにおいても部分計算は有用である. 即ち, GHC プログラムを既知の部分入力を用いて部分的に実行し, 残りの入力に対して冗長な計算をしないプログラムを得ることができ, 汎用プログラムを個別の用途毎に最適なプログラムに特化するなどの目的に応用可能である. 部分計算は基本的には上述のプログラム変換規則を適用して行われるが, ガードに起因する制約の利用[3] 等の工夫を施している.

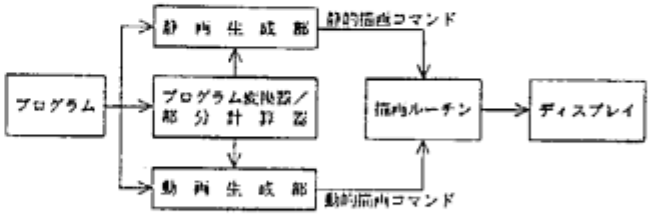

図1 VISTA の構成

GHC プログラムの部分計算で特に注意を要するのは非決定性の扱いである. 原則的には非決定性はプログラム変換/部分計算の前後で保存されなければならない. プロセス単位の入出力の値だけでなく, 入出力のタイミングが全体としてのプログラムの意味を変える場合があるからである. しかし, プログラムによっては実行時の don't care 非決定性が部分計算時に解消されてもユーザの意図に反しない場合がある. 例えば, GHC プログラムでよく使われるストリームの併合操作 merge が

```
merge([H|X],Y,Z):-true|Z=[H|W],merge(X,Y,W).
merge(X,[H|Y],Z):-true|Z=[H|W],merge(X,Y,W).
merge([],Y,Z):-true|Z=Y.
merge(X,[],Z):-true|Z=X.
```

と定義されており, プログラム中で merge([],Y,Z) が呼ばれているとする. UR-set に従えばこのゴールは展開できない. 3番めの節は充足されているが, 2,4番めの節も候補となっているからである. 実際, 上のゴールの呼出し時点で Y が具体化されたストリームであると, 2,3 節あるいは 3,4 節の組合せの中でいずれがコミットされるかは非決定的である. Z の値だけを問題にしている場合は, いずれにしても Y に等しくなるから上のゴールを Z=Y に置換えてよい. このような置換えは, 特にユーザが指示したものに限り適用することにしている.

4. VISTAにおける可視化技術

効率の良い GHC プログラムを作成するには, 複数のプロセスが滞りなく計算を進められるようにプロセス間のデータの流れを調整する必要があるが, 並行して走るいくつものプロセスからなる計算の全体像を掴むことは容易ではない.

またプログラム変換や部分計算の効果についてもプログラムテキストから理解するには相当の努力が必要で

ある．特に変換後のプログラムについては，システムによって適当に生成された変数名や新述語名を含むため，元のプログラムよりも可読性で劣ることが多い．

そこで我々はGHCプログラムの理解の助けとするため，プログラムの計算の様子をグラフィック表示することを考えた．それがVISTAの可視化機構である．

VISTAの可視化機構はPSIのビットマップディスプレイを用いた描画システムとして実現されている．システムの機能は大別して，プログラムの静的構造を示す静的表示と，プログラムの実行の様子を動画で示す動的表示の2つがある．どちらも基本的には，矩形を用いてゴールを示し，矩形の間を直線で結合してゴール間の変数共有を示すものである．

静的表示ではまずユーザが述語毎に描画の指示を行い，解析ルーチンは指示で要求されているレベルまでプログラムの構造を解析し，相対的配置を決める．それに基づいて描画ルーチンは表示の大きさや場所を決定し共有変数を示す線とともに表示する．

動的表示では元のプログラムに描画ルーチンの呼び出しゴールを挿入し，そのプログラムを実際に実行することにより描画ルーチンに必要な情報（ゴールの分割，消滅など）が送られ，描画ルーチンはその情報に基づいて逐一表示を更新して動画を得る．

可視化にあたっては考慮すべき点が多々存在する．まず，選択的な表示機能である．プログラムの実行時に発生するプロセスや変数の数は一般に膨大なもので，当然その全てを表示することは不可能である．そのため表示すべきものを選んで表示しなくてはいけないが，現在はユーザがコマンドで指定する方式で考えている．

また，どのような矩形の配置や結線が見易いかというのも大きな問題であるが，これには今のところ決定的な方法はない．現在はデータの依存関係の有無のみで判断して，依存関係のある場合にはデータの流れに従って左から右に，依存関係のない場合には縦に配置することにしている．

5．可視化例

例として並列パーサMeta-PAXを扱う．Meta-PAXは与えられた入力文と文法から可能なあらゆる構文木をボトムアップに構成する汎用の構文解析インタプリタである．文法が固定された段階で，この汎用Meta-PAXはその文法専用のパーサに特化される．Meta-PAXのプログラム部分は次のように与えられている．

```
expand1(Cat,InS,OutS):-true|
  rules(Cat,Rules),
  expand1_1(Rules,InS,OutS).
expand1_1([Head|Rules],InS,OutS):-
    atomic(Head)|
  expand1(Head,InS,OutS1),
  expand2(Head,InS,OutS2),
  expand1_1(Rules,InS,OutS3),
  merge(OutS1,OutS2,OutS3,OutS).
expand1_1([(R->Head)|Rules],InS,OutS)
    :-true|
  expand1_1(Rules,InS,OutS1),
  merge([(R->Head)*InS],OutS1,OutS).
expand1_1([],_,OutS):-true|OutS=[].
```

文法は次のように与えられる．

```
rules(v,R):-true|
  R=[vp,(np->vp),(adv->vp)].
```

この文法規則について特化されたプログラム部分は次のようになる．

```
expand1(v,InS,OutS) :- true |
  expand2(vp,InS,OutS2_226),
  OutS=[(np->vp)*InS,(adv->vp)*InS
        |OutS2_226].
```

部分計算前後のMeta-PAXパーサの動作はVISTAの動的表示機能を用いて図2のように表示される．

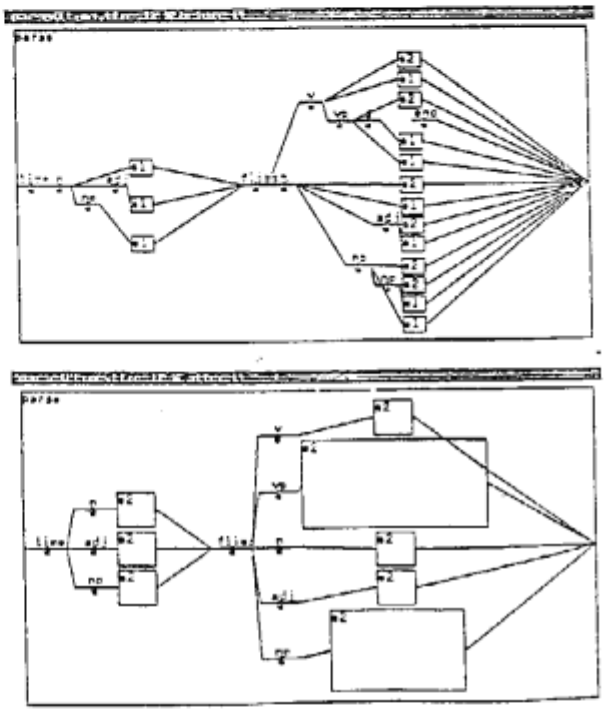

図2 部分計算前(上)，後(下)のMeta-PAXの動作

6．今後の展望

本論文では，並列論理プログラムの変換／可視化システムVISTAの主要技術について述べた．現版のVISTAでは，プログラム構造の可視化方式として，トランスレータを介してプログラム中に可視化メソッドを埋め込む方式を採っているが，プログラム部分と可視化部分の分離を図るため，部分計算による可視化メソッドのコンパイル方式の導入を検討中である．今後は，より理解性に優れた視覚化技法の検討を進めるとともに，制約プログラミング等のパラダイムをも扱えるシステムへ発展させていく予定である．

参考文献

[1] K. Furukawa, A. Okumura and M. Murakami, Unfolding Rules for GHC Programs, New Generation Computing, 6(1988)

[2] K. Ueda and K. Furukawa, Transformation Rules for GHC Programs, FGCS'88, Tokyo, 1988

[3] H. Fujita, A. Okumura and K. Furukawa, Partial Evaluation of GHC Programs Based on the UR-set with Constraints, LP'88, Seattle, 1988